**COMP90024 Cluster and Cloud Computing**

**Assignment 2**

# Australian Cities Analytics Report


**Team 32**

Ruixi Huo 815821

Yunjie Jia 864538

Yi Song 827781

Chong Liu 796337

Xiaoting Huang 786408

# Table of Contents

Ruixi Huo, Yunjie Jia, Yi Song, Chong Liu, Xiaoting Huang

# Abstract

This report presents all the works have been done by team 32 for COMP90024 Cluster and Cloud Computing Assignment 2 building an Australian Social Media Analytics system using Nectar (National eResearch Collaboration Tools and Resources) and combining datasets from AURIN to find some useful information about Australian lifestyle in multiple cities.

# 1. Introduction

Twitter is one of the most popular social media platforms. Twitter users can share their thoughts and opinions by posting a about 140 characters long post and may attach geo information onto the post if they want to, which allows analysing tweets geographically and find correlations between tweets and research datasets from various sources such as AURIN.

The system contains a multi-node tweet harvester, clustered CouchDB to store tweets and perform build-in map reduce function and a front-end website to display results. Each team member's tasks and responsibilities are shown in the table below:

| Team Member | Responsibilities |
|---|---|
| Xiaoting Huang | Tweet Harvest, Boto, Ansible |
| Chong Liu | Data Cleaning, Sentiment Analysis and Data Analysis |
| Yi Song | Map Reduce |
| Ruixi Huo | Caching development, coordination and architecture design, review |
| Yunjie Jia | Web Design |

*Table 1: Tasks and Responsibilities of Team Members*

# 2. Related Works

The feature of social media contents being huge in volume and high in veracity has made itself naturally an ideal resource for doing research from the big data perspective. Given that nowadays big data processing framework such as Hadoop and Spark being abundant and matured, majority of the studies are now focusing on the veracity feature rather than the volume feature of big data. Namely, the studies on this day and age are mostly established with very sufficient computing resources such as IaaS cloud structures or high-performance computing clusters utilizing existing big data processing framework with the research task being dealing with the heterogeneity of the data contents and retrieve meaningful patterns among different kinds of people, such practice is referred by many researchers as feature extraction. (H. Andrew Schwartz, 2013) (Gonzalez, 2015)

Machine learning and statistical methods are extensively used for feature extraction. For example, Schwartz et.al. has analysed over 700 million tokens from Facebook messages using clustering method and successfully drawn a pattern indicating significant variant of posting styles from people of different age and genders. In addition to such general-purpose feature extraction, the practice can also be done effectively for some specific fields (H. Andrew Schwartz, 2013). A research done by Muhammad et.al. has yielded over 80% accuracy in classifying tweet contents to be a "disaster related event" (Muhammad Imran) and Azadeh et.al. even succeeded in labelling adverse drug reaction from pre-processed, tweet contents, which are all surprising results as social media contents are originally considered to be more related to recreational topics.

In Australia, a dedicated group called AURIN (THE AURIN JOURNEY, 2010) has been continuously working on social media feature extraction study and practices and there are substantial research results on Australian society related topics such as average income by age/suburbs/occupations etc. Such data will be used for comparing purposes for the teams' study on performing simple feature extraction practices over cloud resources.

# 3. System Design

## 3.1 Resource Utilization

For supporting the analytical project, the team is assigned with the capacity to use up to 8 computing cores and a maximum of 250GB storage volume by setting up virtual machines over a research-purposed IaaS cloud called Nectar. The team will allocate the 8 processors evenly over 4 independent virtual machines, each of which is attached with a 50GB volume and will collaboratively fulfil the task of harvesting, querying, visualization and the backup of the tweet data.

## 3.2 System architecture

Major purpose of the target system is to provide visualized data on certain topics extracted from social network service contents (i.e. twitters). Therefore, we can naturally divide the system into 3 components, respectively:

- **Main CouchDB service:** A NoSQL database supporting parallelized accumulative analysis using MapReduce. The technology is taken considering the big data nature of the desired system for its feature of being volume-intensive and less structured and MapReduce and NoSQL are proven technologies for data processing on this regard.

- **Twitter Streaming process:** Multiple backend tweet streaming process harvesting tweets from official tweet API. The system will use a 2 instance of the harvesting process running on the background to pull tweet contents with meta-data from twitter database to local CouchDB database. Each thread will use its own geofilters so that the tweets crawled from each harvester is separated in locations. To avoid duplicate tweets being streamed, the harvester process will make use of CouchDB's duplicate removal feature in setting the CouchDB record ID to be the same as twitter ID.

- **HTTP Server:** Django is a high-level Python web framework that supports rapid development and clean, pragmatic design. A http-based web server is built to provide visualizing services for end users to get summarized knowledge through an intuitive graphical user interface, which uses Google Map APIs to represent data geologically.

## 3.3 Auxiliary Components

To secure the high durability and maintainability of the system, the following components are added as supportive modules with backup/restore and caching purposes:

- **Subordinate CouchDB service:** For the purpose of data backup and avoiding the constant communication between instances. Each instance running harvesters processes will have a local CouchDB hosted where tweets streamed from the local harvesting process will store. The subordinate database will only push the increment of data periodically to the main CouchDB database. Those subordinate services will essentially be a mirror of the major CouchDB instance and by another scheduled job pulling from the major instance.

