Assignment 2 Report

CAB432 Cloud Computing – semester 2 2019

John Santias N9983244

Ka long Lee n9845097

2019

# Introduction

Tweetery aims to provide users with a visual representation of the emotional value of tweets on a search or trending topic. Utilising the Twitter API for getting tweets into the web server, cleaning the tweet body and performing emotional analysis with IBM Watson API. The average emotion scores of the chosen topic is displayed on a graph rendered on the client side.

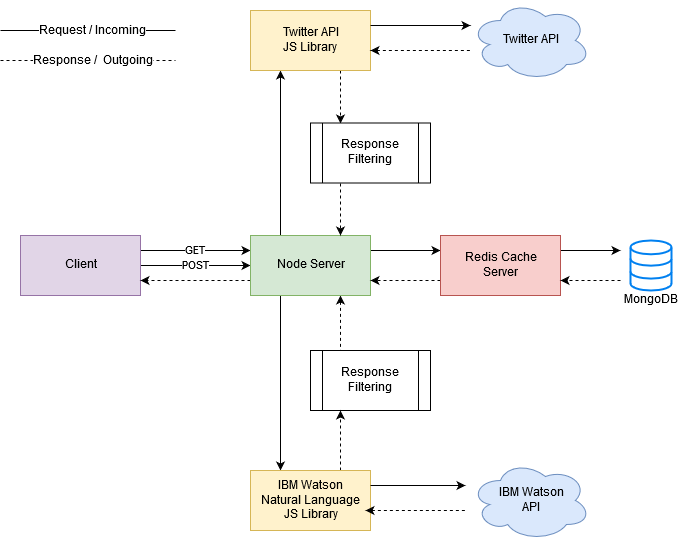
When the user searches for a topic, the query is sent to the node server for further processing. The server uses the search query to get tweets from Twitter API. The received tweets are then combined and sent to IBM’s Watson Natural Language Understanding API to analyse the emotions. These results are stored on a remote global Redis Cache server which can be returned to users who searches for the same topic. Results are also stored on MongoDB for long term storage and can be copied to the cache upon user request. The node server will repeat this processing for each of the last seven days.

Between the client and node server, and the node server and Redis cache server, is a load balancer to assist in handling high volume of traffic that can be generated by popular trending topics and queries. The load balancer assists in the scaling of our servers.

# Technical Description

## Cloud Infrastructure

## Architecture Diagram



# Client Side

The client can initiate two types of request, a GET and a POST request. When a user requests to access the application, a GET request is sent to the server. The server returns the index page with a nav bar, a list of trending topics and a search bar. When the user initiates a search, the form body sends a POST request to the server for processing.

The server handles the query sent from the form, processes the API calls with the query and displays the results in a graph.

# Server Side

The node server manages the incoming client request and outgoing responses. When it receives a GET request, the server renders and returns the home page with a nav bar, search bar and trending topics. There is no heavy processing in this request because the client hasn’t searched for anything. The client is only requesting the home page.

When the server receives a POST request, it calls the Twitter API endpoint with the body query obtained from the search form. This API endpoint returns a JSON response with an array of objects. Each object containing the tweet message, user information etc. Tweet messages are only extracted, combined and passed to IBM’s API for analysing the emotions. This information is stored in the cache named with a key and mongoDB by date and key. This information is also rendered and sent to the client for display. When the user initiates the same request and query, the server will check if the data is in the cache or db. If so, that data is returned to the client and prevents calls to the APIs.

# Development

Tweetery will be developed in stages in order to have tight grip n the scope of the application and reduce the number of issues towards the end of development.

## Stage 1:

Stage 1 will involve setting up the skeleton of the project. Creating two folders one dedicated for the client side, and the other dedicated for the server-side. API keys will also need to be created. Dependencies will need to be installed for the API to return data. By the end of this stage, it is expected that the application can call the Twitter API based on a search query, analysed for emotion data and returned to the client.

## Stage 2:

Stage 2 will involve settings up the cache for short-term storage and database for long-term storage. Ensuring that the analysed results are stored on the database and cache for future requests.

## Stage 3:

Stage 3 will involve deploying to the cloud, testing and bug fixing. Manual scaling.

## Stage 4:

Stage will involve monitoring the application hosted on the cloud to find the most optimal way for auto-scaling.

# API and Packages

## Twitter API

<https://developer.twitter.com/en/docs.html>

Twitter is a social network platform for users to share messages, images and videos on a feed. Messages can be categorised by hashtags (#) and searches can be conducted on those hashtags or keywords. Tweetery uses this API to retrieve tweets based on a hashtag or keyword.

## IBM Watson Natural Language Understanding API

<https://www.ibm.com/watson/service/natural-language-understanding>

The IBM Watson Natural Language Understanding API is one of IBM’s Watsons machine learning API. It is used to process advanced test analysis and extract metadata from content such as concepts, entities, keywords, categories, sentiment, emotions, relations, and semantic roles. Some of which can provide a score for the tone or a list of words that have great meaning for the user.

Tweetery uses this API to analyse the emotion of the tweets gathered and return a score between zero and one. Given the highest score is the emotion analysed in the content.

## MongoDB

<https://www.mongodb.com> & <https://mlab.com>

MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program and uses JSON-like documents with schema. mLab was used to host MongoDB databases on a cloud provider (AWS, GCP, Azure). Tweetery uses one table to store emotion analysis.

## ChartJS

<https://www.chartjs.org>

ChartJS is a community-maintained project for developers to visualise data in eight different ways. Each of them animated and customisable. Tweetery uses this library to visualise the emotional scores of the given topic.

# Use Cases

*As a user, I want to see a list of trending topics and select one to see the average emotional stats.*

The user navigates to the home page of tweetery. User selects a trending topic from the right-hand side and sees the results.

*As a user, I want to select or search multiple trending topics to compare the emotions of those topic.*

The user selects/searching multiple topics and sees the results on the same page.

