COMP90024 Cluster and Cloud Computing Assignment 2 Group 18 Project Report

Australian (Victoria) Social Media Analytics

Yixiong Ding	671499
Yijie Mei	861351
Tiange Wang	903588
Wuang Shen	716090
Ruifeng Luo	686141

1 May, 2017

Lecturer: Richard Sinnott

Contents:

1. **Introduction ……………………………………………………………..…..3**
2. **System description…………………………………………………………...4**
3. **System architecture………………………………………………………..…..7**
4. **Front-end design and Visualization…………………………………..….9**
5. **NeCTAR Research Cloud…………………………………………...……...15**
6. **Sentiment analysis…………………………………………………….…….16**
7. **Conclusion…………………………………………………………...……19**
8. **User Guide……………………………………………………………..…….19**
9. **Team members and roles……………………………………..…………….23**
10. **Reference………………………………………………………………........24**

1. Introduction

There are vast amounts of active Twitter users and tweet data in Australia. Based on the result of up-to-date research, in January of 2018, the Twitter attracted 3 million of active Australian users (Cowling, 2018). The thousands of tweets they posted provide various information which could be utilised and mined. According to the numbers of valid data caught by the web crawler, average 10% of them attached to the coordinates which are valuable for the groups and society to process the area-based sentiment analysis. With the abundant data and the support of big data technique, the researchers of the group designed the cloud computing architecture with related methods to implement the analysis, including National eResearch Collaboration Tools and Resources (NeCTAR), Virtual Machines (VMs), Australian Urban Research Infrastructure Network (AURIN) and the open source database CouchDB. The NeCTAR provides the cloud services and infrastructure for this project. And the VMs are utilised to run the script, harvest the valid tweets data and coordinate with CouchDB to store the data with the format of JSON. In addition, the AURIN also offer some significant data for sentiment analysis aspects, such as age distribution and income circumstance in the Victoria state, Australia.

The report illustrated that how to harvest, store and analyse tweets data through the cloud-based system. The following section introduces the system capabilities. In section 3, it demonstrates the system infrastructure. The section 4 demonstrated the front-end design and the visualization while the section 5 discusses the pros and cons of NeCTAR. In the section 6, the process of sentiment analysis, we invoke a sentiment library named vaderSentiment to engage the analysis and primarily discover the relationship between the demography of young people in a city in the Victoria state and the number of tweets which are posted in this area, and the correlation between the income, household size and the happiness across the cities in the Victoria state. The section 7 summaries the conclusion while the last section is the guide for users to deploy and implement applications in the project.

The crawler and script we used are stored in the Github repository: https://github.com/Menooker/DonauTwitter Meanwhile, to browse the demonstration, please visit our website for the project: http://115.146.86.137/wordpress/

2. System description

It is essential to store these filtered valid tweets for further analysis. Therefore, with the feature of supporting the storage of a large amount of data and fast retrieval speed, CouchDB becomes the choice and is used as preferred storage application. CouchDB is an open source document-oriented database management system which uses JSON as the storage format, MapReduce as API and JavaScript as the query language (En.wikipedia.org, 2018). CouchDB allows to store the mass of data, including AURIN data and tweets data and is easy to distribute the data to several physical nodes and coordinate and synchronise the consistency of subdata reading and retrieval between nodes for next operation based on the feature of the distributed system.

The system primarily focuses on the implementation of tweet data filter and harvest and valid data storage. In the stage of preparation, follow the guideline of ansible playbook, four VMs with different roles are installed and initialised to deploy to cooperate with the NeCTAR through the Python package Boto 3 which offers the interface for cloud services. Two of the VMs are utilised for harvest function, and CouchDB storage since the workload of the harvest process is the heaviest. And for remaining two ones, one is for the web, and the other is used to analyse. The responsibility of the CouchDB is to store not only the valid data from the harvest VM but also the AURIN data which are in the JSON format with sufficient geographic information, to support the subsequent analysis.

In the part of front end design, the fundamental technique is Wordpress accompanied by PHP and MySQL based. The web primarily presents the interface, the research topic and the navigation to show different functions which are realised in this project. Then, in the sentiment analysis aspect, it reads data stored in the CouchDB and runs the process of analysis in Python with the package vaderSentiment. These two crucial processes will be illustrated in the next sections.

2.1 The harvest for Tweet data

We utilised the Python package named TwitterAPI to help us harvest the valid tweet data. This package on the GitHub is with abundant features which provide various searches and application based on keyword, location, coordinate and user id (GitHub, 2018). Thus, in the stage of the crawler, we selected it as the tool for harvesting the tweets with coordinates. In the process of harvest, TwitterAPI offers two options to collect different periods of tweets, Streaming API and REST API (Search API). The Streaming API primarily focuses on the intraday and real-time tweets which are not able to present sufficient tweet data for us. The REST API supports developers to access and read tweet data which posted in the past 7 days. Therefore, for obtaining the sufficient amount of data, we created a Python program named TwitterCrawler and used both of them to optimise the effectiveness of the harvest. Meanwhile, we applied harvest and CouchDB on the same VM to ensure that the valid tweet data would be delivered and stored into the CouchDB directly. And the system allows multiple harvest and CouchDB to run concurrently. For collecting the tweet with the geographic information, we utilised the search method which determines a location with coordinates as the center of the search circle and a fixed length as the search radius, and any tweets in the past 7 days in this area which satisfy the search criteria would be captured and stored in the CouchDB. For instance, suppose that we set the centre of Melbourne (-37.81 latitude and 144.96 longitude) as the centre of search circle and 250 km as the radius. Then run the crawler, we could obtain the majority of valuable tweets in Victoria state since many Victoria active Twitter users live in this circle.

