COMP90024: Cluster and Cloud Computing, Sem 1, 2018
Assignment 2: Australian Social Media Analytics
Team - 47

Prarthana Padia - ppadia@student.unimelb.edu.au
Pallabi Bhattacharya - pbhattachar@student.unimelb.edu.au
Bhavya Singhal - b.singhal1@student.unimelb.edu.au
Khanh Tan Nguyen - k.nguyen206@student.unimelb.edu.au
Shajid Mohammad - shajidm@student.unimelb.edu.au

1 Introduction

The internet and mobile technologies has led to the rise of social media providing platforms for dissemination of information along with generation of a variety of content. Research in social media has greatly intensified in the past few years due to the numerous research opportunities provided by the varied content. With large amount of data available, it is essential to develop the solution for data analysis with architecture which can support big data.

The focus of this assignment has been to harvest tweets from across the cities of Australia on the Nectar Research Cloud and undertake variety of social media analytics scenarios that tell interesting stories of life in Australian cities. The key technology available to researchers today is the cloud computing architecture. Multiple Virtual Machines (VMs) can be utilized to harvest, store, and analyse twitter data. The cloud services utilized in this assignment is The National eResearch Collaboration Tools and Resources research cloud (NeCTAR) which operates based on the openstack protocol. Data for the analysis topics are also provided as part of Australian Urban Intelligence Network (AURIN) for correlating with the Twitter data.

The project can be divided into four components :

1. Creating the infrastructure on NECTAR
2. Building and deploying the twitter harvesters
3. The web application and visualization of the analysis
4. Automation through boto/ansible script.

2 NeCTAR Research Cloud

NeCTAR research cloud is based on OpenStack which is an open source cloud technology. It provides us with a computing infrastructure, software and services to store, access, and analyse data, remotely, rapidly and autonomously. In this section we discuss the advantages as well as the issues and challenges faced.

2.1 Advantages

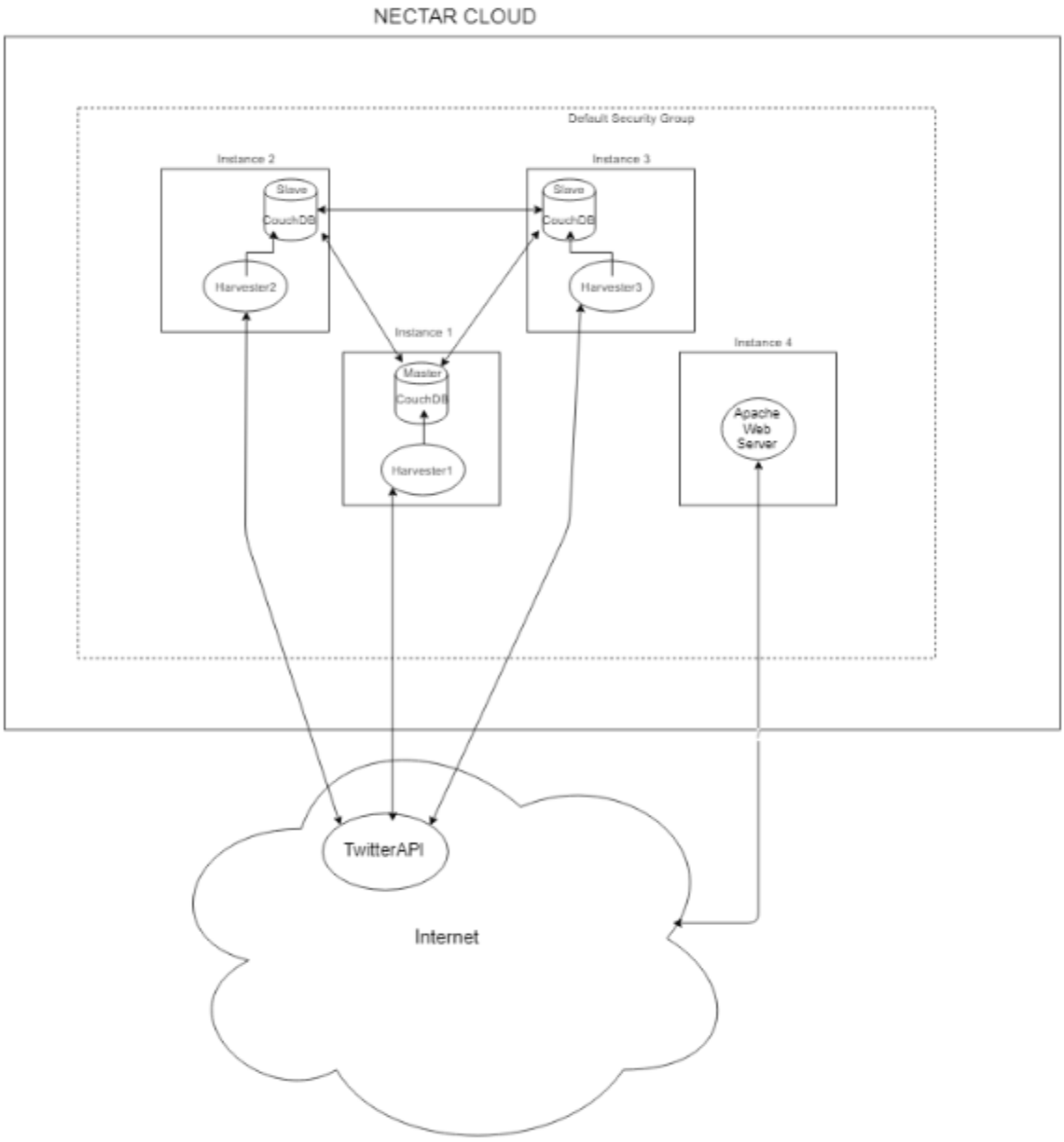
1. NeCTAR has large storage with instant availability and scalability.
2. The deployment process is streamlined by booting up the VMs from many standard available images.
3. Availability zone is easily defined.
4. It allows several users to access one VM and work on it simultaneously.
5. Users can create their own infrastructure with multiple VMs, with the resources available.
6. Deployment can be automated using multiple platforms like Boto, Ansible etc.
7. Since it's the cloud, the storage is scalable as well as elastic.

2.2 Issue and Challenges Faced

1. While provisioning the infrastructure for the project, the host where the VM was provisioning turned out to be a faulty host. Every time a volume was attached to the VM, it just provisioned for a while and then didn't attach. Nectar Support pointed out it was a faulty host, and took it out of production.
2. Whenever ansible scripts are used deploy softwares in VM, roughly half of the time the host machines faces SSH issue to the VM it wants to connect to. This issue occurs using same or different keys on a random basis. After repeating the same steps multiple times, sometimes it work, sometimes it doesn't which renders it difficult to figure out what went wrong.
3. The security groups have to defined and rules have to managed carefully so as to not allow mishap to the infrastructure.

