2019

Tweet Your Sentiment

Kevin Nguyen, n9463933

Quang Huy Tran n10069275

25/10/2019

CAB432

Assignment 2

Contents

1.0 Introduction

The proposed application will feature a way for users to view the most common search terms commented by users on Twitter. Twitter is a microblogging and social networking service where users post and interact with messages known as "tweets". These tweets can spark opinions of interest relating highly debated topics, allowing other users to discuss, debate and educate others via these tweets. Hashtags are utilised in tweets to categorize that tweet as a topic to allow other users easier access to tweets with the same hashtag.

Interestingly, Twitter has a reputation of displaying dominant perspectives and for its users to also express these common perspectives. Thus this app will allow users to view the most common keywords being discussed in a Twitter hashtag. Upon a users query of a hashtag, tweets will stream in with sentiment analysis being performed on the text to determine if they have positive or negative sentiment. For the respective sentiments, the most common words will be displayed on a bar graph for the user to see.

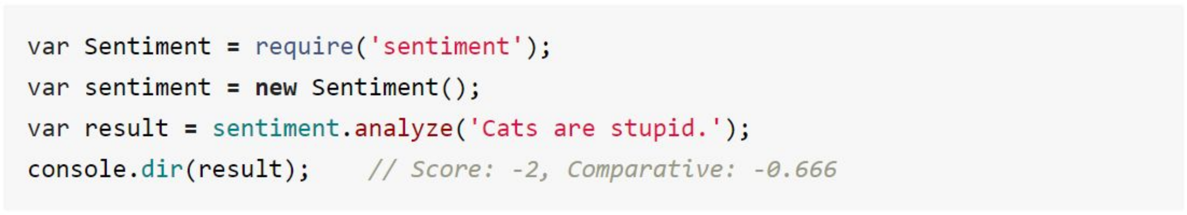
2.0 Services Utilised (APIs and Technologies)

2.1 Twitter Stream API:

To gather data for sentiment analysis on tweets, Twitter's streaming API (status/filter) was utilised to connect to Twitter's endpoint which can provide tweets in real time. To connect to the streaming API, a long HTTP request is made where the stream is indefinitely consumed. Tracking keywords are either provided or chosen by the user to specify the keywords to be tracked by the streaming API.

2.2 Node Sentiment Analysis:

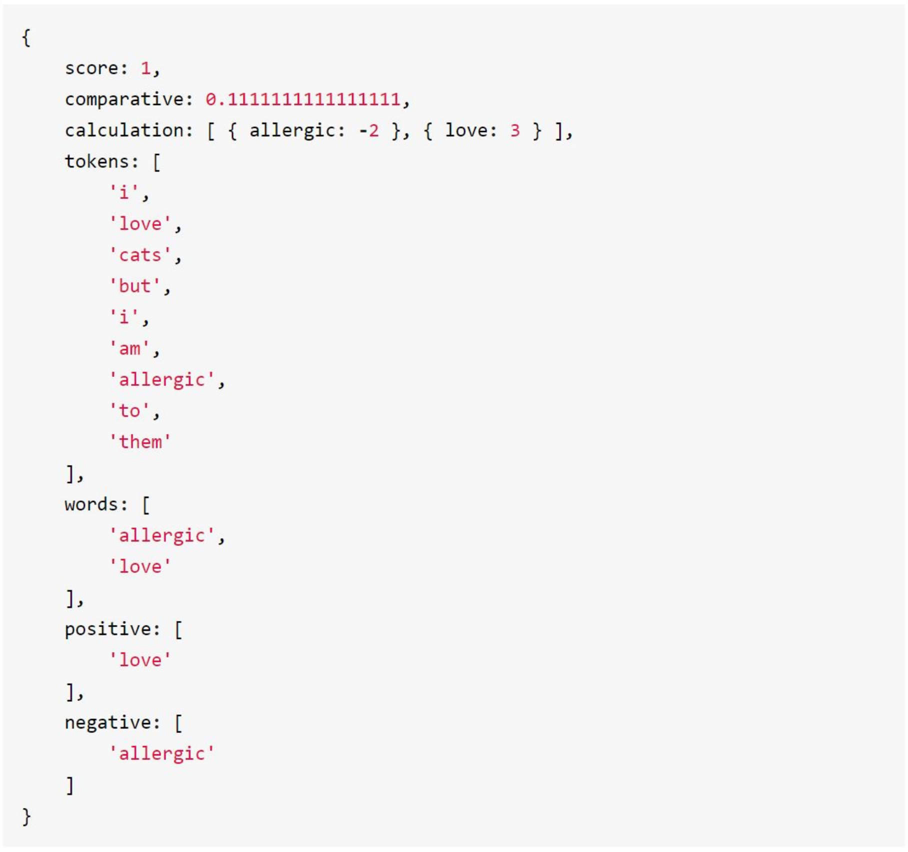
The Sentiment module in Node.js was utilised to assess the sentiment of tweets for the application. The module uses the AFINN-165 word list and Emoji Sentiment Ranking to analyse the sentiment ranking for arbitrary blocks of text input. An example of the usage of this module is shown in the screenshot below.



```
var Sentiment = require('sentiment');
var sentiment = new Sentiment();
var result = sentiment.analyze('Cats are stupid.');
console.dir(result);    // Score: -2, Comparative: -0.666
```

Figure 1: Node Sentiment Module Usage Example

AFINN is a list of words rated for valence with an integer between negative five (negative sentiment) and positive five (positive). Sentiment analysis is performed by cross-checking the string tokens (words, emojis) with the AFINN list and getting their respective scores. The comparative score is simply: sum of each token / number of tokens. An example is displayed below where tokenization takes place on the input string and the sentiment of the text is returned. This app will basically only use the score provided.



```
{
    score: 1,
    comparative: 0.1111111111111111,
    calculation: [ { allergic: -2 }, { love: 3 } ],
    tokens: [
        'i',
        'love',
        'cats',
        'but',
        'i',
        'am',
        'allergic',
        'to',
        'them'
    ],
    words: [
        'allergic',
        'love'
    ],
    positive: [
        'love'
    ],
    negative: [
        'allergic'
    ]
}
```

Figure 2: Explanation of AFINN

2.3 D3Js Visualisation Library:

D3.js is a Javascript library utilised to create dynamic and interactive data visualisations in web pages. The data visualisation option utilised for this web application is their lollipop chart as shown in the figure below. D3.js was chosen to display the most common words in the tweets with positive and negative sentiments respectively for a particular topic on Twitter. Initially, it was decided that a word cloud would be utilised however, the usage of word clouds are dissuaded as they do not clearly signify the difference in the proportional difference between different words and the numerical value of single words.