- **Scheduled backup process:** The CouchDB backup is archived by a scheduled task sending request to the secondary CouchDB of importing data from primary database triggered regularly by Linux Cron.

- **Scheduled caching process:** With the given amount of data being over tens of Gigabytes in volume and the limited computing resources (2 cores on the instance

hosting primary CouchDB), it is not a wise choice to provide ad-hoc queries when website users request visualized data. As such, the system will instead have a scheduled job (controlled using Python's built in functionality of having the process sleeping for a certain time after execution) requesting the view of CouchDB regularly and cache the result into a file ready to be consumed by the web service to ensure the visualized data can be done within reasonable time frame.

- **RESTful API for data visualization:** The scheduled query process will cache the CouchDB query results into a JSON file. Interface to those result file from external domains will be given by a restful API (written in python flask) running in the background which will allow GET request from external resources for each cached view. Cross origin request is allowed for this RESTful service for consuming the service using JavaScript.

## 3.4 Role Allocation

The overall architecture gathering the mentioned components is illustrated by Figure 1 where 4 virtual instances are assigned with tasks logically according to the rule of the instance. As implied by the figure, 4 instances will be respectively responsible for 3 kind of tasks, respectively:

- Querying and caching – The instance running the main CouchDB service will be in charge of handling view requests and performing Hadoop queries. The mentioned caching process will periodically call the main CouchDB service hosted locally and cache the result into JSON files whose interface is provided by the RESTful API.

- Harvesting – Two instances will be running the harvesting API storing data in the local CouchDB services. The local CouchDB services will periodically run push/pull tasks between the main CouchDB instance and essentially mirroring each other as mutual replications.

- Visualizing – The last instance will be responsible for consuming the data API and provide visualize data to end-users over a website.

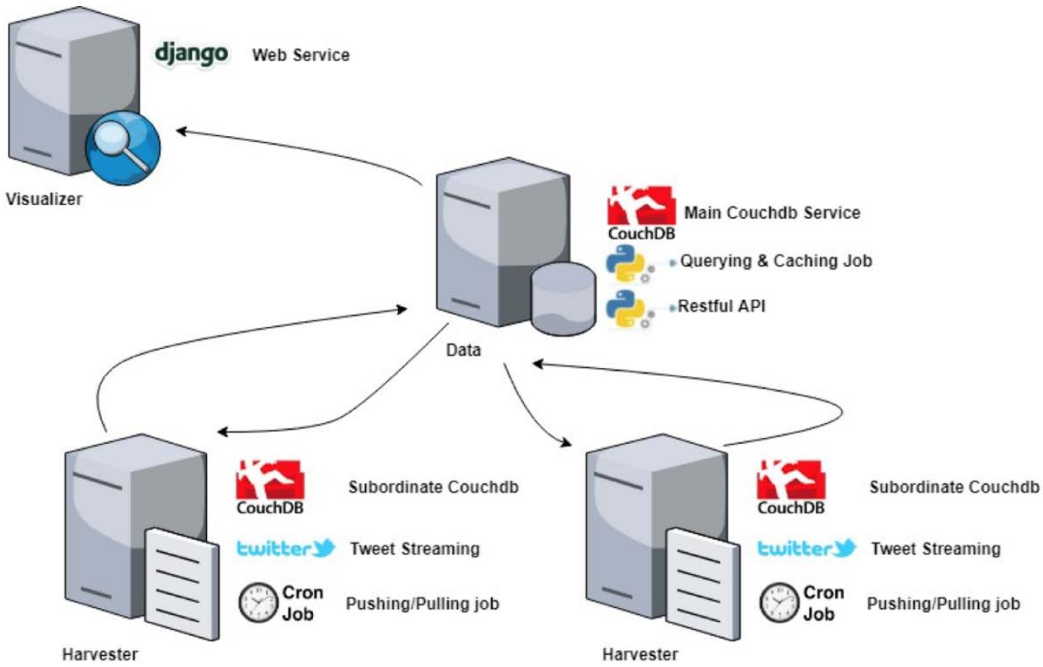


*Figure 1: Overall system architecture*

## 3.5 Security and Fault Tolerance Discussion

The system security relies significantly on the built-in security feature of the nectar service. All instances are allocated with login security groups allowing minimal inbound packets to instances. The following rules are added in addition to the default configuration of the login security group to loosen the network constraint for providing necessary accesses:

- Inbound – Allow – Port 5984 from (Instance 2 Address): For allowing access to CouchDB service (for replication/backup purpose only)
- Inbound – Allow – Port 5002: For allowing access to restful API.

These two extra configurations will bring very limited security concern to the whole system as access from the CouchDB is only provided to another backup instance for replication tasks and cross origin requests is still disabled in CouchDB (as per default CouchDB configuration). The updating and view querying will only be done locally (the caching service sit on the same instance) and the restful API accessible from port 5002 only allow GET operations, meaning no operations on cached data other than viewing is allowed. Therefore, the whole system can be considered as robust from the security perspective.