In particular, the TwitterAPI package has some limitations.

- Duplicates of tweets

In the process of harvest, the crawler may encounter and collect the repeated tweets then deliver to the CouchDB which influence the analysis result and accuracy. Through several tests, we found that adding the parameter 'tweet id' in the crawler is able to filter reduplicate tweets effectively. The specific process is achieved through the CouchDB which assign the document id '_id' to equal 'tweet id' for each tweet. Meanwhile, in the Python program, we used 'try-catch block' to detect the duplicate error as the CouchDB reject the insertion operation of two same document id. Once it triggers the error, the 'try-catch block' would ignore this tweet and capture the next target tweet.

- Request limitations

According to the specification of TwitterAPI, for Search API, after a user login with the key and secret, it allows one user to send 180 requests per 15 minutes with the maximum of 100 tweets for each request. In other words, it permits one user to capture 72000 tweets per hour. Once the number of requests exceeds the frequency limitation, the API will present the error 'HTTP 429: too many requests.'. Considering the low proportion of tweets with coordinates in the total number of tweets, if researchers would like to gain sufficient amount of tweet data, they have to calculate the probable time cost.

2.2 Search API & Streaming API

As introduced above, Search API and Streaming API compose the TwitterAPI package. The Search API is used to review and capture tweets data in the past 7 days. Apart from the limitations for the number of requests, the access of some tweets is restricted since not all of the tweet data are accessible and captured. Meanwhile, it focuses on matching the search condition (relevance) rather than all the eligible tweets (completeness) as some users and tweets will be missing during the usage of Search API (Developer.twitter.com, 2018).

By contrast, Streaming API avoids some of these limitations. The Streaming API offers a low- latency access to tweet data for developers (Developer.twitter.com, 2018). In addition, it is real-time search and harvest and support to collect the complete intraday tweets (Developer.twitter.com, 2018). Therefore, for intraday tweet data, the Streaming API will theoretically capture more valid tweets than the Search API. Nonetheless, these tweet data are only the fraction of numerous eligible tweets.

2.3 CouchDB

The CouchDB database is based on the document with high scalability, availability and reliability (En.wikipedia.org, 2018). The MapReduce function which embedded in the CouchDB is the only method to browse the data. The MapReduce provides convenience for developers with insufficient experience of distributed system programming to implement the software on the distributed system (En.wikipedia.org, 2018). It combines the function 'map' with the function 'reduce', which maps and create a new set of key-value pairs and utilises concurrent 'reduce' function to confirm that each mapped key-value pair to share the identical key set (En.wikipedia.org, 2018). In this project, we used it in the CouchDB to filter the tweets without geographic information.

3. System architecture

Shown in the sketch diagram below (Figure 1) is the overall architecture of the system. First, we allocated four VMs to process the harvest/CouchDB, web and analysis. The harvester/CouchDB works in the pair to enhance the efficiency and number of valid tweets. As shown, the harvester captures raw tweets from Search and Streaming API then stores them as the documents into the CouchDB. For the VM of the analyser, it receives the tweet data which are processed by the MapReduce in the CouchDB and begins to engage the sentiment analysis. Then it delivers the analysed tweet data to the VM for the web. The last VM of web reads the data from the console and presents to the browsers as the format of tables and charts.

In the CouchDB, the harvester delivers the tweet data into it and stores the raw data as the document. AURIN data could also be applied to store in the CouchDB and is delivered with the document in JSON format to the analyser. However, in our project, the AURIN data are downloaded directly and transferred to the analyser and the web which skipped the stored procedure.

In addition, if it is necessary to handle more complex data or process further study, the system architecture is scalable. Firstly, the number of VMs is easily increased. Hence, for some complicated circumstances, developers do not worry about the efficiency of the harvester too much as they are able to deploy more harvesters and CouchDB. Secondly, there exist external storages in each set of harvester and CouchDB in order to expand the capacity of the database. Meanwhile, the MapReduce function could not only filter the tweets without the coordinates but also be applied to some simple sentiment analysis. Then, the more massive system is able to operate as the description above. Multiple harvesters handle tweet data and deliver to the CouchDB. Each VM could call the MapReduce function to process the basic analysis, such as the analysis on tweet posted time.

In the environment outside the system, basically, many familiar analytical programming languages own libraries for CouchDB, such as Python, Java...etc. Thus, it is convenient to proceed further analysis.