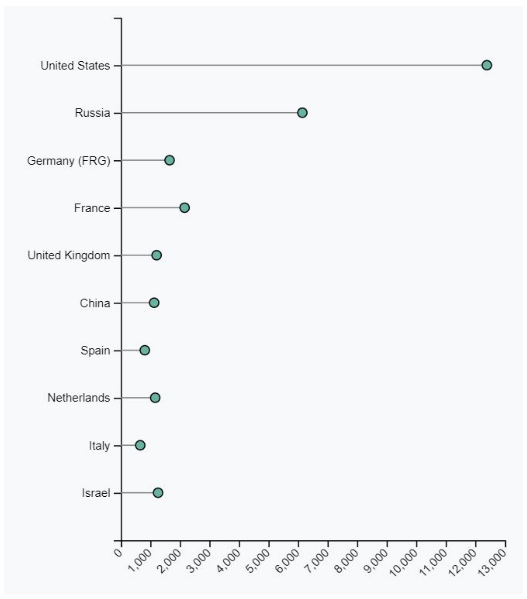


Figure 3: D3.js lollipop chart example

2.4 Azure Cosmos DB

Cosmos Database (DB) is a horizontally scalable, globally distributed, fully managed, low latency, multi-model, multi query-API database for managing data at large scale. Cosmos DB is a superset of Azure Document DB and is available in all Azure regions. Each Cosmos DB contains a container which is horizontally partitioned, replicated across multiple regions and can have fixed or unlimited collections. Cosmos DB supports Document database model and SQL API, which provide a JSON document model with SQL querying and JavaScript procedural login. The SQL API allows us to store and access data from a Node.js application quickly.

In our project, two containers of project database in Cosmos DB are used to store extracted tweets and trend keywords. After calculating the sentiment score of each tweet, it is saved in Cosmos DB as a collection with a tweet ID and a tag based on the keyword provided. In terms of trend keywords, each keyword is used to create the id of the new trend collection.

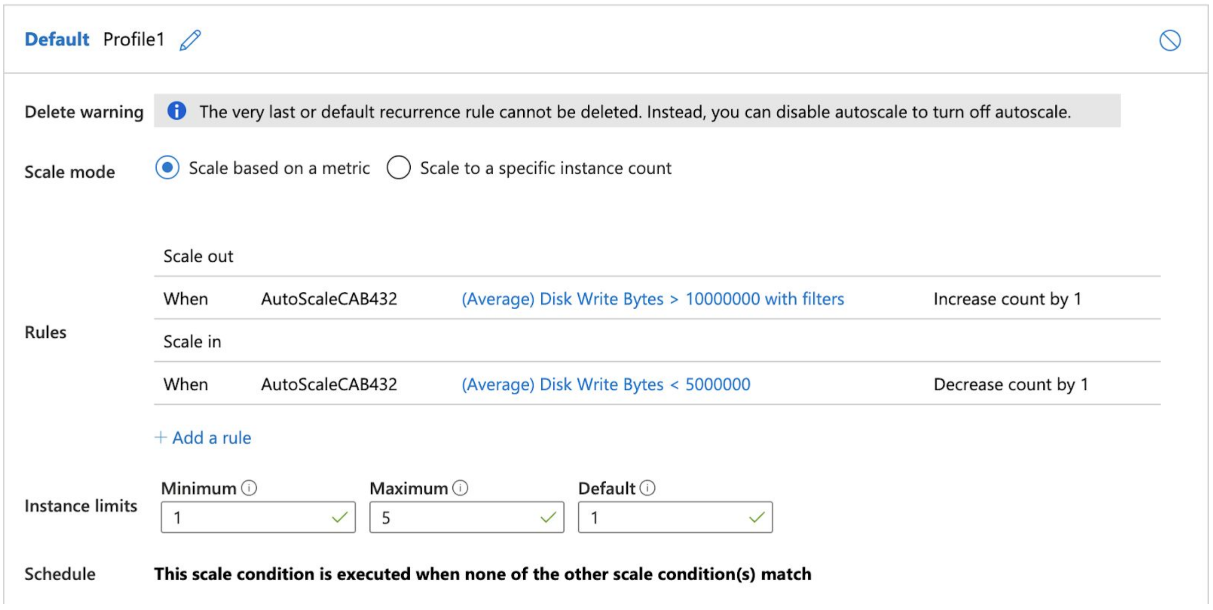
2.5 Virtual Machine Scale Set

Azure virtual machine scale sets (VMSS) let you create and manage a group of identical, load balanced VMs. The number of VM instances can automatically increase or decrease in response to demand or a defined schedule. Scale sets provide high availability to your applications, and allow you to centrally manage, configure, and update a large number of VMs

In the project, an image is captured in one VM initially and then is used to deploy the virtual machine scale set. The initial size of VMSS is 2, the minimum and maximum of the scale set are 1 and 5 respectively.

The auto-scaling metric is set based on average disk write bytes

- Scale in (increase 1 instance): > 10 MB during 1 minute
- Scale out (decrease 1 instance): < 5MB during 1 minute



Default Profile1

Delete warning	The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.
Scale mode	[✓ Scale based on a metric] Scale to a specific instance count

Rules

Scale out			
When	AutoScaleCAB432	(Average) Disk Write Bytes > 10000000 with filters	Increase count by 1
Scale in			
When	AutoScaleCAB432	(Average) Disk Write Bytes < 5000000	Decrease count by 1