When comes to fault tolerance, the system does have replications on very essential components (i.e. the CouchDB data). When comes to other module such as tweet harvesting process and the caching process, the system will have to rely on the modules' code quality in

terms of having a robust try-catch-log-retry procedure to minimize the single point of failure concern as each tweet harvesting process does not replicate each other as they are configured to crawl tweets from independent locations and there is only one RESTful API instance, which put strict requirement on code qualities. Fortunately, the system's logging process as coded can provide sufficient information for developers to investigate what went wrong.

## 3.6 System Deployment and Scalability Discussion

From the functional perspective, the system does not use any feature specific to Nectar clouds. Namely, the whole system can be easily migrated to other cloud infrastructures such as Amazon EC2 or other open stack-based cloud services. An automatic deployment scripts (Detailed in **Section 3.7**) can be used to establish virtual machines with any configurable computing resources (rather than 4 cores on 2 nodes) and install system components all at once. According to the system design its foreseeable that the impact of using different number of cores on virtual machines is minimal and the effectiveness of adding more cores is deemed to be high for the following reason:

CouchDB and Tweet harvesting process are two major components of the entire system where the background technology of CouchDB is MapReduce having the nature of being able to work independently among mapper and reducer threads while the tweet harvester threads will establish separate connections to the tweet server for the streaming.

The only identified side effect of establishing more tweet streaming process is that if two processes have close geolocation filters, the chance of duplicate tweet being harvested will go relatively high and the ratio of the "useful" tweets per thread will drop. It is therefore somewhat important to have a proper region allocation to reduce the chance of the above scenario.

## 3.7 Deployment Automation Detail

Ansible is used as the management tool to automate the creation and deployment of virtual machines and set up software environment. A server with Ansible client properly installed and access to remote servers under management with SSH connections are the minimal requirements for using Ansible.

The configuration process of the existing system can be replicated using the playbook *config_whole_sys.yml* with command:

```
ansible-playbook -i ~/Ansible/inventory/hosts.ini -u ubuntu
config_whole_sys.yml
```

within the same directory as the playbook file. Three types of servers are managed: database server, harvester server, and web server. The host IP addresses are grouped in a *hosts.ini* file in order to support different configurations with a unique key file to each of the hosts as the following format

```
[dbserver]
115.146.95.234 ansible_ssh_private_key_file=~/Ansible/keys/cloud.key

[harvesterserver]
115.146.95.66 ansible_ssh_private_key_file=~/Ansible/keys/nectarkey.pem
115.146.95.115
ansible_ssh_private_key_file=~/Ansible/keys/nectarkey.pem

[webserver]
115.146.95.226 ansible_ssh_private_key_file=~/Ansible/keys/huor2.ppk
```

All servers share some common requirements. Therefore, a common role is played on all server to install common packages. Then, specific roles are played on designated servers for particular needs. Tasks in the common role includes updating apt cache, installing essential packages, such as python3, python-pip, python3-pip, and git, transferring the SSH key for GitHub access to the remote hosts with required permission level and cloning working repository to ensure the harvester application is available upon correct configuration.

After that, three types of roles will be played on different group of hosts. *config-dbserver*, *config-harvesterserver*, and *config-webserver* roles will be performed in sequence on *dbserver*, *harvesterserver*, and *webserver* group of hosts respectively. On *dbserver* group, CouchDB is installed and configured as per official documentation. Some other python libraries are installed on *webserver* to provide visualization functions on web application. For *harvesterserver*, CouchDB is installed with several python libraries to support harvester functionalities. Django is installed as well to support web application for visualizing analysis results.

One key benefit of using Ansible is to simplify scaling procedure by automatically prepare the virtual machines for program launching. To further streamline the process, boto python library is employed to create new instances and attach of volume to the newly created instance. New instances can be created with *create_instance.py* using command:

```
python3 create_instance.py
```

An SSH connection will first be established with Nectar Research Cloud using the specified key pair. New instances will be launched in *Melbourne-np*, which coincide with other instances in the system. The instances will boot from image *Nectar Ubuntu 16.04 LTS (Xenial) amd64* and be allocated 2 cores. The instance will be secured and will allow access with an SSH key pair through port 22. A 30-GB volume will be created in *Melbourne-np* for attachment when the new instance is ready. New instances can take anywhere from a few seconds to a couple of minutes to setup, so the program is featured a *wait* function to check for the state of new instance every 15 seconds for up to 5 minutes until it is running, so that the volume can be attached to the instance and the IP address can be obtained. The IP address of the new instance will be written to *newhosts.ini* file and grouped under *harvesterserver* hosts. Automatic configuration of the new server can be performed with playbook *config_harvester.yml* with command:

```
ansible-playbook -i ~/Ansible/inventory/newhosts.ini -u ubuntu --key-
file=~/Ansible/keys/nectarkey.pem config_harvester.yml
```