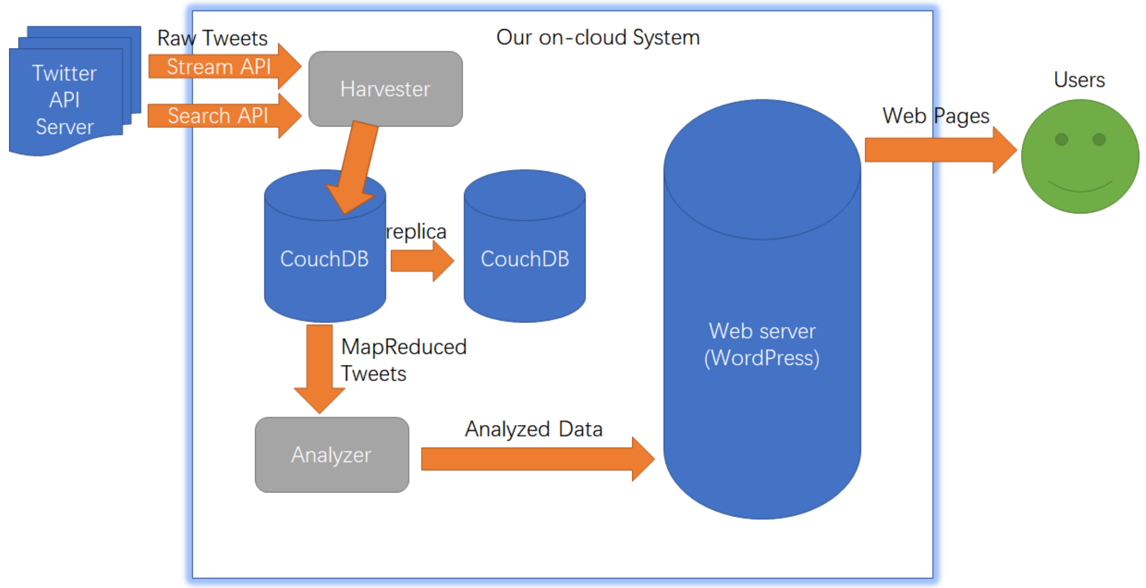

Figure 1. The sketch diagram of the system architecture.

The project contributes to design and achieve the architecture above. The realisation of specific functions for each VM and the deployment for the VMs and NeCTAR will be discussed in the following sections.

4. Front end design and Visualization

WordPress is an open-source content management system (CMS), which is built based on MySQL database and PHP server, in this project we employ WordPress to develop our front-end system and data visualization.

4.1 Front end Setup

As other front-end systems, WordPress has to be installed on the web server. There are two circumstances:

1. The web server is an Internet hosting service, for example, WordPress.com;
2. The web server is a local network host in its own right, in this case it could be a computer running the package WordPress.org. The web server is a local network host in its own right, in this case, it could be a computer running the package WordPress.org.

Since we use the cloud services provided by the NeCTAR in this project, therefore we install and set up the WordPress system as the circumstance 2 in the following steps:

1. Use scripts to deploy the necessary supports on the cloud server, which are:
   - PHP version 7.2.4-1
   - MySQL version 14.14
2. Create a database for WordPress system and also a relative MySQL user with all privileges to access and modify it;
3. Find and edit the system file `wp-config.php` to add the database information, such as Database Name, Database Username, Database Password and Database Host, etc;
4. Upload the WordPress files by using Filezilla accessing the IP address of the cloud server and run the installation script by accessing the URL in the web browser. The URL is the same as the path where we put the WordPress files on the server, for example:
   - If WordPress files are in the root directory, the URL is : `http://example.com/`
   - If WordPress files are in the subdirectory `test`, the URL is : `http://example.com/test/`

   As we put the WordPress files in the subdirectory `wordpress` and a copy in the root directory, our front end can be accessed by the both URLs:
   - http://115.146.86.137
   - http://115.146.86.137/wordpress

4.2 Data Visualization

Data visualization is the process of combining data from different sources in order to develop the virtual and logical view of information so that front-end applications are able to access these information. Users are able to view information using front-end applications. In data visualization, we retrieved data stored in the cloud and make it into useful information on the web based on the three interesting scenarios we chose. In the section 4.2.2 and 4.2.3, we are demonstrating approaches to visualize each different data, AURIN and Harvested tweet data.

4.2.2 AURIN Data

AURIN (Australian Urban Research Infrastructure Network) is the collaborative network of leading researchers and data providers across the academic, government, and private sectors, which provides accurate and trustworthy data of urban information around Australia. In order to analyse harvested Twitter data, we download relevant data to make the comparison.

When handling AURIN's data, the datasets we try to find is based on Statistic Area 3 that including five cities - Melbourne, Geelong, Bendigo, Ballarat, Melton. It focuses on the three different datasets list below. Afterward, we convert each retrieved data from aurin into the different visual format so that it is easy to use for comparison to harvested Twitter data based on scenarios.

