COMP90024 Cluster and Cloud Computing Assignment 2

2021 Semester 1 Team 35

Team Members:

Jiacheng Ye	904973	Shanghai, China
Shiyi Xu	780801	Melbourne, Australia
Yuyao Ma	1111182	Yinchuan, China
Yujing Guan	1011792	Fuzhou, China
Zexin Yu	10328021	Dalian, China

YouTube Video Links:

- Scenarios and Web app Demo: https://youtu.be/ekXI2uaiMQM
- Ansible Demo: https://youtu.be/KPVLuHOlf5Q

Github Link:
https://github.com/maxpoi/Oz-Twitter-Analysis

1 Introduction

Today's era can be said to be the era of big data. Countless data are spread on the Internet every day. How to get valuable content from these data and analyze them to get valuable results is an interesting and challenging task.

Our team exploit a multitude of virtual machines (VMs) across the UniMelb Research Cloud for harvesting tweets through the Twitter APIs, use CouchDB as the Database and the kinds of data analytics (e.g. MapReduce) that CouchDB supports as well as data from the Australian Urban Research Infrastructure Network (AURIN - https://portal.aurin.org.au). The key point of this project is to harvest tweets from across the cities of Australia on the UniMelb Research Cloud and undertake a variety of social media data analytics scenarios that tell interesting stories of life in Australian cities and importantly how the Twitter data can be used alongside/compared with/augment the data available within the AURIN platform to improve our knowledge of life in the cities of Australia. Therefore, four interesting scenarios are designed to show the interesting facts of several major cities in Australia. In addition, the technology stack as well as main problems occurred in the implementation of the project are also discussed.

2 System Design and Architecture

2.1 System Design Diagram

Our team deployed system on Unimelb Research Cloud, overall architecture design is shown below in Figure 1.

In our system design diagram, our developers use Ansible to create 3 instances and allocate the resource, after setting all cloud services, we use docker for Ubuntu to deploy Cluster CouchDB, front-end and back-end. In terms of the crawler for data, it has two sources from Twitter API and AURIN and processed or analyzed with sentiment tool, then post database and put data on the CouchDB with MapReduce. The back-end server create Restful API and adds resource to return HTML according to the URI the client requests. Above all is the overall design of our system.

The description of reason that we choose these tools is list in the rest of section 2.

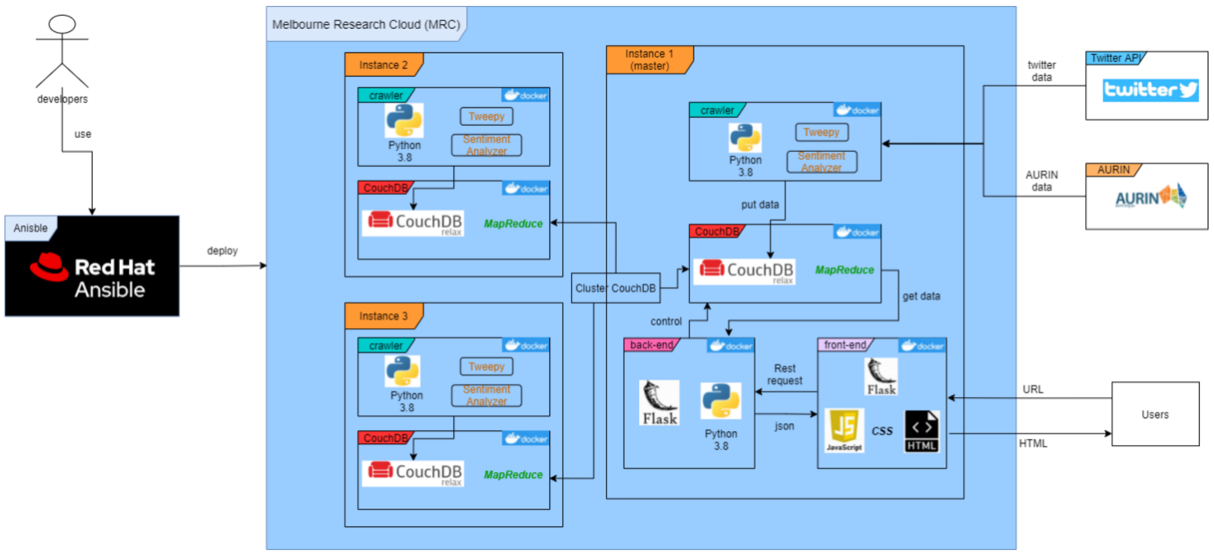


Figure 1 System Design Diagram

2.2 Global management

2.2.1 CM Tool - Ansible

There are many moving parts to deploy a complex cloud system. If blindly implemented, it is easy to make mess with the existing software or tools and developers do not know which step they are in during the configuration of system. Besides, configuring environment manually without changed record may lead to redundancy on operations, and some requirements are hash which may cause errors or re-do work during the deployment.

To solve this problem, configuration management (CM) tools is a great choice to standardize the deployment process instead of manual operation. The CM tool realizes the concept of automation. Automation is a mechanism that was previously used to define provisioning scripts using tool-specific languages and functions to bring the server to an ideal state.