It will have all packages installed and the harvester application ready on the server to run. Figure 2 is an example of configuring a new *harvesterserver*.

```
PLAY ***************************************************************************

TASK [setup] *******************************************************************
The authenticity of host '115.146.95.115 (115.146.95.115)' can't be established.
ECDSA key fingerprint is SHA256:w/TO+OLpz3m19C1GPmNnYkqNQAIQFV+AGUUZ2L/COWs.
Are you sure you want to continue connecting (yes/no)? yes
ok: [115.146.95.115]

TASK [common : update packages] ************************************************
ok: [115.146.95.115]

TASK [common : install required packages] **************************************
changed: [115.146.95.115] => (item=[u'python-pip', u'git', u'python3', u'python3
-pip'])

TASK [common : pass git key to server] *****************************************
changed: [115.146.95.115]

TASK [common : clone repo] *****************************************************
changed: [115.146.95.115]

TASK [config-harvesterserver : install required packages] **********************
changed: [115.146.95.115] => (item=[u'couchdb'])

TASK [config-harvesterserver : install dependencies] ***************************
changed: [115.146.95.115] => (item=tweepy)

TASK [config-harvesterserver : install dependencies] ***************************
changed: [115.146.95.115] => (item=numpy)
changed: [115.146.95.115] => (item=couchdb)
changed: [115.146.95.115] => (item=TextBlob)
changed: [115.146.95.115] => (item=Tensorflow)

PLAY RECAP *********************************************************************
115.146.95.115             : ok=8    changed=6    unreachable=0    failed=0
```

*Figure 2: Successful configuration of harvesterserver with Ansible*

The combination of Ansible and boto can accelerate on-demand scaling out processes with two simple command lines with one creating a new instance and the other set up the software environment. *config_harvester.yml* playbook and the *create_instance.py* program are designed to facilitate needs in speeding up the tweets collection with twitter harvester. Multiple harvester servers can be launched to collect more data.

# 4. Twitter Harvester

Tweets are the main input for analysis. As a large volume of tweets is needed, two instances are launched to gather tweets. The harvesters collect Tweets from all Australia using both Stream API and Search API made available by Twitter.

The harvester application can be run using command:

```
nohup python ~/ COMP90024-Assignment2/Harvester/ tweet_harvester.py
```

It allows the harvester to run in the background and detach from terminal. *tweet* database and *user* database are created in CouchDB to store tweets and searched users. A bounding box of [113.338953078, -43.6345972634, 153.569469029, -10.6681857235] retrieved from Natural Earth (Cultural Vectors, 2018)is used to define the coordinate boundaries of Australia. The bounding box is divided equally into two sub bounding boxes. Each harvester is harvesting tweets from one sub bounding box. Tweepy python library is utilized to connect to the Twitter APIs. The harvester starts with Stream API searching for tweets within the boundaries. Tweet ID of each tweet is a unique identifier and is used as the primary key to the tweet database. ID is extracted from all incoming tweets and compared against the existing tweets in the database to avoid duplicates. The entire tweet is saved under its tweet ID for further processing to support changes in requirement. For example, *media* attribute of tweets is not used in the current solution. However, if *media* attribute is needed in other analyses, storing processed tweets instead of raw tweets will cause inconsistency and all store data will be wasted.

For each incoming tweet, *screen name* attribute is also extracted. If the user is not in the user database, Search API is responsible for getting all tweets from the user with the screen name. Only new tweets with a coordinate inside or on the boundary box are saved. Then, the user is added into the user database with user ID being the primary key. Due to the rate limit of Search API, the API is set to wait for rate limit to avoid errors.

Because of network instability and difference in the speed of processing between the harvester application and APIs, exceptions may be raised during the harvesting processes. They are properly handled through reconnection and abandon any incomplete tweets.

# 5. Data Analysis

Data analysis can be divided into the following three steps: data cleaning, sentiment analysis and data analysis.

## 5.1 Data Cleaning

After tweets have been collected using Twitter API, they need to be transformed to a cleaner format by removing useless information contained in the twitter metadata to reduce storage

space required and speed up processing speed. After careful consideration, the following information has been selected to keep for analysing purpose.

```
doc = {
        'place_name' : place_name,
        'timestampe' : timestampe,
        'day' : day,
        'time': time,
        'text' : text,
        'length': length,
        'bounding_box': box,
        'sentiment' : sentiment,
        'coordinates' : coordinates,
        'userid' : userid,
        'hashtags': hashtags,
        'user_location': user_location
}
```

*Figure 3: Cleaning Tweets*

Place_name , coordinates, bounding_box and user_location are used to identity where the tweet was posted and Google map plot; day and time are used to summarise how many tweets were posted on a particular day of the week and within an hour of a day in 24 hours format; userid and timestamp are used for tracking users' location history and whether the user is happy or unhappy over a period of time; hashtags are used to find what does people care about in different places and text is used for sentiment analysis.

## 5.2 Sentiment Analysis

One effective way to analyse text based social media posts is to perform sentiment analysis also known as opinion mining to get the author's opinion over a particular topic. These sentiments can be combined with geographical information and data from AURIN to find correlation between the sentiments on social media over a particular research field within a certain area. This requires only collecting tweets which contains geographical information. Tweets collected using twitter search API will have full text but tweets collected from twitter stream API only contains part of the textual context. For example, 'text': "I'm at Curtin

Student Guild in @trisreed https://t.co/rTjeeyeo6P". This increased the difficulty of sentiment analysis.