Weekly personal income:

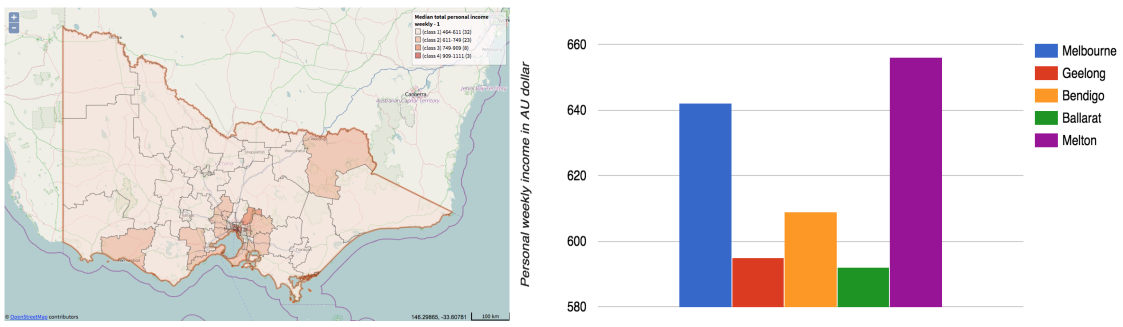

Figure 3. AURIN data - personal income.

Persons with higher education:

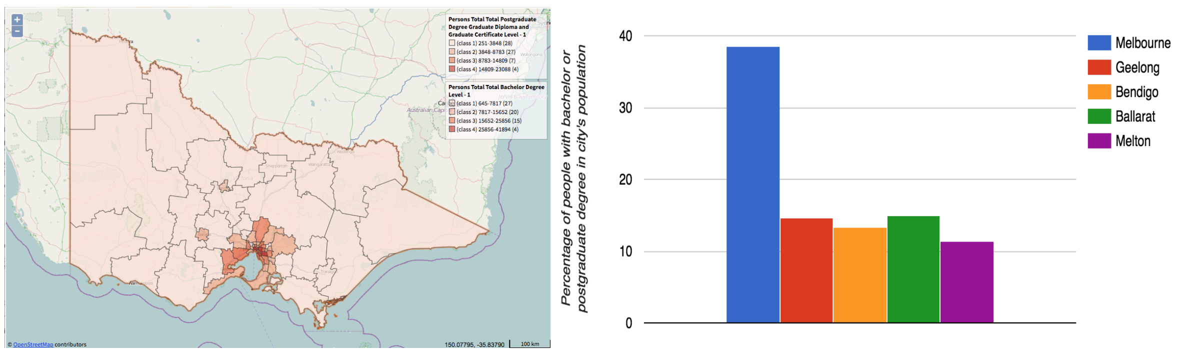

Figure.4 AURIN data - people with higher education.

Median age of persons:

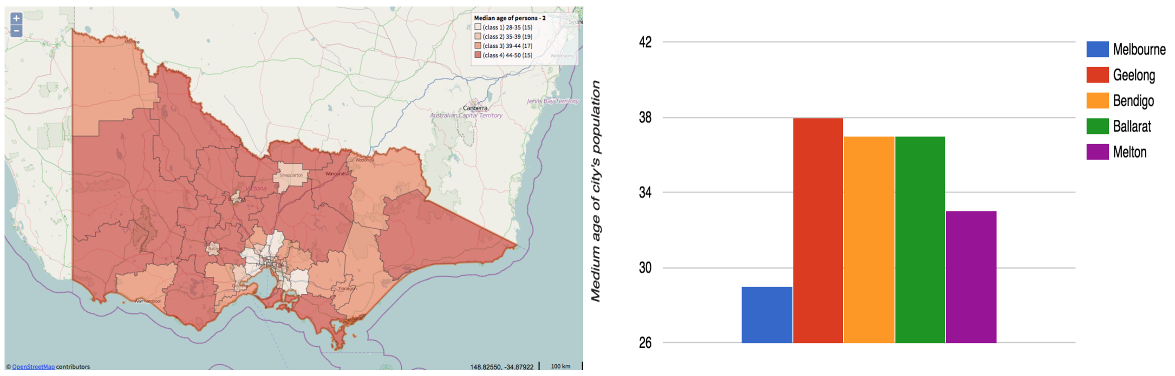

Figure.5 AURIN data - Median age of persons.

4.2.3 Harvested Twitter Data

According to totally crawled 2,961,080 tweets, there are top five cities listed with the ranking of the number of tweets relevant. And the picture below is the data visualization of the related Twitter data.

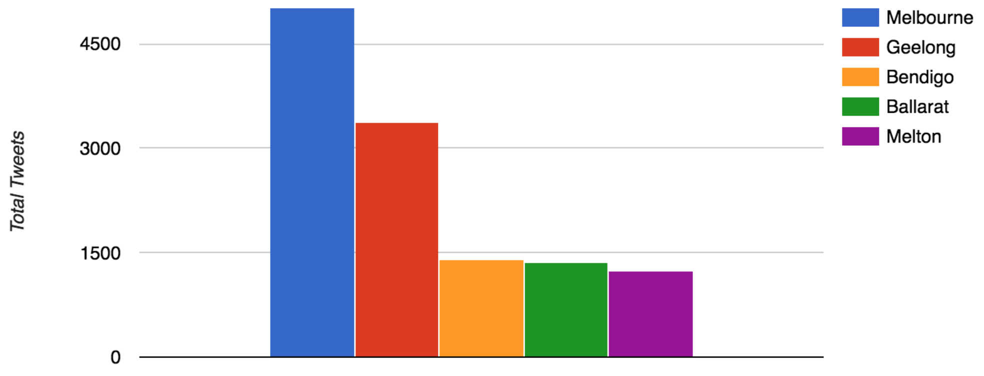

Figure. 6 Total tweets.