+ Add a rule

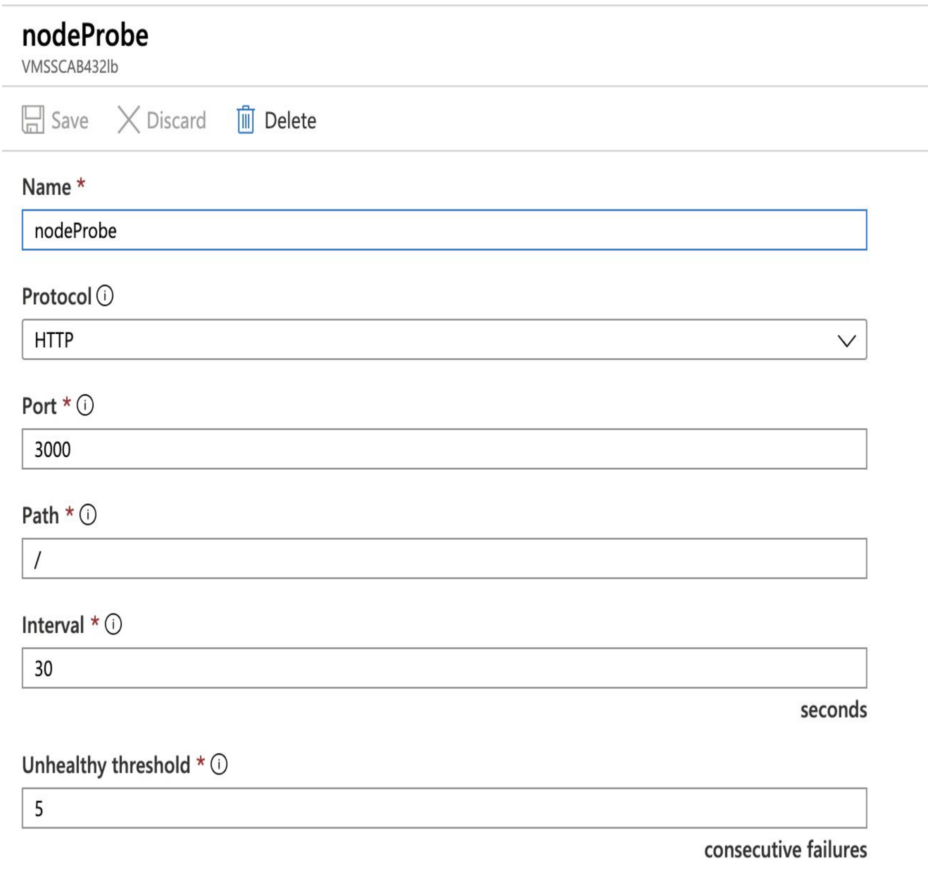
Instance limits

Minimum	Maximum	Default
1	5	1

Schedule **This scale condition is executed when none of the other scale condition(s) match**

Figure 4: Auto-scaling configurations

2.6 Azure Load Balancing Configuration


nodeProbe
VMSSCAB432lb

Name *
nodeProbe

Protocol
HTTP

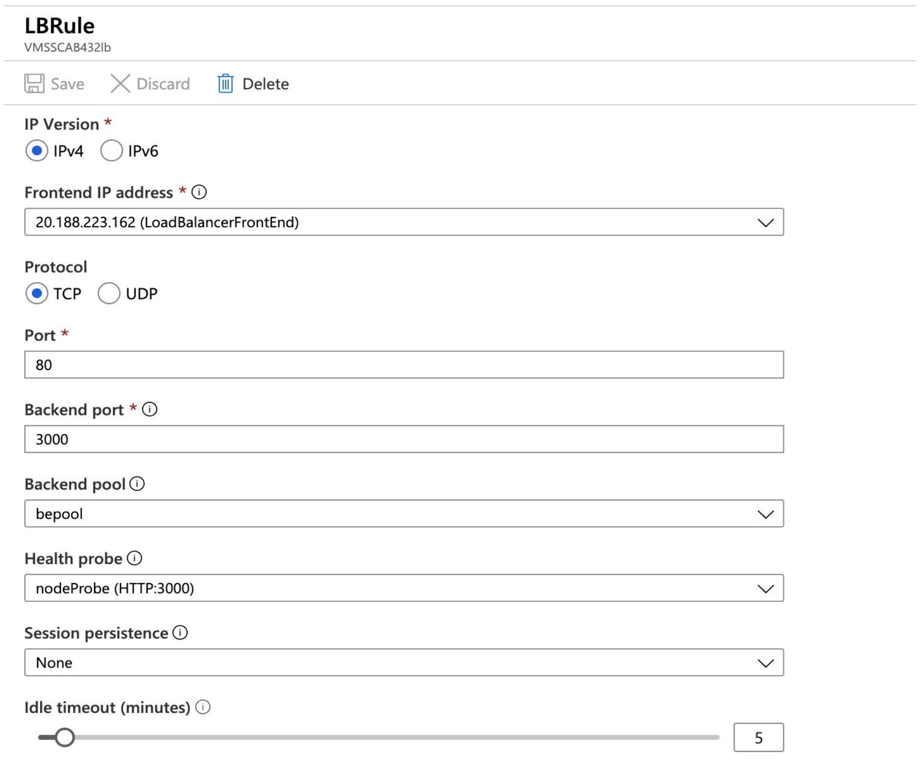
Port *
3000

Path *
/

Interval *
30
seconds

Unhealthy threshold *
5
consecutive failures

Figure 5: Health Probe of Load Balancer


LBRule
VMSSCAB432lb

IP Version *
[✓ IPv4] IPv6

Frontend IP address *
20.188.223.162 (LoadBalancerFrontEnd)

Protocol
[✓ TCP] UDP

Port *
80

Backend port *
3000

Backend pool
bepool

Health probe
nodeProbe (HTTP:3000)

Session persistence
None

Idle timeout (minutes)
5

Figure 6: Load balancing rules of Load Balancer

2.7 Redis Storage

Redis is an open-source, in-memory data structure store, used as a database, cache and message broker. Along with Azure Cosmo DB, Redis was utilised for the client to cache their results in memory for quick access in this project. This allows recent queries to be accessed at much faster speeds.

2.8 Docker

Docker was utilised to encapsulate the application into a container for easy download and access. The figure below depicts the Dockerfile utilised for the containerisation of Tweet Your SentimentThe build is derived from version 10 of node. The source code is then copied into the base directory of the image along with setting the work directory for that source code. The *RUN* command is then performed to install the required Node packages for the Docker image. Finally, port 3000 is exposed and the necessary commands are utilised to run the application.

```
# Install node v10

FROM node:10-alpine

# Set the workdir /var/www/myapp

WORKDIR /var/www/myapp

# Copy the package.json to workdir

COPY app/package.json ./

# Run npm install - install the npm dependencies

RUN npm install

# Copy application source

COPY /app .

# Expose application ports 3000

EXPOSE 3000

# Start the application

CMD [ "npm","start"]
```

Figure 7: Docker file utilised

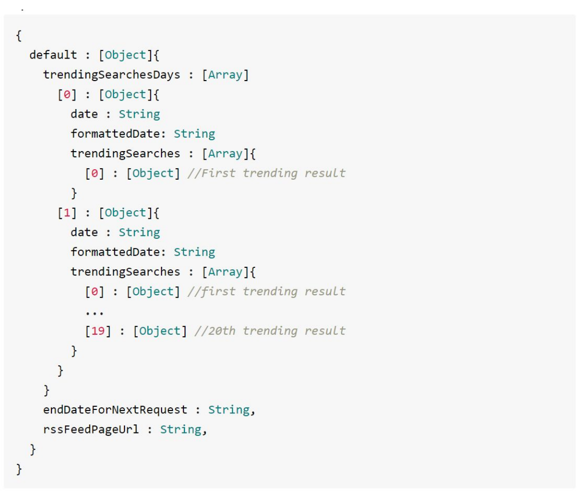
Additionally, Docker Compose is used to run both containers (server and redis) into cloud instance with file docker-compose.yml presented below. The settings are pretty straight-forward except for the restart elements in both containers are set "always" to start on each cloud instance.

```yaml
version: '3'
services:
  myapp:
    container_name: myapp
    restart: always
    build: .
    ports:
      - '3000:3000'
    networks:
      - redis
    depends_on:
      - redis
  redis:
    image: redis:latest
    restart: always
    command: ["redis-server", "--bind", "redis", "--port", "6379"]
    networks:
      - redis
networks:
  redis:
    driver: bridge
```