TextBlob is used to perform sentiment analysis over collected tweets. Textblob is used for simplified text processing. It works with both text and emoticons. TextBlob can classify a piece of text sentiment into positive, negative or neutral ranging from [-1, 1]. TextBlob's sentiment analysis accuracy is tested over sample_twitter corpus' 5,000 manually classified positive and negative tweets from NLTK library. The accuracy is 94.78% for negative tweets and 98.12% for positive tweets. However, TextBlob does not work with emojis such as: 😆, 😭, 🤓 and so on.

```
>>> import textblob
>>> from textblob import TextBlob
>>> TextBlob(':-)').polarity
0.5
>>> TextBlob(':-(').polarity
-0.75
```

*Figure 4: TextBlob over Emoticon*

If higher sentiment analysis accuracy is required, more sophisticated sentiment analysis methods such as TensorFlow can be used.

## 5.3 Data Analysis

The data is analysed at three different levels: national level, city level and suburb level. In order to visualize the data, table, bar-chart and map is used in the web site. The basic visualization process like below:

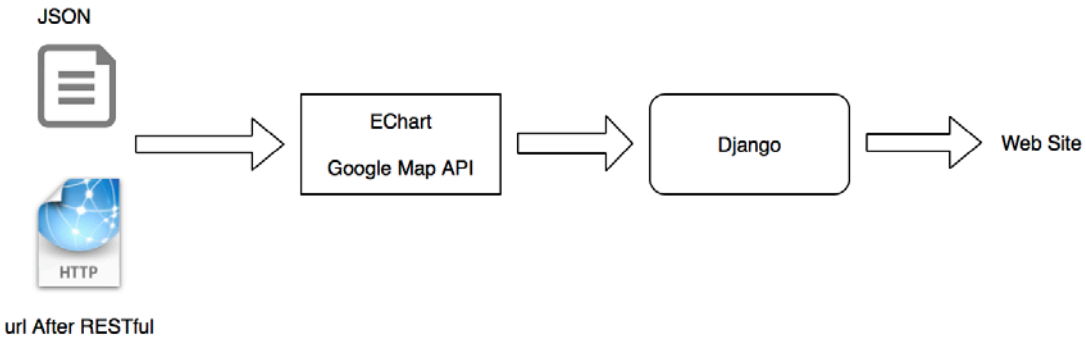


*Figure 5: Visualization Process*

The data in AURIN is stored as shapefile and converted into JSON file and the MapReduce data of tweet is stored as URL.

## 5.3.1 National Level Analysis

National level analysis summaries how many tweets were posted on each day of the week (Monday - Sunday) and how many tweets were posted at different time of the day on a 24 hours scale. Also, the most common hashtag, the most common average length of tweet and the most common region who sends the most tweet are analysed. Twitter as a global social media platform, it uses Greenwich time to timestamp events. Therefore, the time should plus 10 hours to convert to Australian Eastern Standard Time.

### 5.3.1.1 Hour-based

The 24-hour based tweet is summarized as below:

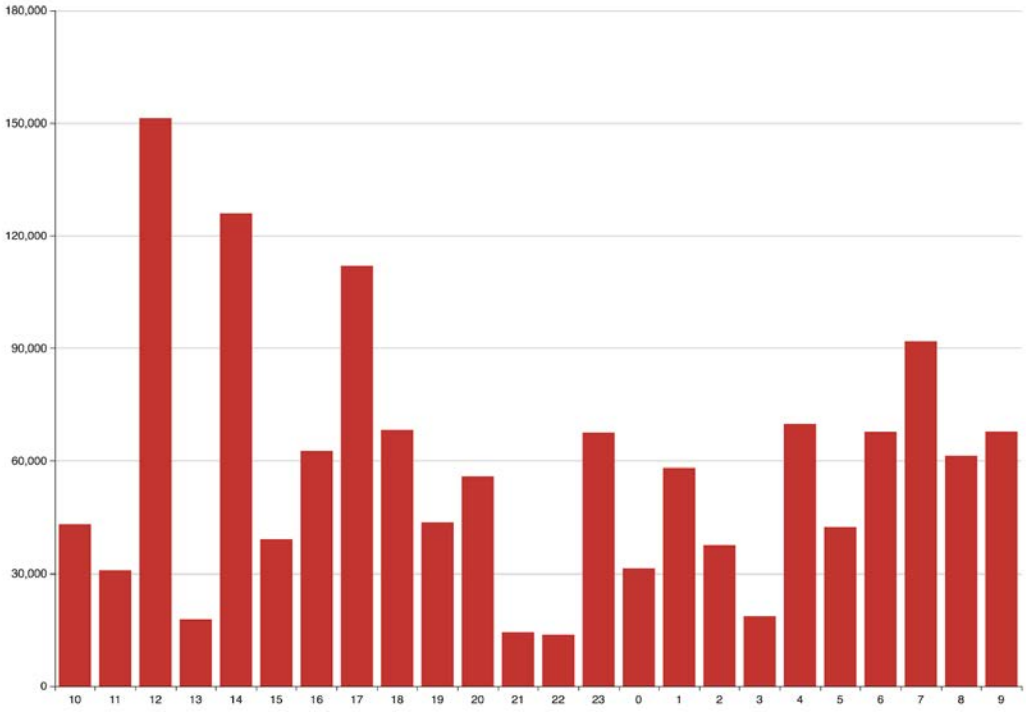


*Figure 6: Hour sum Table*

From the above figure, which shows that 12pm is the most popular time to send tweet. The reason maybe that 12pm is the noon break time, after long time work, people tend to send tweet to relax and describe their mood to their friend. 12pm is much lower, after lunch, people may need to have a rest and prepare for next work or study. The same reason for time 21-22 pm. Also, from the time period between 0am and 7am, it can be seen that still lots of people send tweet message, which is not a good phenomenon. Mobile phone makes people sleep less in the night, it is harmful for health and it also not good for next day's work or study.