3 Architecture

The designed architecture utilizes four virtual machines (VMs) in the assigned group project in NeCTAR cloud. One VM operates as a Web server for the front end which is used to do the analysis on the data. It is designed so that the result of the analysis is stored in the same place as the web server for the front end. This virtual machine will have CouchDB installed for storing the data. The other three instances are used to deploy the twitter harvesters and store the tweets as json document in the CouchDB.

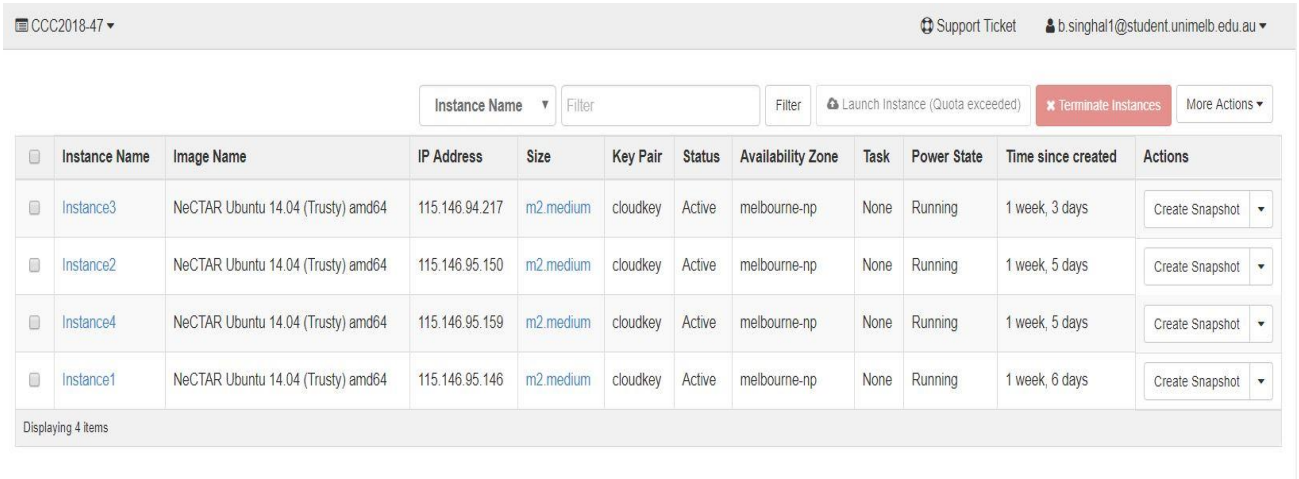


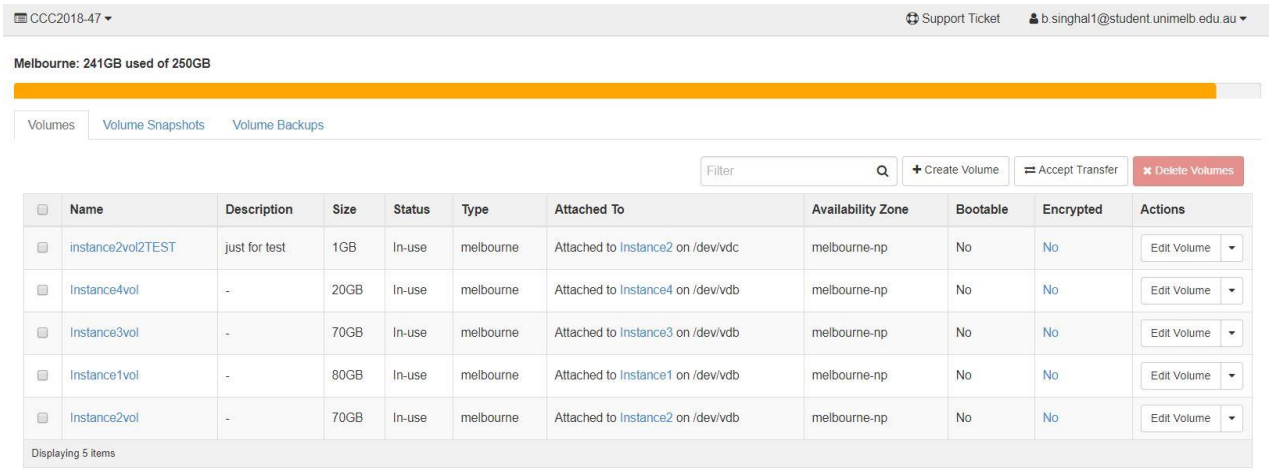
Within the CouchDB, the views are created for the analysis and stored as
separate document in the same database where the harvested tweets are stored. The data imported from AURIN is used in analysis for correlating with the twitter data. The JSON files in the aurin dataset are stored in CouchDB as a document in a separate database. However the shapefile in the aurin dataset is accessed directly from the rest services for the analysis.

3.1 Features

1. The architecture is scalable if the research requires large amount of research data, the VMs for harvesters can be replicated as much as the resources from NeCTAR allows the researchers to.
2. External storage is also attached on the Harvester as well as the replicated analysis VM to expand the capabilities of the system. Multiple harvesters can be configured to store the tweets harvested within its own local virtual machine.
3. Then multiple VMs can use similar MapReduce function to create values of interest.
4. These values can then be aggregated using analytic tool such as Python as long as the tools have library to interface with the CouchDB.
5. The architecture doesn't have a single point of failure in terms of both harvesters as well as the Database. CouchDB is deployed as a cluster, and multiple harvesters are deployed for collection of data.

3.2 Configuration Details of Infrastructure





4 Twitter Harvester

Twitter provides two APIs for fetching tweets, Search API and Streaming API. The Search API allows the user to both read and write tweets, and we can use search query to gather tweets based on keywords which we provide. We have used Search API since it goes back in time to collect tweets on a subject. It fetches whatever tweets has been posted based on the search keyword in the past 7 days. This API has a fairly rich set of operators that can filter results based on attributes like location of sender, language, and various popularity measurements which helps in the sentiment analysis.

4.1 Working of Harvester, Error and Redundancy Handling

1. To access the APIs, we first create twitter application after logging in to twitter. After getting the consumer key and consumer secret from the application, we generate the access tokens.
2. We have created python programs which act as the tweet harvesters. The harvesters access the APIs and gather tweets using the Twython library.
3. The harvesters are setup in the same instances with CouchDB installed to persistently fetch tweets and store them in the database.
4. The tweets are fetched as json objects and is stored in the database as documents by the unique tweet id as the document id.
5. In order to avoid redundancy, there is a check based on tweet id for duplicate tweets before storing it in the database.
6. Exception is handled for the twitter search api rate time limit (180 requests per 15 minutes) and the program goes to sleep for the calculated remaining time in case of overhead.