Figure 8: docker-compose.yml

2.8 Google Trends API

The Google Trends API package in node.js was utilised to obtain the topics in trend in Australia. It is a simple package with an example shown below.



```
{
  default : [Object]{
    trendingSearchesDays : [Array]
      [0] : [Object]{
        date : String
        formattedDate: String
        trendingSearches : [Array]{
          [0] : [Object] //First trending result
        }
      [1] : [Object]{
        date : String
        formattedDate: String
        trendingSearches : [Array]{
          [0] : [Object] //first trending result
          ...
          [19] : [Object] //20th trending result
        }
      }
    }
    endDateForNextRequest : String,
    rssFeedPageUrl : String,
  }
}
```

Figure 9: Example of Google Trends API output

3.0 Use cases

1. As a user of Twitter, I want to be able to view the most common words in tweets about a topic for positive and negative sentiments respectively, so that I can see the common words associated with the respective sentiments.
2. As a user of Twitter, I want to be able to view the sentiment score for tweets of a certain hashtag on twitter, so that I can see the current sentiment users have towards that topic
3. As a user of Twitter, I want to be able to view the current trends being tweeted on Twitter so that I know which topics are hotly debated right now.

4.0 Technical Breakdown

4.1 Architecture and data flow

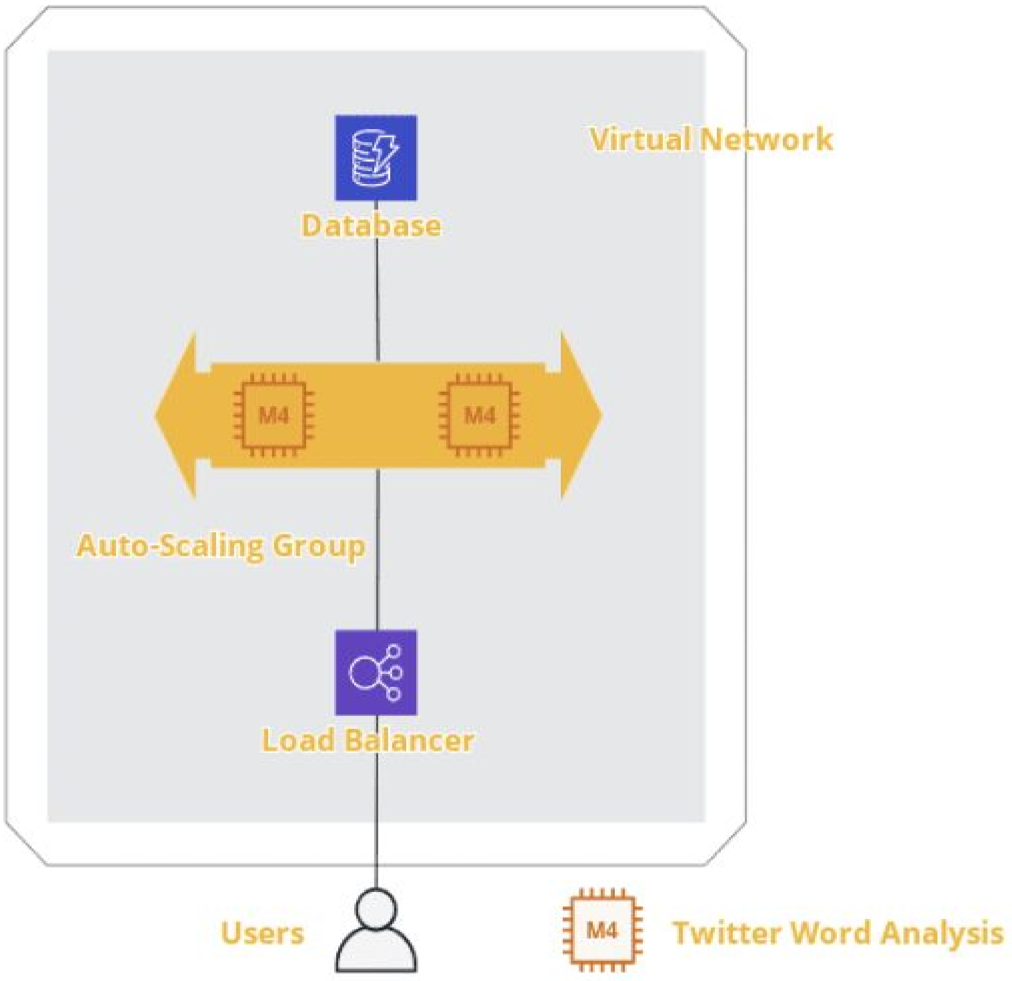


Figure 10: Architecture Diagram

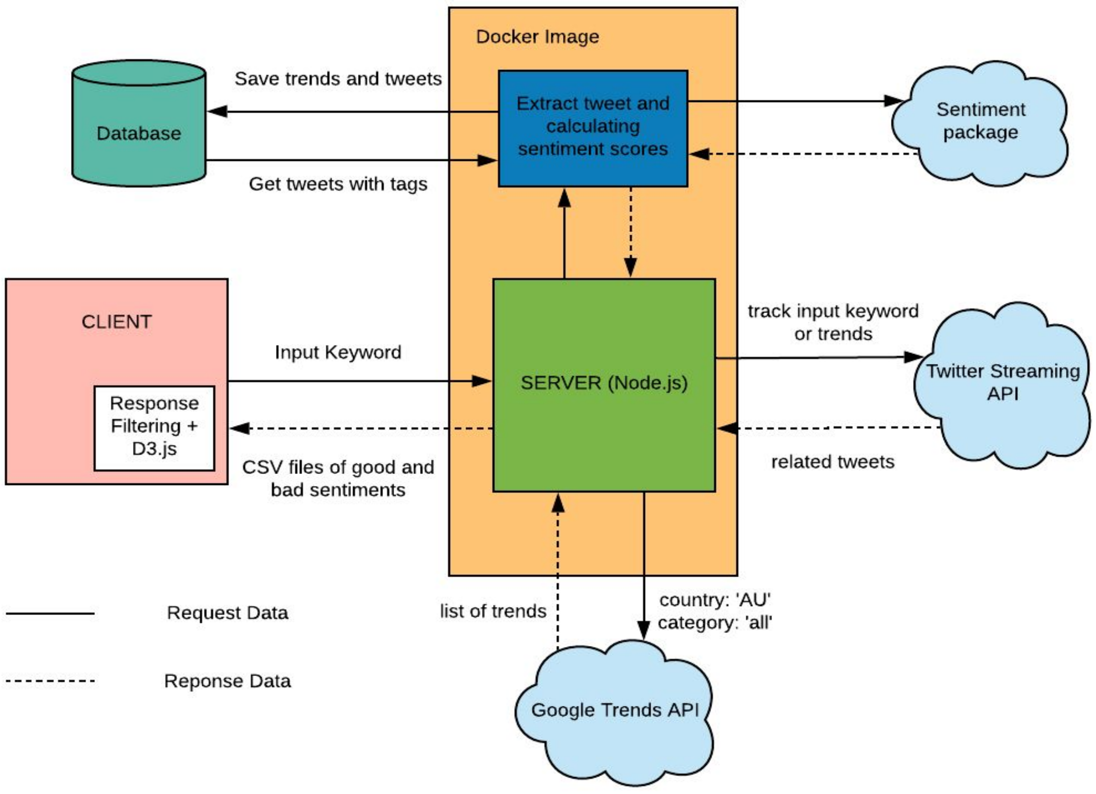