5. NeCTAR Research Cloud

5.1 NeCTAR Research Cloud Analysis

This section discusses the benefits and the issues of the NeCTAR Research Cloud.

5.1.1 Pros of NeCTAR Research Cloud

1. Using on-demand computation and storage resources without having to purchase and maintain the hardware on our own.
2. Relatively low cost and being open to researchers (with free trial!)
3. Users can in a group can cooperate by sharing one virtual machine instance at the same time. Users can share the image with pre-installed software with others.
4. Accessible by the whole Internet with fixed IP (which is good for researchers who want to host a server on the cloud)
5. Easy backup utilities – for both VMs and volumes
6. Easy security and access control
7. Support for AWS API – easy to dynamically manage with scripts and easy to scale

5.1.2 Cons of NeCTAR Research Cloud

1. The web interface of the cloud often responds slowly (in minutes)
2. Potential data loss is possible, as root disk (vda) and Ephemeral Disk (vdb) are persisted only when termination or snapshotting of the instance.
3. Amazon AWS API for NeCTAR used by Boto does not support tagging the instances on creation of them. As a result, random names are tagged to the instances created by Boto scripts.
4. Amazon AWS API for NeCTAR cannot return the IPs of the instances right after they are newly created. Thus, the Boto script have to wait for some time and re-fetch the information of the instances to get the IPs and the IDs of the new instances. It is not efficient if large number of instances are created.

5.2 Image creation and deployment

Image creation and deployment are done in the script “deploy/deploy.sh” in the source package. The script first invokes a python script “CreateInstance.py” which is based on Boto. And it then calls several Ansible scripts to install the harvester, web server and the analyzer as well as their dependencies.

The script “CreateInstance.py” loads the configurations from the file “config.json”, which includes tokens and the keys for the AWS API of NeCTAR. It also includes the parameters for creating the VMs, such as image name, number of VMs, the size of the volumes and so on.

The details of the configuration file have been shown in the previous section. The script utilizes Boto library to create the VM instances and the volumes by the user configurations, and attach the volumes to the respective VMs. Finally, it outputs a host file named “hosts”, which can be used by Ansible. The host file includes the types of the VMs (DB, web or analysis) and their IPs.

After creating instances, the “deploy.sh” script sleeps for a minute to wait for the VMs to boot. Then it invokes the Ansible scripts in the “deploy/playbook” directory, including “basic.yml” for mounting the volumes and basic packages that should be installed in all VMs, “couchdb.yml” for CouchDB installation and configuration, “harvester.yml” for harvester and analyzer deployment and “webserver.yml” for web server deployment. Note that the script installs CouchDB 2.1.1 by Docker and connects the DB nodes (currently two nodes) into a CouchDB cluster.

6.Sentiment Analysis

Sentiment analysis is supported by the system to find how residents’ emotions vary with places they are living. Targets will be restricted to five cities in Victoria State (Melbourne, Geelong, Bendigo, Ballarat and Melton) for this project. Each tweet will be classified as a tweet with negative, positive or neutral emotion based on tweet’s text content. External python package – vaderSentiment will be applied to handle the task of tweet’s sentiment classification (Hutto & Gilbert, 2014).

6.1 vaderSentiment package

vaderSentiment is a sentiment analysis tool designed for analyzing sentiments shown in posts from social media (like Tweeter). Instead of searching for simple terms appeared in sentences, vaderSentiment can deal with more complex situations like negation, emotions, emojis, initialisms and slang words. A method - polarity_scores inside of a Class called SentimentIntensityAnalyzer in the package will be used to calculate the negative, positive and neutral scores for each tweet. These scores present how many components of a sentence being classified into these three emotions.

Classification rules for tweets:

1. Positive score – Negative score > 0 => Positive sentiment.
2. Positive score – Negative score < 0 => Negative sentiment.

As we are only interested in relative sentiment for each tweet, so if a tweet's positive score is higher than negative score, it will be classified as positive sentiment, even if it's neutral score may be higher.

6.1.2 Limitation of vaderSentiment package:

VaderSentiment tool does not support 100% accuracy to the system, so the tool may not work well on some tweets. For example, vaderSentiment can't identify malformed word or typo in the sentence, so sentences with this kind of words are highly likely to be classified as neutral tweets.