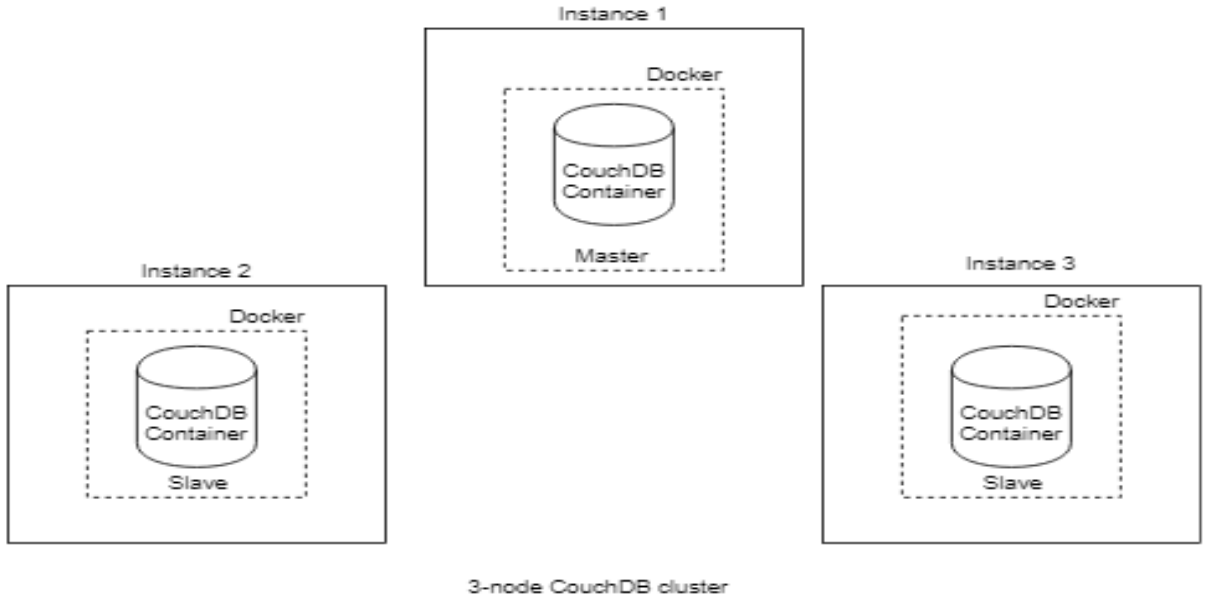
4.2 Limitation

1. We are using one keyword in our search query and can make up to 180 search requests per 15 minutes. That works out as 180 keyword being searched per 15 minutes or 720 keywords searched per hour.
2. Due to this limitation, we deployed three harvesters in three instances to collect large amount of tweets and populate our database.

5 CouchDB Cluster

CouchDB is a open source document based database for storing large amount of data. It is a NoSQL database where the data is stored as documents. We are using CouchDB version 2.1.1

which comes with improved performance as it no longer decompresses the documents just to determine their uncompressed size. Also it is now possible to modify shard maps for system databases. We are using the MapReduce function in CouchDB to analyse our Twitter data for sentiment analysis and aggregate the Aurin data and Twitter data used in scenario analysis.



5.1 Implementation

The data is stored in the following databases -

1. sentiment_tweets - Contains the tweets fetched by the harvesters for sentiment analysis.
2. sc_tweets - Contains the tweets fetched from Twitter for analysing our scenario.
3. divorce_rate_2016 - Contains the json files imported from AURIN

The database is setup as a 3-node cluster with Instance 1 (115.146.95.146) as the master node and instance 2 (115.146.95.150) and instance 3 (115.146.94.217) are the respective slaves. CouchDB is setup as containers using Docker. First Docker was installed in the instances followed by the CouchDB images.

Any modification in the database in one instance is replicated over all the three instances. The three node cluster architecture of the database makes it fault tolerant.

6 Scenario Building for Correlation from AURIN data

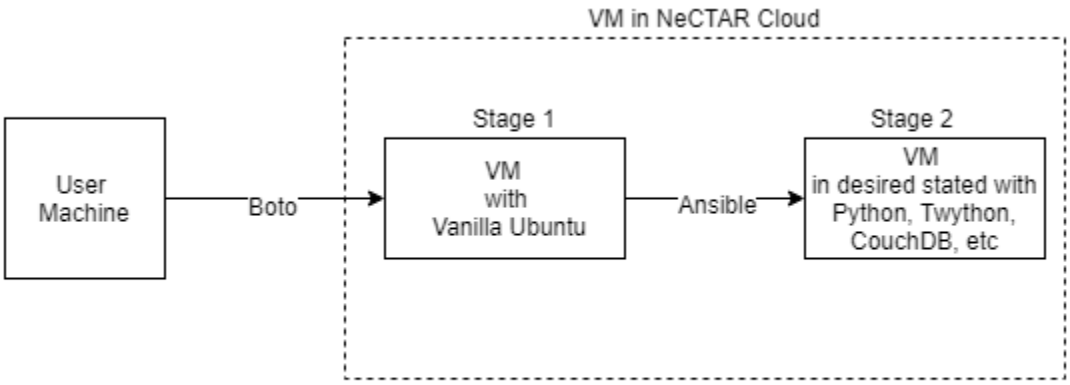
With aforementioned capabilities of harvesters and CouchDB, we are exploring various scenarios using AURIN data and tweets fetched using specific scenario keywords. We deployed three other harvesters to fetch tweets based on our scenario keywords.
The keywords are based on frequently used words and hashtags by Twitter users that trend in urban popular culture, with divorce rates in Melbourne area. We have created a text file which contains these words that are frequently used by twitter users with reference to the current party culture and fetching tweets from Twitter based on these keywords.
These include keywords like #netflixandchill #drinking #clubbing #nightout #happyhour #pubcrawl etc.
We hoped to correlate this scenario to dataset collected from AURIN namely "number of divorces" in 2016 in the city of Melbourne for males and females aged between 15-54 years.

6.1 Implementation

1. For the extra scenario analysis, we are correlating the tweets containing party culture references with AURIN datasets related to divorce rate.
2. Initially, three harvesters specific to three different geolocations were deployed to fetch keywords specified in our scenario-keywords text file and store it in a separate database named sc_tweets.
3. Another harvester code was written which accessed the sc_tweets database to fetch each document in the database, get the username, fetch all the tweets posted by the user and store the tweets in the database which contain our scenario keywords in the text field.
4. The harvester was able to fetch more than 27 thousand tweets, however, since there was no coordinates available for the tweets, we switched to tweets provided by Richard to get tweets with geo location.

7 Automation

Boto and Ansible scripts to deploy a scalable infrastructure on Nectar cloud based on an SSH key.



7.1 Boto

1. Boto is a Python package which is used for creating and provisioning the infrastructure on NeCTAR cloud which includes creation of VMs inside a security group.
2. The VM is provisioned inside an availability zone provided by NeCTAR and is accessed using ssh keys.
3. External persistent storage can be attached to the VMs in the form of volumes.
4. The rules in the security group can be managed based on the security and access requirements.