Figure 11: Data Flow Diagram

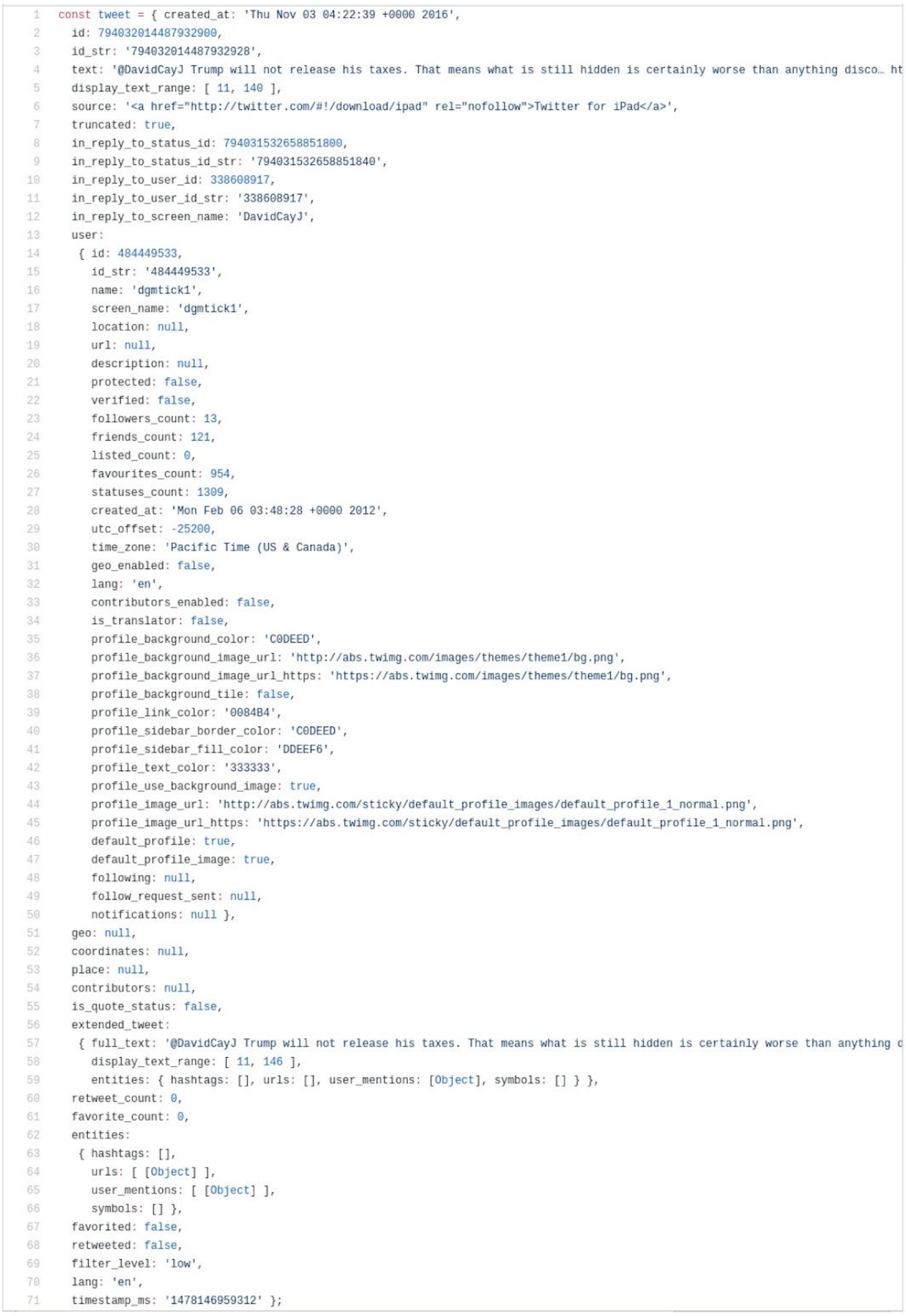
4.2 Client / server responsibilities and Data flow

Initially, upon the boot up of the application the Google Trends API is utilised to get the current trending topics in Australia. These topics are then set as links on the web page for the user to click on if they wish to decide to query that topic. Upon the input of a keyword for Twitter to track or alternatively, a trend is clicked, the server creates an API request with the keywords to the Twitter streaming API where the relevant tweets are retrieved. The server then filters these tweets for their block of text which is then analysed by the Sentiment package. The tweets and their sentiment scores are then stored in the Azure Cosmo database. The tweets and their tags are extracted from the database and sent to their client as an array of tweets of good sentiment and bad sentiment respectively. The server then processes these tweets into two CSV files containing the words and their frequencies. The CSV file is then served to the Client where the D3.js library utilises the CSV files to display the corresponding graphs. In addition to this, the Client is also served the total sentiment score for that topic.

On top of this application was the use of the Azure Load Balancing configuration tools where the application was scaled automatically based on the incoming traffic of client requests. From the architecture diagram 'M4' represents the application shown in the data flow diagram.