#### 5.3.1.2 Day-based

Also, a week-based tweet information like below:

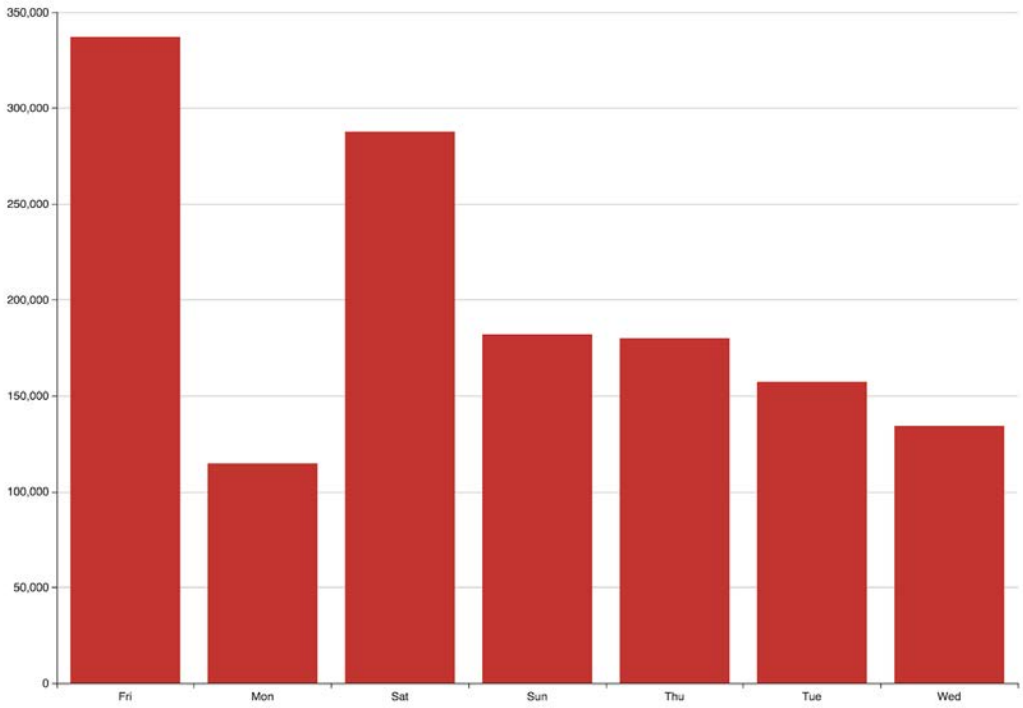


*Figure 7: Day sum Table*

People tend to send most tweet in Friday, the reason might be that Friday is the start of weekend. After a long time of work or study, people would like to have a relax and share their moments with their friends. The same reason for Saturday. As to Sunday, people will have to prepare for next work or study period. This is also applicable to remaining workday except for Monday. Monday is the first day of workday, people are down in spirits, thus, Tweet in Monday is less than in other days. These data obey the basic lifestyle of people.

#### 5.3.1.3 Region-based

| Location | Count |
|---|---|
| Sydney, Australia | 73875 |
| Australia | 60900 |
| Melbourne | 42180 |
| Sydney | 39576 |
| Sydney, New South Wales | 35022 |
| Melbourne, Australia | 34824 |
| Brisbane, Australia | 33516 |
| Melbourne, Victoria | 25089 |
| Brisbane, Queensland | 20676 |
| Perth, Australia | 20124 |

*Figure 8: Top 10 region who sends most tweet*

This region ranking obeys the population distribution of Australia. More people tends to more tweet sent [5]. Sydney has most population than Melbourne, thus, the count of tweet in Sydney is much larger than in other region of Melbourne.

#### 5.3.1.4 Length-based

| Average Length Of Tweet | Count |
|---|---|
| 99 | 75612 |
| 101 | 71886 |
| 96 | 66582 |
| 121 | 45735 |
| 140 | 42624 |
| 97 | 40890 |
| 98 | 40104 |
| 100 | 40101 |
| 93 | 39438 |
| 102 | 38526 |

*Figure 9: Top 10 average length of tweet*

The above table shows the top 10 average length of tweet, which demonstrates that people tend to send tweet with around 99-140 words, which is night too long nor too short for most people. The range of words count is suitable for describing their moods, their life and etc.

### 5.3.2 City Level Analysis

City level analysis compares difference between major Australian cities. What are the most common hashtags to find out what does people care about.

| Hash Tag | Count |
|---|---|
| starwars | 27054 |
| Repost | 24699 |
| Brisbane | 24426 |
| photo | 21576 |
| Perth | 15228 |
| sydney | 13515 |
| cat | 13137 |
| fleaman | 13131 |
| furbabies | 13131 |
| kaiser | 13131 |

*Figure 10: Top 10 of hashtag*

The above table shows the top 10 most common hashtag in tweet. From it, it shows that the most common topic is Star Wars, that maybe caused by Star Wars in Concert recently in Australia. For remaining hashtag, like photo and cat. With the development of mobile phone, people like to take photos and share their moments of life. As to cat, lovely cat is the most popular pet in the world.