7.2 Ansible

1. Ansible is an automation tool for configuring and managing computers.
2. Ansible scripts are called playbooks which are written as simple YAML files.
3. In this project it is being used to automate the installation of dependencies inside our virtual machines namely -
    a. CouchDB
    b. Docker
    c. Docker Compose
    d. Python3
    e. Twython
    f. Tweepy
    g. VaderSentiment
    h. Apache 2

7.3 Ready-to-Run Environment

After installing the required softwares, the ansible script copies all the harvester code and the related files to the newly created VM thus creating a ready to run environment.

The ansible playbooks scripts which are called via the shell file "deployboto.sh" then provision various software on the VMs.

Steps to start deployment:

1. Go to Scripts folder.
2. Run the following shell file with command:
   sh deployboto.sh

8 Sentiment Analysis - Implementation

Sentiment analysis has been applied to in the section of Melbourne to correlate the divorce datasets (imported from AURIN) with the twitter data. When gathering the tweets, the python harvester utilizes a library called vaderSentiment to do the sentiment analysis and store the sentiment scores as a new field in the tweet json object to CouchDB (Hutto & Gilbert, 2014). The built-in Map function provided by couchDB is used to filter tweets which don't have coordinates. A view is created in the database which stores coordinates and their respective tweet sentiment scores as the key-value pair. Compound score is then used to measure the actual sentiment of the tweet.
Based on the analysis, we are visualizing the results in the form of graphs in one of our virtual machines (Instance 4).

8.1 Vader Sentiment Analysis

"VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media" (Hutto, & Gilbert, 2014). It deals with many complex situations when mining twitter content. Instead of simply searching terms, it solve situations like typical negations, emoticons, slang words etc.
After applying vaderSentiment, the analysis produces a sentiment score in the following format

```
{
   "neu": 0.85,
   "neg": 0,
   "pos": 0.15,
   "compound": 0.3182
 }
```

The pos, neu, neg scores are ratios for proportions of text that fall in each category, which means they should add up to 1 or close to 1 with float operation. The compound is computed using some rules and normalized to be between -1 (extreme negative) and +1 (extreme positive).

The compound score is used to classify whether a tweet is positive, neutral, or negative.
1. Positive sentiment: compound score > 0
2. Negative sentiment: compound score <= 0
The script stores all scores, neu, neg, pos, compound to a new field called "sentiment"in the tweet json and stores it in the database.

8.2 Limitations of VADER Sentiment Analysis

1. The data available on social media are not structured since the users tend to use a lot of out of vocabulary words and special characters.
2. The character limitation in Twitter often makes the user use abbreviations.
3. Moreover, twitter is also mainly used as a platform for advertisement include a large number of URLs, usertags which makes the analysis less accurate.
4. VaderSentiment also doesn't process sarcasm. Vader does classify emojis up to a certain extent, but not all, since vader is still improving and some of the image based emojis, memes are not always supported.
5. A large number of tweets give a neutral sentiment when the tweet consists of a few characters to describe the image inside the tweets.

9 Map-Reduce in CouchDB

The analysis makes use of the MapReduce provided by CouchDB. A map function is stateless, it takes a value or a parameter, and gives <key, value> pairs as the output.
The first step (Map), distributes data across machines, while the second (Reduce) hierarchically summarizes them until the result is obtained. It allows users to run the function against values in parallel.
In this case, the parameter is a tweet for one call. Map function is used to create a view which label the region in which the tweet belongs to. Then reduce function creates an average compound sentiment score for the region. Map function for this task is as follows:

9.1 Map:

The used Map function checks whether a tweet has coordinates, by checking for respective values at the key "doc.place.bounding_box.coordinates" , and if coordinates are found we take the midpoint of these coordinates as a central point to map data and compute against necessary regression models.

```
function (doc) {
  if(doc.place.bounding_box.coordinates){
     var x1 = doc.place.bounding_box.coordinates[0][0][0];
     var y1 = doc.place.bounding_box.coordinates[0][0][1];
     var x2 = doc.place.bounding_box.coordinates[0][1][0];
     var y2 = doc.place.bounding_box.coordinates[0][1][1];
     var x3 = doc.place.bounding_box.coordinates[0][2][0];
     var y3 = doc.place.bounding_box.coordinates[0][2][1];
     var x4 = doc.place.bounding_box.coordinates[0][3][0];
     var y4 = doc.place.bounding_box.coordinates[0][3][1];
     emit( [((x1+x2+x3+x4)/4),((y1+y2+y3+y4)/4)], doc.sentiment.compound);
```

```
  }
}
```

9.2 Reduce:

Using the predefined "reduce": "_count" function to give the total count of number of tweets with respect to geographic coordinates.

10 Web app and Visualization

The web app is build by mainly using CanvasJS which is a HTML5 and Javascript charting library. The library is embedded in html code on client side by calling the remote CanvasJS library online.

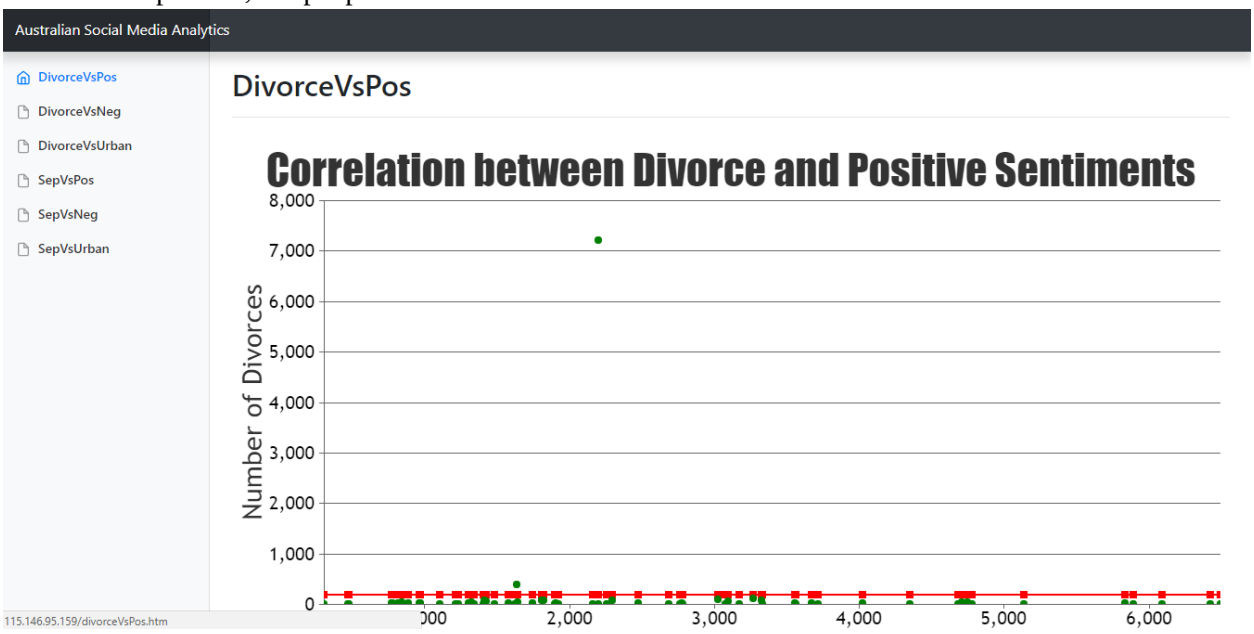
When users enter the web address to access the web app, CanvasJS will connect to server to get data in JSON format, and then render charts on web page. For the project, in order to present the result after analysing, CanvasJS will plot both scatter chart and line chart in the same graph.