5.0 Response filtering / data object correlation

<u>Raw Data:</u>



```
const tweet = { created_at: 'Thu Nov 03 04:22:39 +0000 2016',
  id: 794032014487932900,
  id_str: '794032014487932928',
  text: '@DavidCayJ Trump will not release his taxes. That means what is still hidden is certainly worse than anything disco… ht
  display_text_range: [ 11, 140 ],
  source: '<a href="http://twitter.com/#!/download/ipad" rel="nofollow">Twitter for iPad</a>',
  truncated: true,
  in_reply_to_status_id: 794031532658851800,
  in_reply_to_status_id_str: '794031532658851840',
  in_reply_to_user_id: 338608917,
  in_reply_to_user_id_str: '338608917',
  in_reply_to_screen_name: 'DavidCayJ',
  user:
   { id: 484449533,
     id_str: '484449533',
     name: 'dgmtick1',
     screen_name: 'dgmtick1',
     location: null,
     url: null,
     description: null,
     protected: false,
     verified: false,
     followers_count: 13,
     friends_count: 121,
     listed_count: 0,
     favourites_count: 954,
     statuses_count: 1309,
     created_at: 'Mon Feb 06 03:48:28 +0000 2012',
     utc_offset: -25200,
     time_zone: 'Pacific Time (US & Canada)',
     geo_enabled: false,
     lang: 'en',
     contributors_enabled: false,
     is_translator: false,
     profile_background_color: 'C0DEED',
     profile_background_image_url: 'http://abs.twimg.com/images/themes/theme1/bg.png',
     profile_background_image_url_https: 'https://abs.twimg.com/images/themes/theme1/bg.png',
     profile_background_tile: false,
     profile_link_color: '0084B4',
     profile_sidebar_border_color: 'C0DEED',
     profile_sidebar_fill_color: 'DDEEF6',
     profile_text_color: '333333',
     profile_use_background_image: true,
     profile_image_url: 'http://abs.twimg.com/sticky/default_profile_images/default_profile_1_normal.png',
     profile_image_url_https: 'https://abs.twimg.com/sticky/default_profile_images/default_profile_1_normal.png',
     default_profile: true,
     default_profile_image: true,
     following: null,
     follow_request_sent: null,
     notifications: null },
  geo: null,
  coordinates: null,
  place: null,
  contributors: null,
  is_quote_status: false,
  extended_tweet:
   { full_text: '@DavidCayJ Trump will not release his taxes. That means what is still hidden is certainly worse than anything d
     display_text_range: [ 11, 146 ],
     entities: { hashtags: [], urls: [], user_mentions: [Object], symbols: [] } },
  retweet_count: 0,
  favorite_count: 0,
  entities:
   { hashtags: [],
     urls: [ [Object] ],
     user_mentions: [ [Object] ],
     symbols: [] },
  favorited: false,
  retweeted: false,
  filter_level: 'low',
  lang: 'en',
  timestamp_ms: '1478146959312' };
```

Figure 12: Example of raw tweet from Twitter

Filtered Data:

```
{
   "id": "cab432-tweets-1187255156011913217",
   "tags": "Independent-Broad-based-Anti-corruption-Commission",
   "text": "lizcourserants It also agues that we should base our morality on law because it made it into law so its based on expert opinion  the law in particular they cite is no longer valid due to being ruled unconstitutional httpstcoqWPbGNpbNQ",
   "score": -1
}
```

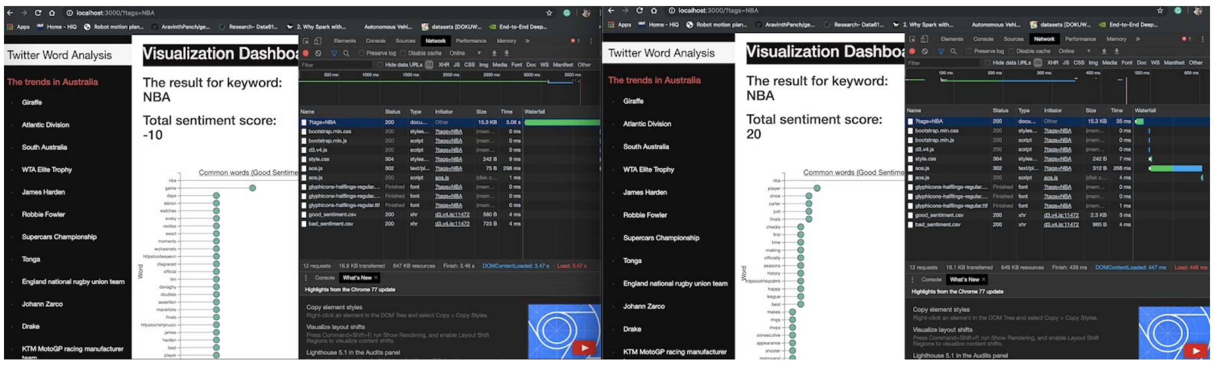
6.0 Test plan

Tasks	Expected Result	Result
Display Basic D3.js lollipop bar chart	Basic lollipop chart displayed on web page view	PASS
Test Twitter API	Display and parse tweets to display onto the console to view.	PASS
Sentiment Analysis on Tweets	Perform Sentiment Analysis on each of tweets and display these results on the console	PASS
Get all the trend topics on twitter	Obtain all the topics in trend on Twitter in Australia and display to the console	PASS
Uploading the filtered tweets with their sentiment score onto Cosmo DB	Data successfully uploaded and viewed on the Azure portal	PASS
Extracting the filtered tweets with their sentiment score from Cosmo DB	Data successfully downloaded the data from Azure portal with correct queries	PASS

Counting the number of words in a list of text strings	Function written to count the number of words via inputs of blocks of strings.	PASS
Create CSV file for D3.js	Function written to create a CSV file for the word and frequencies	PASS
Display custom CSVs for positive and negative sentiment	Displaying both D3.js lollipop charts based on custom CSVs on view	PASS
Integrating Redis	Redis integrated for local caching	PASS
Integrate data extracted from Cosmo DB and d3.js visualisations	Searching a tweet or trend topic will obtain data, pass it through sentiment analysis and display the D3.js charts	PASS
Make sure you can return to home page	Clicking on title should return you to home view	PASS
Dockerize Application	Write a docker file for application and ensure it works	PASS
Docker Compose	Write docker compose for app and Redis containers	PASS
Deploy with Load Balancing	Application should scale up based on the incoming traffic	PASS

Test Persistence

The figure below proves the persistence capability of our website.



The result for keyword: NBA

Total sentiment score: -10

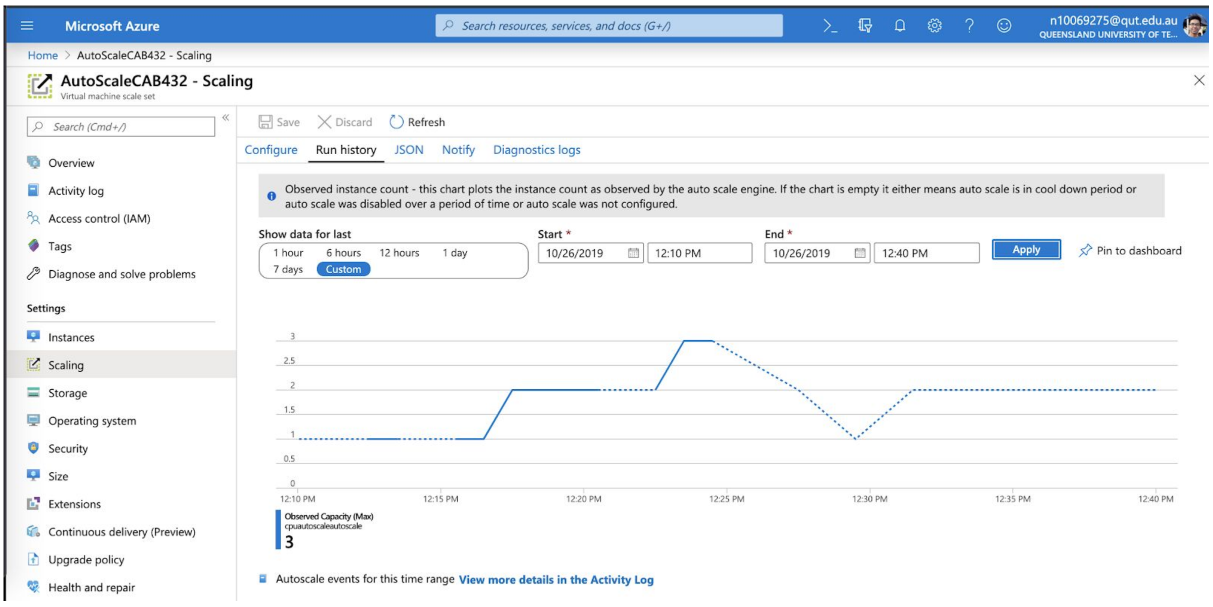
The result for keyword: NBA

Total sentiment score: 20

Figure 13: (Left image) Initial load, (Right image): Refresh the page

Test Scaling

Postman is used for testing the auto-scaling capability (12 requests with 1000 iterations and 30ms delay)