6.2 TwitterAnalytics python file

TwitterAnalytics.py is the main script used to extract information from harvested tweets data. Each city will be an object of Area class, with name, totalCount (total amount of tweets), posCount (total amount of positive tweets) and negCount (total amount of negative tweets) attributes. Cities, where tweets sent from, can be easily detected by "location" tag of tweets. TotalCount will be updated after detected a tweet from the same city, while posCount and negCount will be updated respectively if a positive tweet and negative tweet being detected by vaderSentiment tool.

6.3 Fetching the harvested data with MapReduce

The harvester saves every tweet it gets from Twitter into the CouchDB, but the analyzer is only interested in tweets containing location information. It is observed that only 6.1% of the fetched tweets are tagged with location (180921 tweets of 2961080 tweets in total). Thus, it is inefficient to iterate over all tweets in the database. The analyzer will first create a view using MapReduce API of CouchDB. To support creating views by Python in CouchDB, the project constructs a Python module called "couchdb_map" located in "analytics" directory in the source package. The module provides a utility function "map2" to easily create a view with MapReduce and fetch the documents in the view. The function takes a JavaScript function in string as the map function and a Python function that will iterate all documents in the view. The analyzer uses this function to filter and fetch all tweets with location. The map function is shown below:

```
function(doc){
        if(doc.location!=undefined){
        emit(doc._id)
        }
}
```

6.4 Analysis result:

Sentiment/City	Melbourne City	Geelong	Bendigo	Ballarat	Melton
Positive	37.62%	39.23%	37.63%	41.91%	12.04%
Negative	17.45%	17.74%	19.77%	33.38%	5.86%
Neutral	44.93%	43.04%	42.59%	24.71%	82.10%
Positive/Negative	2.16	2.21	1.90	1.70	2.05

Figure. 7 Sentiment portions for each city

We are more interested in relative happiness (Positive/Negative) in our scenarios, as this ratio can get rid of the noise from Neutral portions, and more precisely present people's happiness. Therefore, the happiest city will be the one with highest Positive/Negative ratio rather than the one with the highest positive portion or lowest negative portion.

As shown in the table, most city's positive portions and negative portions are close to 40% and 19% respectively. Melbourne City, Geelong, and Bendigo have similar portions of negative, positive and neutral tweets. Ballarat has highest portions of negative and positive tweets, while Melton has lowest portions of negative and positive tweets as it has highest Neutral portions. Regarding Positive/Negative ratio, Geelong has the highest ratio, while Ballarat has the lowest ratio. Therefore, the rank of cities regarding happiness will be Geelong(1st), Melbourne City(2nd), Melton(3rd), Bendigo(4th) and Melton(5th).

6.5. Scenarios Analysis

In this section, we will combine harvested result and AURIN data to explore whether there exist correlations in designed scenarios.
Extra scenarios for this project are:

· Are people happier (express more positive sentiment) in areas with more income?
· Are people happier (express more positive sentiment) in areas with younger population?
· Are people more likely to send tweets in areas with higher portion of residents with higher education?

Besides, only one factor is considered for each scenario. However, in the real world, people's sentiment and behavior are affected by many different factors. It is normal that targeted cities' residents are affected by factors (like income, age, and education level) in a different level. Thus, the analysis will be more focused on the main trending showed in the result.

6.5.2 Scenarios Results

Scenario 1: Are people happier (express more positive sentiment) in areas with more income?

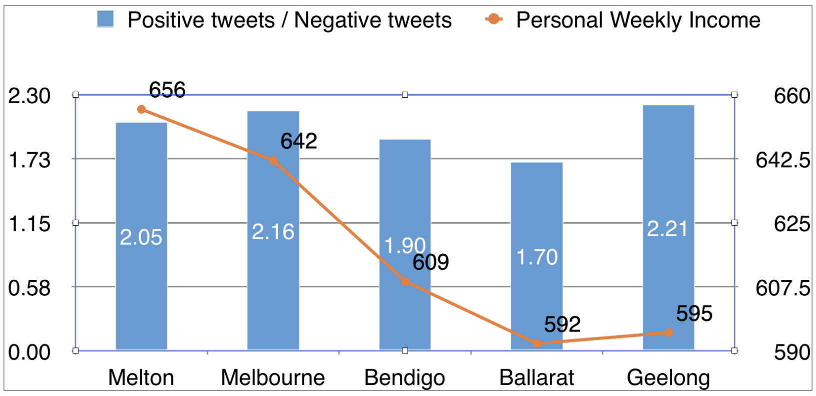

Figure. 8 positive tweets/Negative Tweest VS peronnal weekly income for cities

As shown in the chart, Melbourne and Melton have higher weekly income and positive/negative ratio, while Bendigo and Ballarat have relatively lower weekly income and positive/negative ratio. However, Geelong follows a different pattern. There may be other significant factors having impacts on Geelong's residents' happiness. In general, the trending shown from the data is consistent with the hypothesis.

Scenario 2: Are people happier (express more positive sentiment) in areas with younger population?