### 5.3.3 Suburb Level Analysis

Suburb level analysis summaries how many tweets posted inside the suburb, what is the average sentiment for that suburb and this can be compared against data from AURIN to find correlations. For this part, AURIN data is used as cooperation. The Melbourne is divided into different suburb and paint different colour for each region according to the income level within it. It has 3 different income level, namely, above $4000, $3500-$3999 and $2500-$2900. The figures show as below:

The first line of display box is the number of person belong to the specific income level. Total count is the number of tweet in the suburb. The higher average sentiments mean people feel more happiness in particular region.

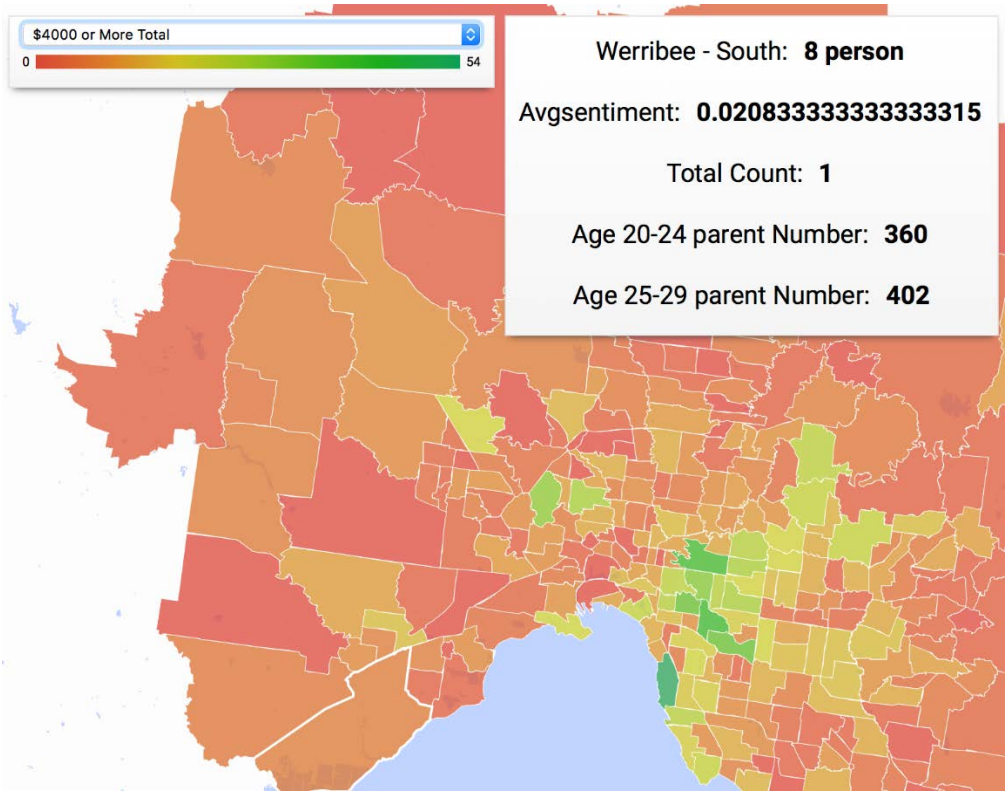


*Figure 11: Werribee - South*

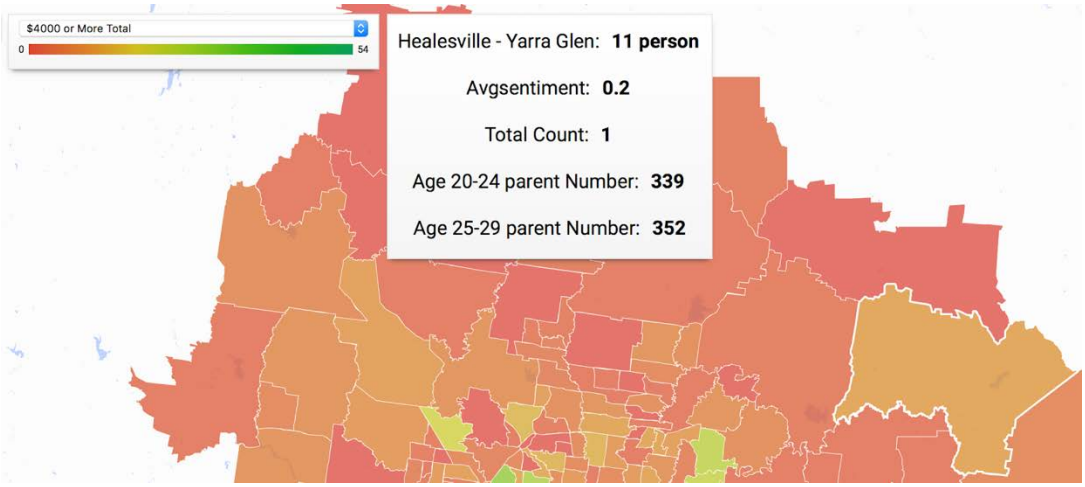


*Figure 12: Healesville - Yarra Glen*

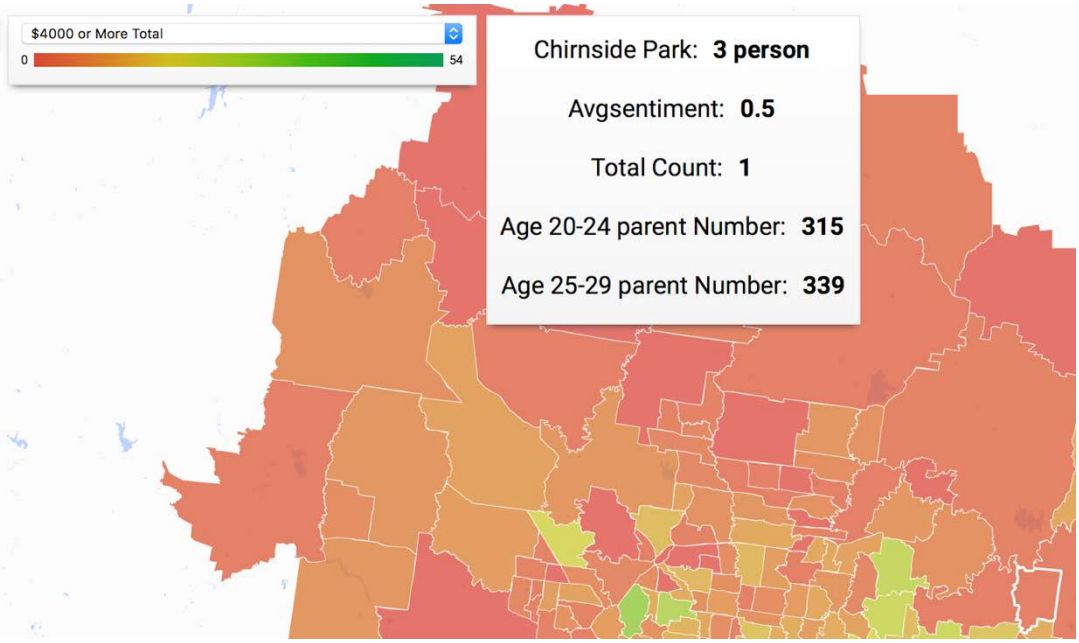


*Figure 13: Chirnside park*

From the figure 11, figure 12 and figure 13, the average sentiment become higher with less parent number between age 20-29 for same income level. The reason for that maybe caused by young couple is easy to start conflict, which will make average sentiment down. Also, from the map data above, the high income is not a determinant for average sentiment, in another word, high income may not lead to a high average sentiment value. Income is not the only main factor to make people feel happiness. For instance, in region Werribee to South, the number of person with income over $4000 is 8, which is much larger than it in Chirnside Park. However, the average sentiments in Chirnside Park is much larger than it in Werribee to South.

# 6. External Links

GitHub repository link: https://github.com/MujiLeica/COMP90024-Assignment2

YouTube video link: https://youtu.be/wGBnFvYN3KY

# 7. Conclusion and Future Improvement

As a summary, this project is built for streaming processing to harvest tweets of Australia and analyse the data on the fly. Therefore, the system has to be highly available and always online. To achieve this, replication is applied at multiple places to increase availability of the system and guarantee the system is high fault tolerant. For example, rather than a single node for CouchDB, a cluster of CouchDB is used to synchronize data between different nodes. It does not matter whether a local CouchDB crashes or the central CouchDB goes down, as