For the scatter chart, the points illustrate the correlation between elements such as the number of divorces, the number of positive and negative sentiment tweets while the line graph shows the line of linear regression of this correlation.

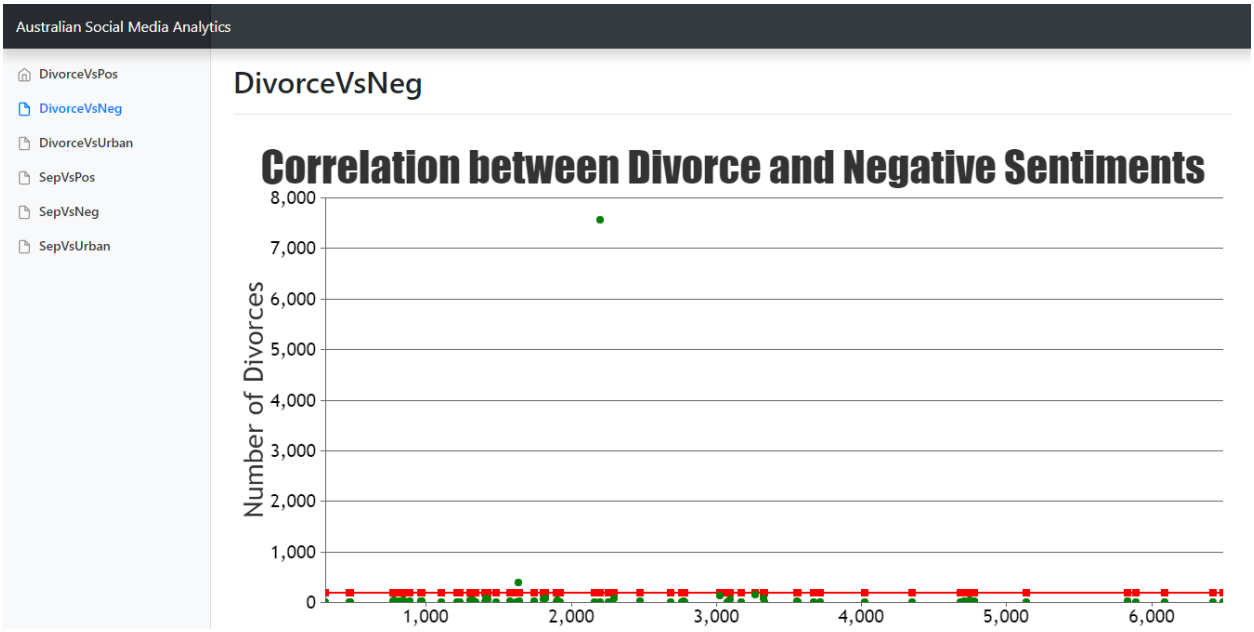
For implementation, the web app is deployed and run on Apache web server and the web app will display total 6 graphs.

Visualization snapshots:

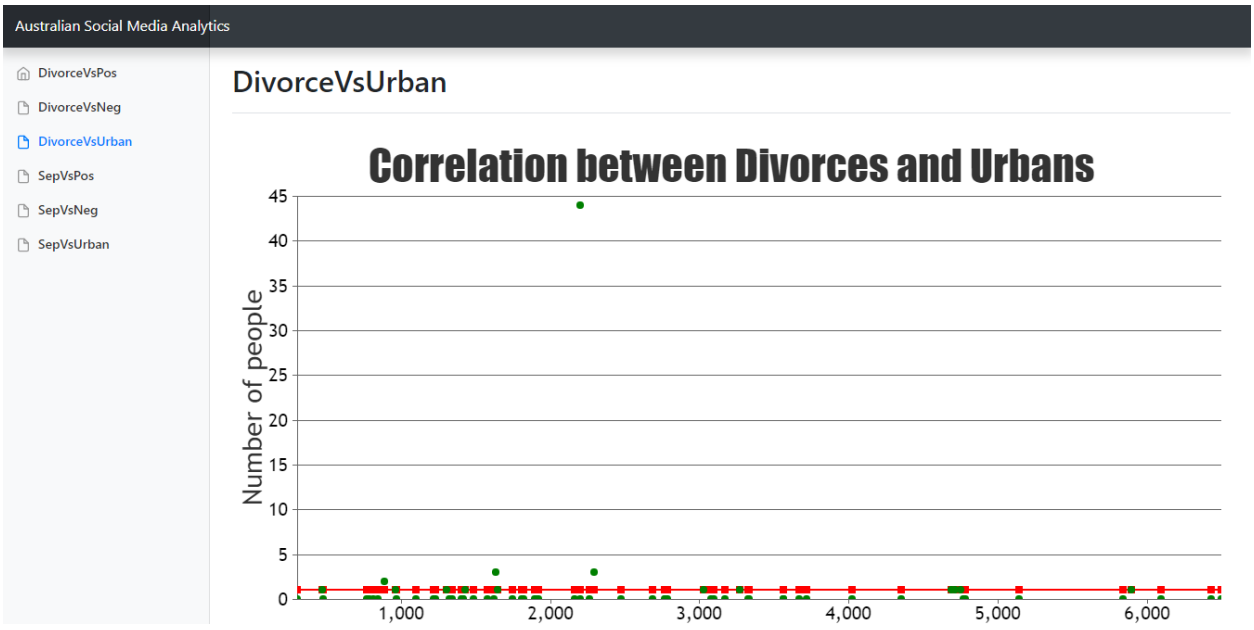
Divorce rate per 100,000 people Vs Positive Sentiments tweet data