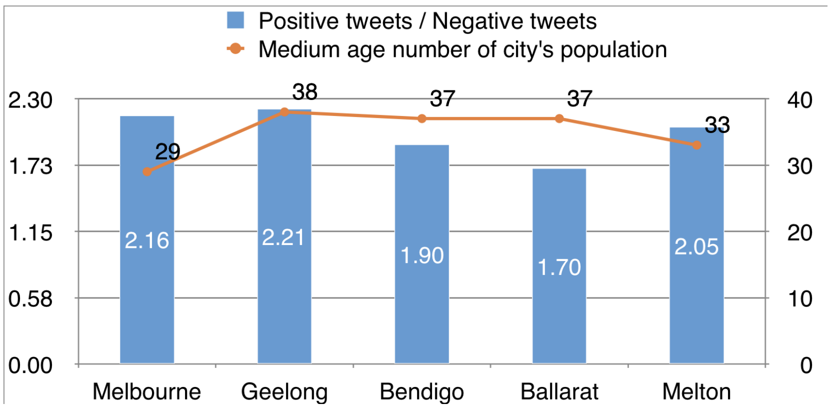

Figure. 9 positive tweets/Negative Tweest VS medium age number for cities

Melbourne and Melton as cities with lower population median age have higher positive/negative rate, while Bendigo and Ballarat as cities with relatively higher median age have lower positive/negative rate. It is interesting to find that Geelong still behaves differently from the hypothesis. The trending shows that residents living in a city with lower population median age tends to be happier.

Scenario 3: Are people more likely to send tweets in areas with higher portion of residents with higher education?

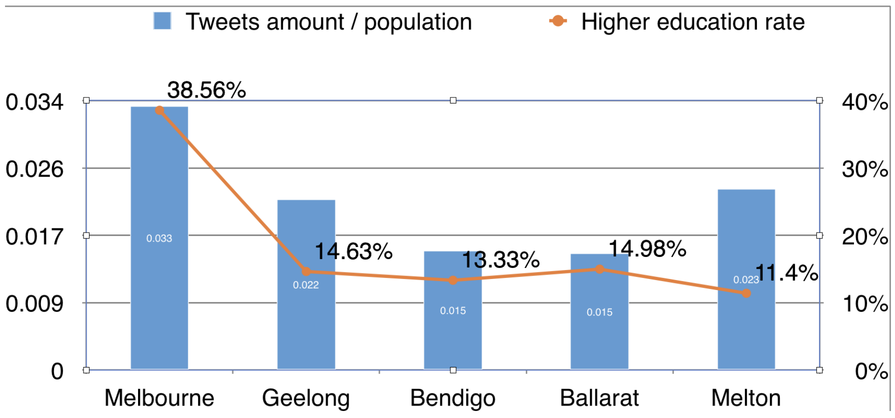

Figure. 10 Tweets amount/ population VS higher education rate for cities

(Adjustment to harvested data: To reduce the noise from the population on the number of tweets sent from areas, we will exam the relation between the ratio of total amount of tweets to population and higher education rate, instead of comparing total number of tweets and education rate directly)

As shown in the graph, people living in cities with higher education rate like Melbourne are more likely to send tweets. Compared with Melbourne, residents living in Geelong, Bendigo and Ballarat have low potential to send tweets. Only Melton is different from the hypothesis.

Therefore, the trending shown from the graph is same as what we expected from the hypothesis.

6.5.3 Problem in Scenarios:
Some factors may be correlated to other factors, which may make scenarios with similar trending, but it will be hard to figure out which factor exactly affect residents' sentiments and behaviors. For example, Scenario 1 and Scenario 2 have similar trending, both of them indicated that people with better internal or external resources would be much happier than others. Based on intuition, resident's income and age might be correlated. The hypothesis that young people will be happier may also be affected by their income. Thus, further research on the correlation among those factors might be required to improve the reliability of those Scenarios.

7. Conclusion

In conclusion, this paper demonstrated cloud-based solutions to analyse Twitter dataset and AURIN dataset to our research. It demonstrated correlations between different datasets. Based on the primary purpose of tweet data filter, harvest, and valid data storage, the system is designed to support storage of the massive amount of data and speed up the twitter data retrieval. In the system architecture, we store tweet data in a document format into the database CouchDB. Then, those data are filtered and retrieved using CouchDB MapReduce function for further analysis. And also, back up. The system also includes sentiment analysis for the various researches which utilise the tweet data stored in the CouchDB.

8. User Guide

This section will discuss how the developer can deploy the system on NeCTAR cloud and how end users can invoke our system.

8.1 System Deployment

The whole infrastructure of the system is able to be automatically deployed using out-of-box scripts provided in the source code package. In addition, simple commands can be used by the maintainer of the system who wants to start the harvester manually.