long as they do not crash together, the service is still available. However, the robustness is maintained by inefficiency in memory, disk space, bandwidth and many other resources.

Even though we tried to make this system as stable as possible, but there are still several points that can fail and may cause a system failure. For example, separated harvesters are used for different areas to increase the speed of streaming, but we might lose data from one area because the harvester for that one crashes. And what is worse is that the use of twitter API is restricted in a certain period of time based on per user, which forces us to switch accounts to have a seamless streaming process. A scheduler is needed to manage all those things in the future. Our current analysis of the data is quite simple and straightforward, and CouchDB is not designed to handle enterprise data and perform sophisticated analysis. In terms of big data processing, Spark is known for its high availability and fault tolerance and it has MapReduce functionality natively. The introduction of Spark to this project can bring it to next level.

# Bibliography

Cultural Vectors. (2018). Retrieved from Nature Earth: http://www.naturalearthdata.com/downloads/110m-cultural-vectors/

Gonzalez, A. N. (2015). Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, 671–681. Retrieved from OXFORD ACADEMIC: https://academic.oup.com/jamia/article/22/3/671/776531

H. Andrew Schwartz, J. C. (2013, September 25). *Personality, Gender, and Age in the Language of Social Media: The Open-Vocabulary Approach*. Retrieved from PLOS ONE: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0073791

Muhammad Imran, S. E. (n.d.). *Practical Extraction of Disaster-Relevant Information from Social Media.* Retrieved from citeseerx: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.3555&rep=rep1&type=pdf

*THE AURIN JOURNEY*. (2010, June). Retrieved from AURIN: https://aurin.org.au/about/the-aurin-journey/