Home > AutoScaleCAB432 - Scaling

AutoScaleCAB432 - Scaling

Virtual machine scale set

Observed instance count - this chart plots the instance count as observed by the auto scale engine. If the chart is empty it either means auto scale is in cool down period or auto scale was disabled over a period of time or auto scale was not configured.

Show data for last: Custom

Start *: 10/26/2019 12:10 PM

End *: 10/26/2019 12:40 PM

Observed Capacity (Max) cpuautoscaleautoscale: 3

Autoscale events for this time range View more details in the Activity Log

Figure 14: The history of auto-scaling instances

8.0 Difficulties / Exclusions / unresolved & persistent errors /

- Original Brief not achieved:
    - The original idea for web application was a similar application that utilised data from Reddit instead. A user was envisioned to search a subreddit where a stream of subreddit comments will come in and hence, allow for the sentiment analysis of these comments.
    - Hot posts would also be shown along with "golded" reddit comments
    - This was not feasible as "Snoowrap", the reddit wrapper package did not allow for the turning off of previous streams as a new query is performed.
    - Thus, the Twitter streaming API was utilised instead
- Display of tweets (Periodic)
    - A desired function was to periodically display update 5 tweets shown on the view for users to just be able to see examples of some of the tweets being analysed.
- Twitter stream doesn't update (bug)
    - Sometimes when a user queries another tracking tag instead, the Twitter Stream API returns the same tweets as before even though the tracking tag has successfully been updated. This right now cannot be fixed.
- Display the data on visualization dashboard
    - The problem of separating the script of D3.js in views folder is not solved now. Our group has an issue of loading the data when moving D3.js code to public folder.

- To get enough time for retrieving the data from database, the request of getting data is waited about 3000 milliseconds for some filtered tweets going into the database.
- Some less common keywords with small amounts of tweets do not allow us to see the results immediately. This has to be refreshed twice or more times to get the correct results.

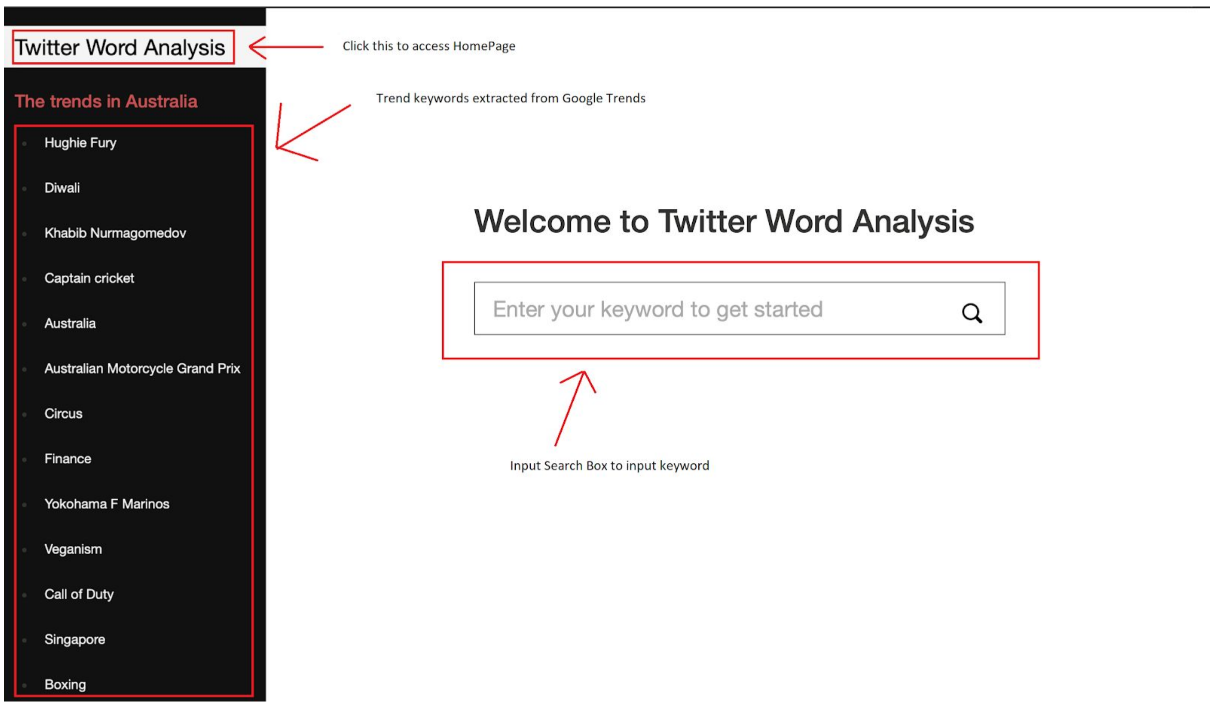
9.0 Extensions

- Letting the users see a live feed (display) of the tweets of the selected topic on the front-end. Maybe by choosing tweets at random to display.
- Classify extracted tweets into different labels such as politics, sports, science or health.
- A live data visualisation of the current sentiment of a topic (maybe through the use of sockets)