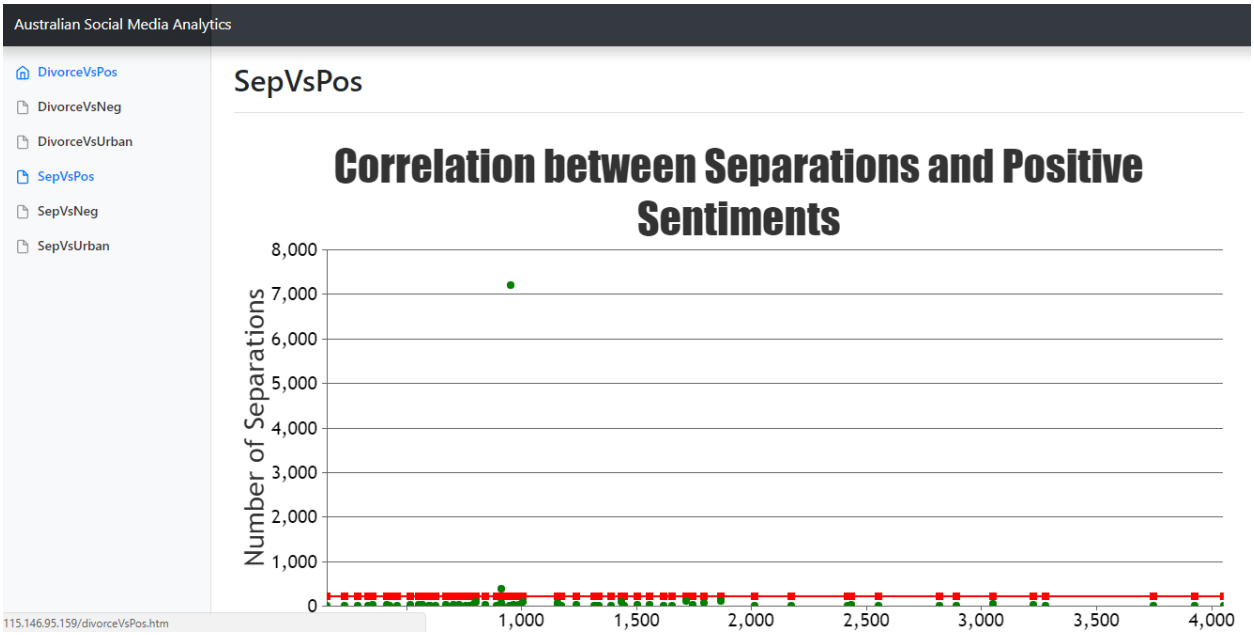
Divorce rate per 100,000 people Vs Negative Sentiments tweet data



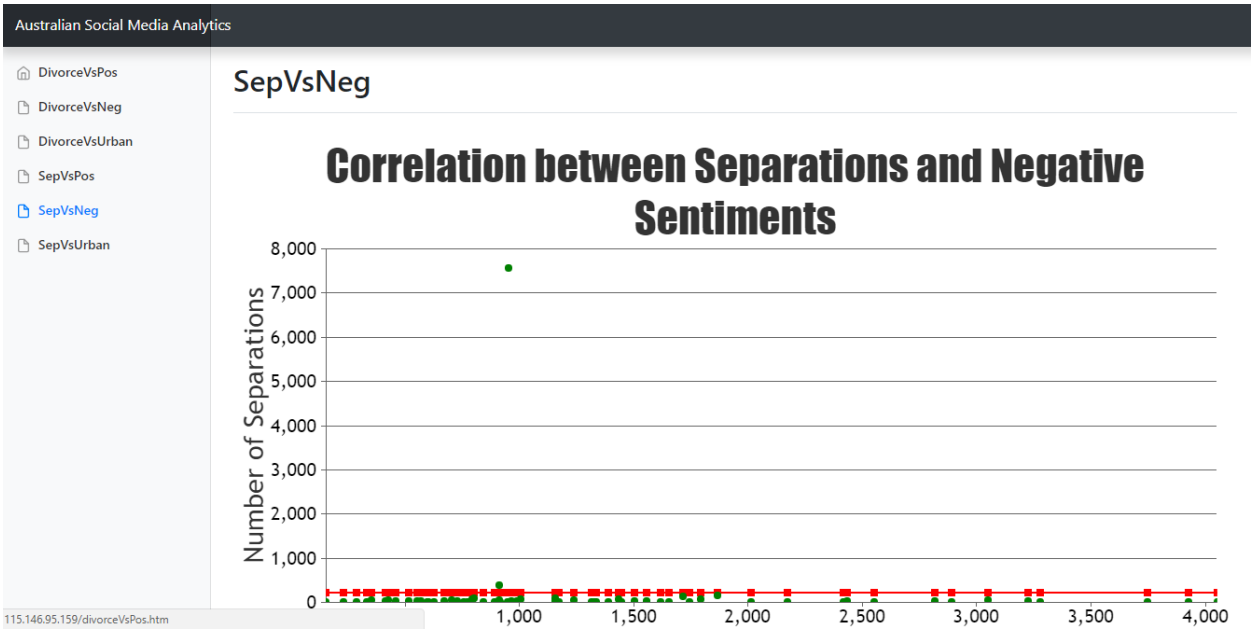
Divorce rate per 100,000 people Vs Urban Influence tweet data



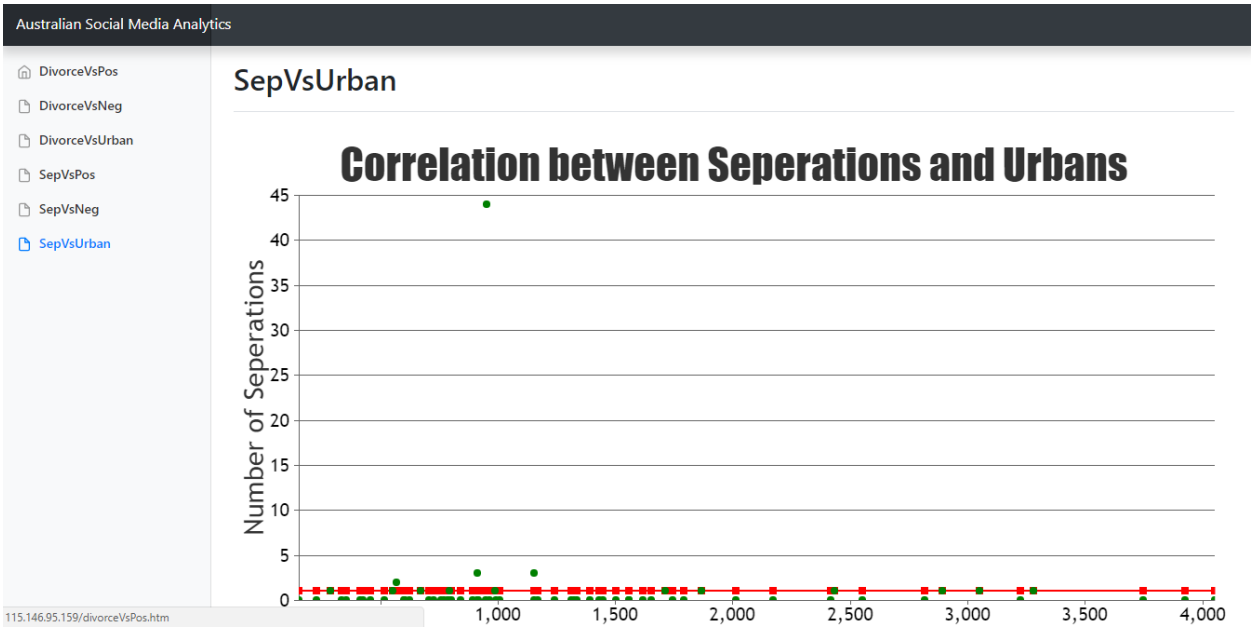
Separation rate per 100,000 people Vs Positive Sentiments tweet data



Separation rate per 100,000 people Vs Negative Sentiments tweet data



Separation rate per 100,000 people Vs Urban Influence tweet data



11 Regression Analysis of data

Ordinary least squares (OLS) regression is a statistical method of analysis that estimates the relationship between one or more independent variables and a dependent variable. This method estimates the relationship by minimizing the sum of the squares in the difference between the observed and predicted values of the dependent variable configured as a straight line.