8.1.1 Automatic Deployment

The creation of the cloud virtual machines and the volumes are implemented using Boto, and the deployment of the harvester, database and the web server including their dependency packages is done using Ansible playbook. The Boto script and the Ansible playbook script are further wrapped in a shell script for easier use of the deployment. Before using the automatic deployment script, Boto and Ansible must be installed in the developer's computer. This report assumes that the computer running the deployment script runs the operation system of Ubuntu.
To use the scripts, the dependencies of the deployment tools should first be installed.

```
sudo apt install ansible
sudo pip install boto
```

Then switch to the directory of "deploy" in the source package, and edit the file "config.json". The file should look like:

```
{
  "access_key" : "XXXXXXXXXXXXXXXXXXXXXX",
  "secret_key" : "XXXXXXXXXXXXXXXXXXXXXX",
  "image" : "ami-e2d5e55e",
  "instance_type" : "m1.medium",
  "region" : "melbourne-qh2",
  "sec_group" : ["ssh", "http(s)","default","coachdb","IMCP"],
  "key_pair" : "menooker",
  "volumn_type" : "melbourne",
  "instances":[
    {"size":75,"type":"db"},
    {"size":75,"type":"db"},
    {"size":50,"type":"analysis"},
    {"size":50,"type":"web"}
  ],
"output_header":"[all:vars]\nansible_user=ubuntu\nansible_ssh_private_key_file=/home/m
enooker/.ssh/id_rsa"
}
```

This configuration file is used by the Boto script to set up the cloud virtual machines, and the meaning of the fields are self-explanatory. Note that the access key and the secret key can be obtained in the web console of NeCTAR.
Also, the configuration file for the harvester should be created in “crawler/config.json” in the source package, with its contents looking like:

```
  {
    "consumer_key" : "XXXXXXXXXXXXXX",
    "consumer_secret" : "XXXXXXXXXXXXXX",
    "access_token_key" : "XXXXXXXXXXXXXXXXXXXXX",
    "access_token_secret" : "XXXXXXXXXXXXX",
    "db_url" : "http://localhost:5984",
    "db_name" : "tweets",
    "keywords" : [],
    "locations" : "141,-39.18,148.050616,-35.932618,141,-35.932618,143.434161,-
  34.145419",
    "geocode" : "-37.3,144.2,250km"
  }
```

The fields of the configuration file for harvester are also self-explanatory. The user should find the tokens and the keys with the web API console of Twitter.

Finally, the creation of the virtual machine instances and the deployment of the system, including CouchDB, harvester and the web server can be easily achieved by invoking the script in the source package.

```
        sh deploy.sh
```

After successfully running the script, the harvester should be running on the VMs which have the type “db” in the configuration file for the Boto script (see the field “instances” of the file).

The web server will run on the instance of the type “web”. And the analytic program will run on the instance of the type “analysis”. The IP addresses of the three types of instances will be available in the file “hosts” in the current directory.

8.1.2 Manual start of the harvester and analyzer

To start the harvester manually, one should login to the harvester instances using ssh and then switch to the directory ~/DonauTwitter/crawler.
To start the harvester using Twitter Stream API and Search API, the following two commands should be used respectively:

```
    nohup python -u TwitterCrawler.py f > f.txt 2>&1  #Stream API
    nohup python -u TwitterCrawler.py s > s.txt 2>&1  #Search API
```

Note that the configuration file in the directory “~/DonauTwitter/crawler” maintains the settings for the harvester, which includes the Twitter API secret keys and tokens, the key words to search, the location of the tweets that the user is interested and the CouchDB’s URL.
To run the analyzer, login to the analytic instance and switch to the directory ~/DonauTwitter/analytics and run the command:

```
        nohup python -u TwitterAnalytic.py > log.txt 2>&1
```

8.2 End-user invocation

1. Open any web browsers (Google Chrome, Safari, Fire Fox, etc.);
2. Enter http://115.146.86.137 in the adress bar;
3. See our analysis and data visualization of the project.

9. Team Members and Roles

Yixiong Ding:

- Data analysis
- Front-end development

Yijie Mei:

- Deploying CouchDB, VMs and NeCTAR

- Twitter harvesting
- Analyzer cooperation with CouchDB

Tiange Wang:

- Twitter harvesting
- System architecture design

Wuang Shen:

- Data processing
- Data analysis

Ruifeng Luo:

- Front-end development

10. Reference

1. Cowling, D. (2018). *Social Media Statistics Australia – January 2018*. [online] Socialmedianews.com.au. Available at: https://www.socialmedianews.com.au/social-media-statistics-australia-january-2018/ [Accessed 1 May 2018].
2. Codex.wordpress.org. (2018). *Installing WordPress « WordPress Codex*. [online] Available at: https://codex.wordpress.org/Installing_WordPress#Things_to_Know_Before_Installing_WordPress [Accessed 9 May 2018].

3. Developer.twitter.com. (2018). *Standard search*. [online] Available at: https://developer.twitter.com/en/docs/tweets/search/overview/standard [Accessed 8 May 2018].
4. En.wikipedia.org. (2018). *CouchDB*. [online] Available at: https://en.wikipedia.org/wiki/CouchDB [Accessed 6 May 2018].
5. En.wikipedia.org. (2018). *MapReduce.* [online] Available at: https://en.wikipedia.org/wiki/MapReduce [Accessed 8 May 2018].
6. GitHub. (2018). *geduldig/TwitterAPI*. [online] Available at: https://github.com/geduldig/TwitterAPI [Accessed 7 May 2018].
7. Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.