10.0 User guide

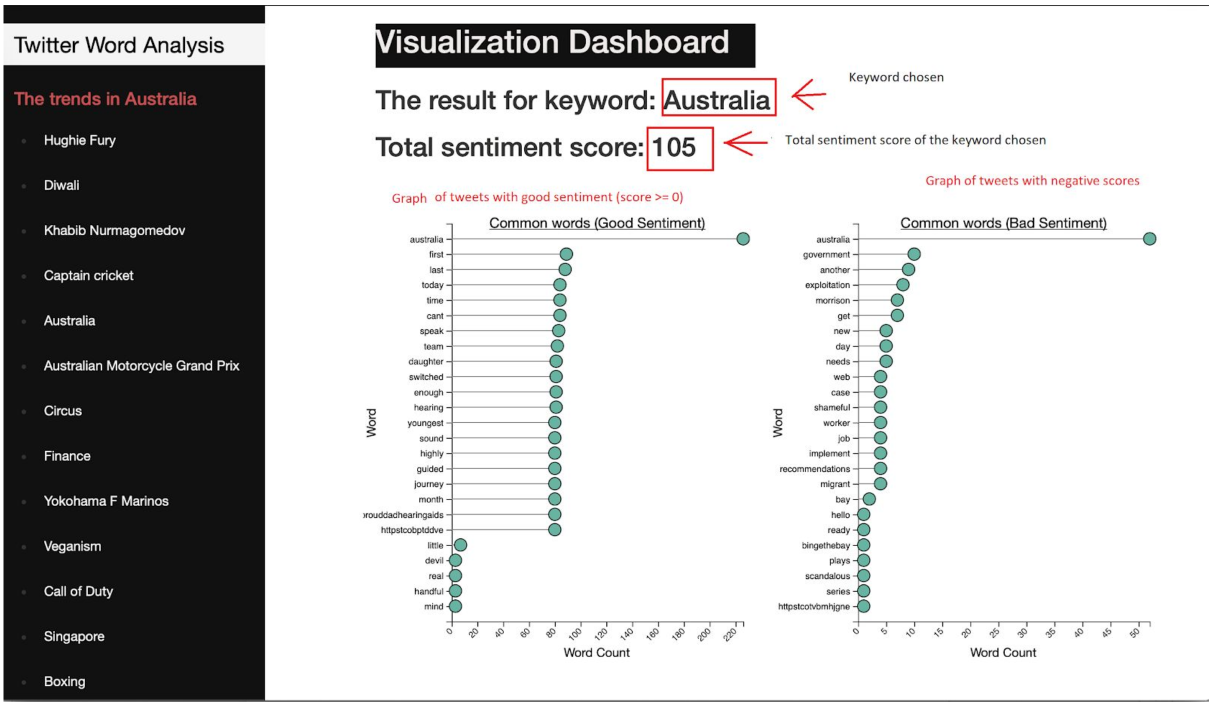
Homepage Website

- User can type a keyword in the search input box or click one of trend keywords on the left side to see the results


Twitter Word Analysis
Click this to access HomePage
The trends in Australia
Trend keywords extracted from Google Trends
Hughie Fury
Diwali
Khabib Nurmagomedov
Captain cricket
Australia
Australian Motorcycle Grand Prix
Circus
Finance
Yokohama F Marinos
Veganism
Call of Duty
Singapore
Boxing
Welcome to Twitter Word Analysis
Enter your keyword to get started
Input Search Box to input keyword

Visualization Dashboard with keyword "Australia"

- Display the results based on the keyword provided. The results include the total sentiment scores of all related tweets and two graphs counting the most common words of tweets with positive & neutral and negative scores.
- Reload the webpage to get updated result



Twitter Word Analysis

The trends in Australia

- Hughie Fury
- Diwali
- Khabib Nurmagomedov
- Captain cricket
- Australia
- Australian Motorcycle Grand Prix
- Circus
- Finance
- Yokohama F Marinos
- Veganism
- Call of Duty
- Singapore
- Boxing

Visualization Dashboard

The result for keyword: Australia

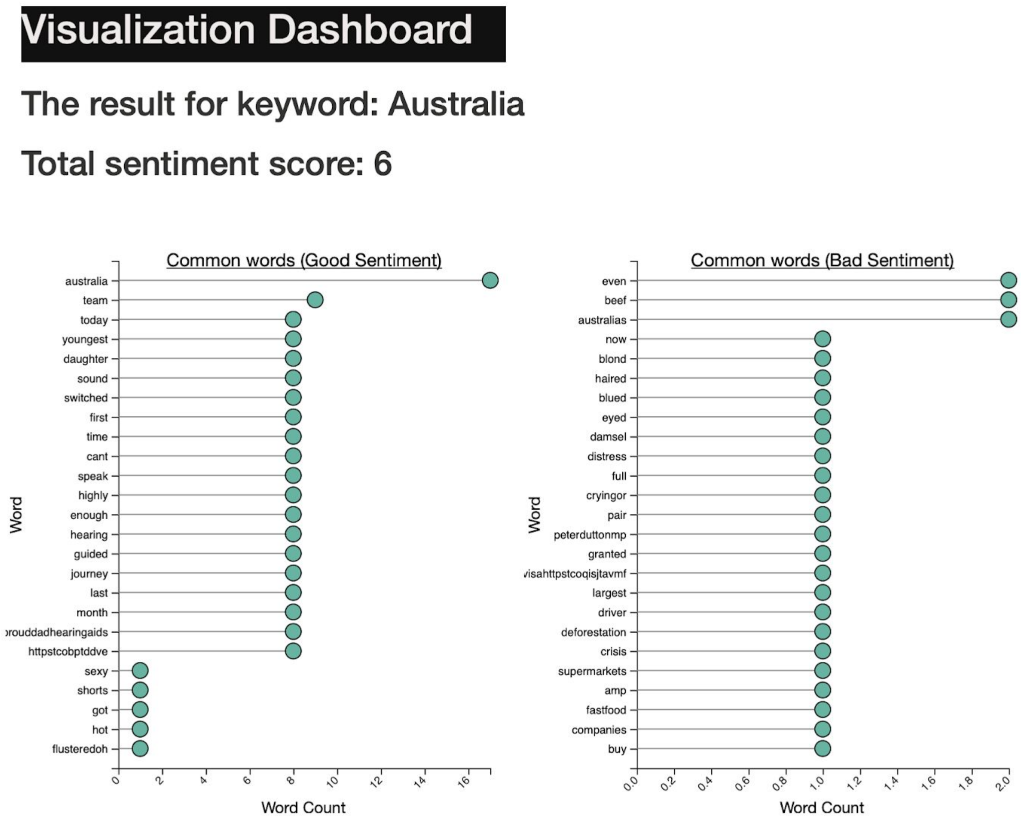
Total sentiment score: 105

Keyword chosen

Total sentiment score of the keyword chosen

Graph of tweets with good sentiment (score >= 0)

Graph of tweets with negative scores



Visualization Dashboard

The result for keyword: Australia

Total sentiment score: 6

11.0 References

[1] Node.js with Azure Cosmos DB SQL API

https://docs.microsoft.com/en-us/azure/cosmos-db/sql-api-nodejs-get-started

[2] Setup Docker Compose for Node.js Express and Redis

https://medium.com/@avishwakarma/getting-started-with-docker-for-nodejs-mongodb-and-redis-b97188d33559

[3] Twitter Streaming API

https://developer.twitter.com/en/docs/basics/getting-started

[4] Node Sentiment

https://www.npmjs.com/package/sentiment

[5] Google Trends API

https://www.npmjs.com/package/google-trends-api

[6] D3.js visualisation library

https://d3js.org

[7] Azure Load Balancer

https://azure.microsoft.com/en-us/services/load-balancer/

[8] Azure Virtual Machine Scale Set

https://azure.microsoft.com/en-us/services/virtual-machine-scale-sets/