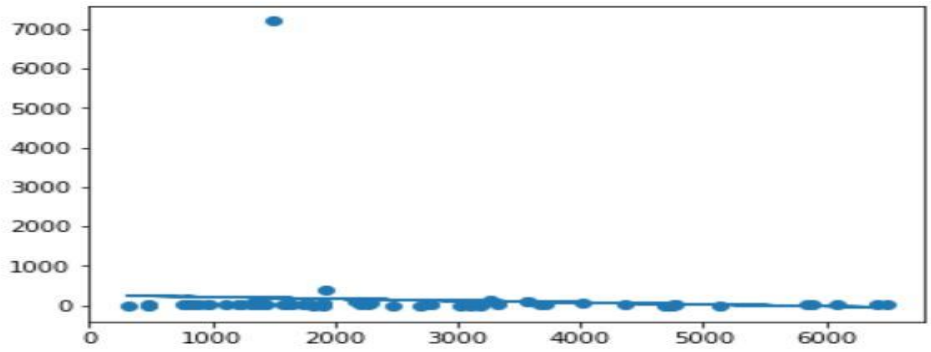
OLS regression analysis was done against data sets used for the project.:

- Number of Divorces were measured against positive sentiment tweets: They were found to be negatively correlated.
- Number of Divorces were measured against negative sentiment tweets: They were found to be positively correlated
- Number of Divorces were measured against urban and pop culture tweets: They were positively correlated, but in this case we had a smaller data set for Urban Pop Culture Tweets, and hence the drastic correlation.
- Number of Separations were measured against positive sentiment tweets: They were found to be negatively correlated.
- Number of Separations were measured against negative sentiment tweets: They were found to be positively correlated
- Number of Separations were measured against urban and pop culture tweets: They were positively correlated, but in this case we had a smaller data set for Urban Pop Culture Tweets, and hence the drastic correlation.

11.1 Results

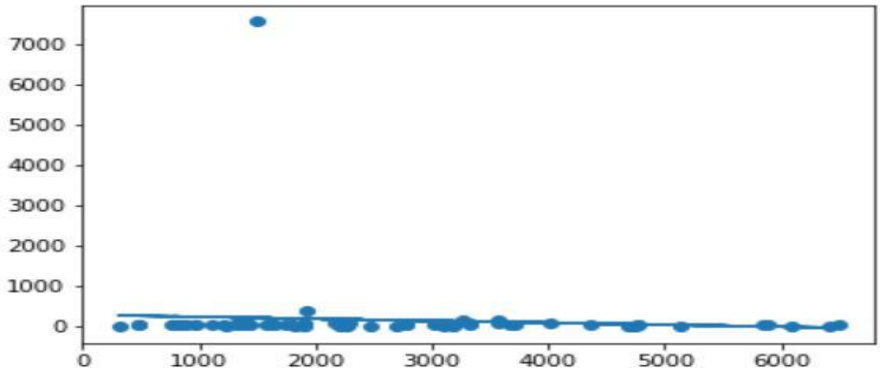
1) Number of Divorces were measured against positive sentiment tweets.

```
# graph of divroce rate(x) vs postive sentiments(y)
plt.figure(1)
plt.scatter(pmDivorce, posSentiment)
m, b = np.polyfit(pmDivorce,posSentiment,1)
#print(m,b)
plt.plot(pmDivorce,m*pmDivorce+b, '-')
plt.show()
```



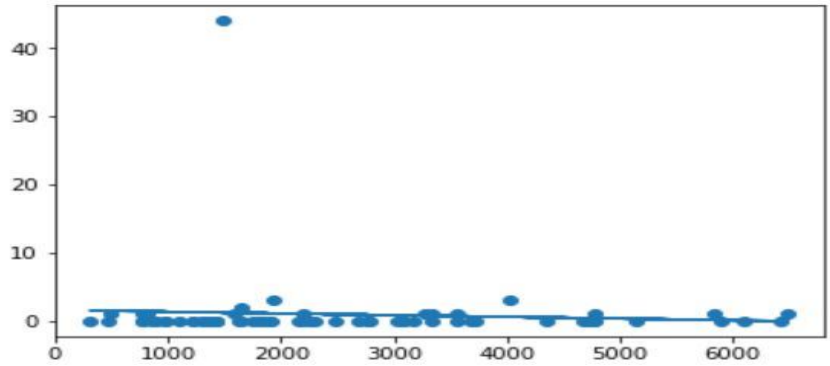
2) Number of Divorces were measured against negative sentiment tweets.

```
# graph of divorce rate(x) vs negative sentiments(y)
plt.figure(1)
plt.scatter(pmDivorce, negSentiment)
m, b = np.polyfit(pmDivorce,negSentiment,1)
#print(m,b)
plt.plot(pmDivorce,m*pmDivorce+b, '-')
plt.show()
```



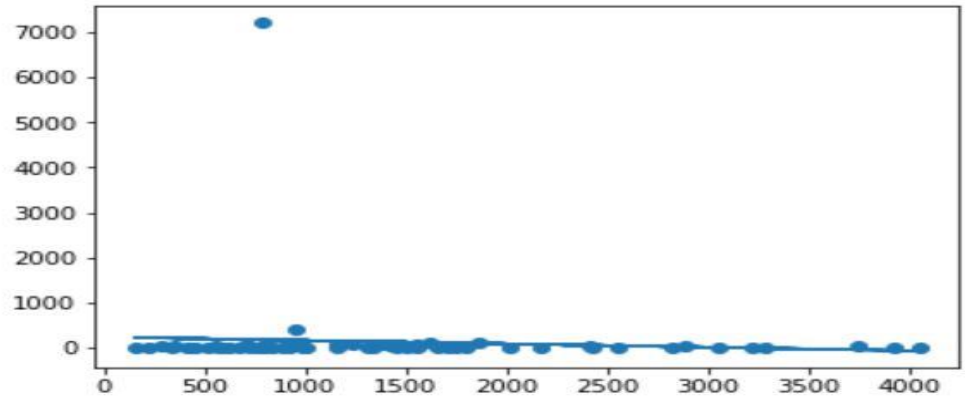
3) Number of Divorces were measured against urban/pop culture tweets.

```
# graph of divroce rate(x) vs urban pop culture (y)
plt.figure(1)
plt.scatter(pmDivorce, urbanCulture)
m, b = np.polyfit(pmDivorce,urbanCulture,1)
#print(m,b)
plt.plot(pmDivorce,m*pmDivorce+b, '-')
plt.show()
```



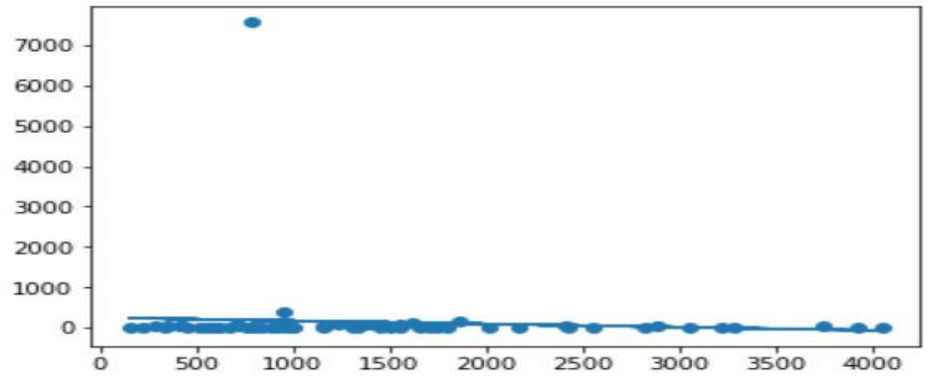
4) Number of Separations were measured against positive sentiment tweets:

```
# graph of separation rate(x) vs positive sentiments(y)
plt.figure(1)
plt.scatter(pmSeparation, posSentiment)
m, b = np.polyfit(pmSeparation,posSentiment,1)
#print(m,b)
plt.plot(pmSeparation,m*pmSeparation+b, '-')
plt.show()
```



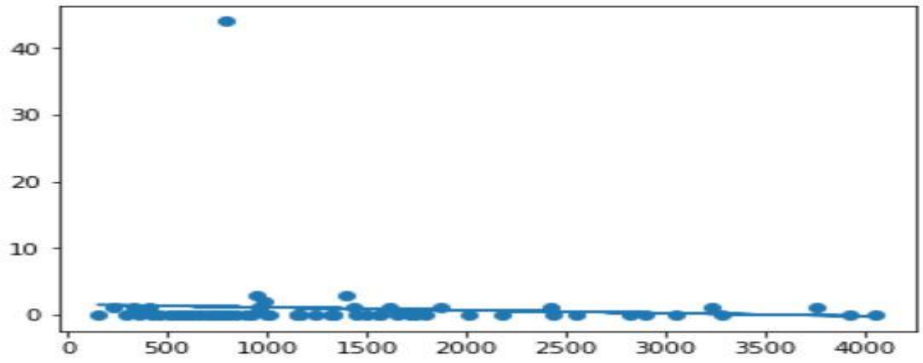
5) Number of Separations were measured against negative sentiment tweets.

```
# graph of separation rate(x) vs negative sentiments (y)
plt.figure(1)
plt.scatter(pmSeparation, negSentiment)
m, b = np.polyfit(pmSeparation,negSentiment,1)
#print(m,b)
plt.plot(pmSeparation,m*pmSeparation+b, '-')
plt.show()
```



6) Number of Separations were measured against urban/pop culture tweets.

```
:  # graph of separation rate vs urban pop culture
   plt.figure(1)
   plt.scatter(pmSeparation, urbanCulture)
   m, b = np.polyfit(pmSeparation,urbanCulture,1)
   #print(m,b)
   plt.plot(pmSeparation,m*pmSeparation+b, '-')
   plt.show()
```



11.2 Conclusion

1)The OLS Regression results are interpreted as follows:

1. For every one increase in a positive tweet count, you can expect on average 11.6007 decrease, in average annual rate of divorce per 100,000 people.
2. For every one increase in a negative tweet count, you can expect on average 9.0891 increase, in average annual rate of divorce per 100,000 people.

3. For every one increase in a positive tweet count, you can expect on average 3.2721 decrease, in average annual rate of separation per 100,000 people.
4. For every one increase in a negative tweet count, you can expect on average 2.6287 increase, in average annual rate of separation per 100,000 people.

2) The system is designed to make it fault tolerant and easily scalable due to being hosted on NeCTAR cloud. The three node cluster architecture of the database makes it fail safe. Thus, if any one of the virtual machines fail, there won't be a single point of failure.
3) The auto deployment script can easily be modified to scale up the operations. To collect more data, the storage capacity associated with the VM could be increased on demand.

12 Team Member Contribution

Name	Contribution
Prarthana Padia	Building Scenario, Working and Deploying Twitter Harvesters, Sentiment Analysis, Working on Analysis, + Collaboration
Pallabi Bhattacharya	Working on harvesting tweets from users, Adding Scenario keywords Uploading AURIN datasets to CouchDB, Report Writing, + Collaboration
Bhavya Singhal	Building Scenario, AURIN dataset collection and building geo-tagged datasets, Infrastructure Provisioning, Automation using Boto and Ansible scripts, + Collaboration
Khanh Tan Nguyen	Webapp, Data Visualization, +Collaboration
Shajid Mohammad	Deploying CouchDB cluster on Docker, Working on Analysis and Map-Reduce on CouchDB, + Collaboration

13 User guide

This section guides through the deployment, and the usage of the system presented in this report. It includes the system deployments, end user invocation for the retrieval of the views, and the access to the virtual machines.

13.1 End user invocation: Manual deployment of harvester

To start the twitter harvester manually, and start doing sentiment analysis and storing json object to couchDB, download and store the private key (cloudkey)
and store it on local machine. Then follow the following steps:
Step 1: Go to terminal or GitBash and type
ssh -i cloudkey -L 5984:127.0.0.1:5984 ubuntu@115.146.95.146
to access the Instance1 after giving the passphrase as input
Step 2: Go to the directory
/volume/twitter_harvester/app
to find the main script harvester1.py.
Step 3: Type
screen python3 harvester1.py
So that we can detach the screen and keep the script running in the background.

13.2 Accessing Python Notebook for Regression Analysis

Step1: ssh -i cloudkey -L 8000:localhost:8888 ubuntu@115.146.95.146

Step2: Click on token: http://localhost:8000/?token=a30fa4afe693ab4057263f614de2be7349aecc858fa67ea0

Step 3: Open the link in your browser: http://localhost:8000/notebooks/Untitled.ipynb?kernel_name=python3#

14 URL To access

BitBucket:
https://bitbucket.org/shajid/australian-social-media-analytics/src/master/
Youtube:
Automation: https://youtu.be/bkjth2cIxBA
Webapp: https://www.youtube.com/watch?v=Yrap2bBTUyU&feature=youtu.be
Regression Analysis: https://youtu.be/t9tNML78rao

15 References:

1. Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA,

2. Bing Liu, Minqing Hu and Junsheng Cheng. "Opinion Observer: Analyzing and Comparing Opinions on the Web." Proceedings of the 14th International World Wide Web conference (WWW-2005), May 10-14, 2005, Chiba, Japan.
3. C.J. Hutto Eric Gilbert. “VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text”