We choose Ansible tool (use YAML to express playbooks) because its features:

- YAML, Jinja2 and sequential execution is easy to learn.
- A simple single command is enough to install Ansible and centralized management servers and daemons does not need.
- Ansible can makes process repeatable and prevent side-effects from re-running scripts.
- Ansible support various custom modules, push or pull, rolling updates for continuous deployment, management for a dynamic list of external data sources.

YAML is a human friendly data serialization standard for all programming languages. Jinja2 is a template suitable for Python language, based on Django's template.

2.2.2 Docker

- Docker is currently the leading software container platform. Docker is used because of five advantages:
- Continuous Integration: Docker can maintain consistency across environments. In the life cycle of development and release, different environments have subtle differences. These differences may be caused by the versions and dependencies of different installation packages. However, Docker can solve this problem by ensuring the consistency of the environment from development to product release. Besides, you can easily put the things that need to be changed into the Docker container, test them, and make your existing container perform the same changes.

- Version Control: Docker containers can be like git repositories, allowing you to submit changes to the Docker image and manage them through different versions. Imagine that if the entire environment is damaged due to an upgrade of a component, Docker can roll back to the previous version of the image, which allows you to quickly replicate and achieve redundancy.
- Portability: All major cloud computing providers, including Amazon AWS and Google's GCP, have integrated Docker into their platforms and added their own support.
- Isolation: Docker also ensures that each application uses only the resources allocated to it (including CPU, memory, and disk space). A special software will not use all of your available resources, otherwise it will cause performance degradation or even stop other applications from working completely.
- Security: From a security perspective, Docker ensures that applications running in containers are completely separated and isolated from applications in other containers, and it also has complete control over communication traffic and management. Docker containers cannot view processes running in other containers. Each container uses only its own resources.

2.3 Melbourne Research Cloud

Cloud computing is a model that allows universal, convenient, and on-demand network access to shared configurable computing resource pools (such as networks, servers, storage, applications, and services), which can be quickly configured with minimal management effort or service provider interaction and release (Mell & Grance, 2011).

The various flavours of cloud computing have been developed, the most common delivery cloud models include Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS).

MRC (Melbourne Research Cloud) which based on openstack belongs to Iaas and the managed responsibilities include applications, data, runtime, middleware, O/S.

2.3.1 Resource Allocation

We are allocated with 4 instances (servers) with 8 vCPUs (32GB memory total) on the MRC. We set up 3 instances with 9GB for each. The Usage condition of MRC is shown in Figure 2.

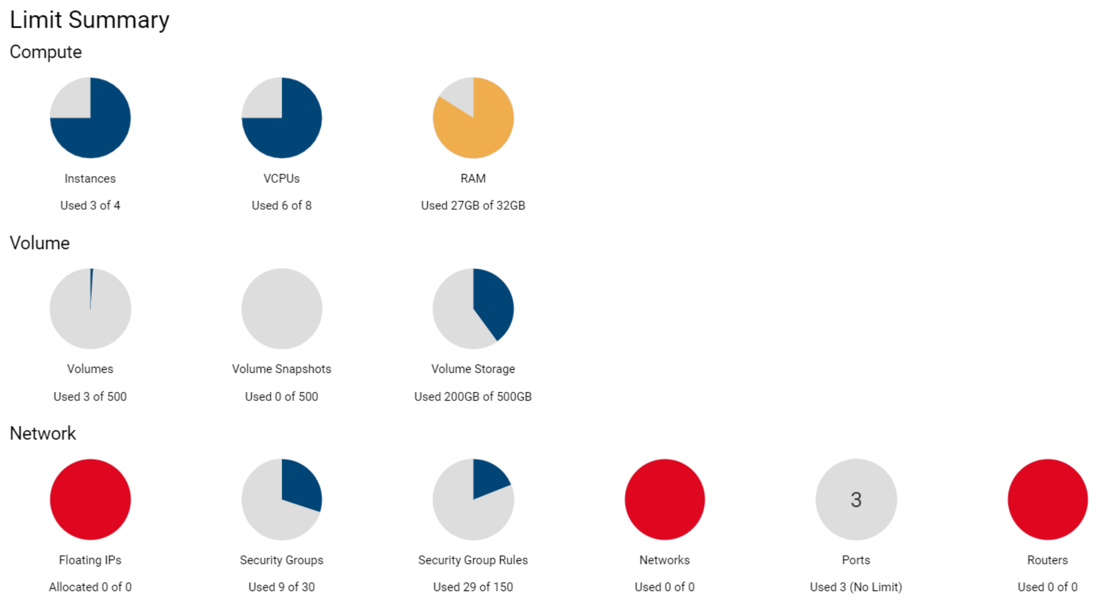


Figure 2 The Usage condition of MRC

2.3.2 Advantages

There are many reasons why someone would see cloud infrastructure as a potential fit:

2.3.2.1 Costs

Unlike traditional IT such as the AWS and AZURE, Iaas does not require any upfront, capital expenditures, and end users are only billed for what they use. There is also no hardware cost. For MRC, the usage of the cloud services is free. We can see the cost comparison of various cloud models which is shown in Figure 3.

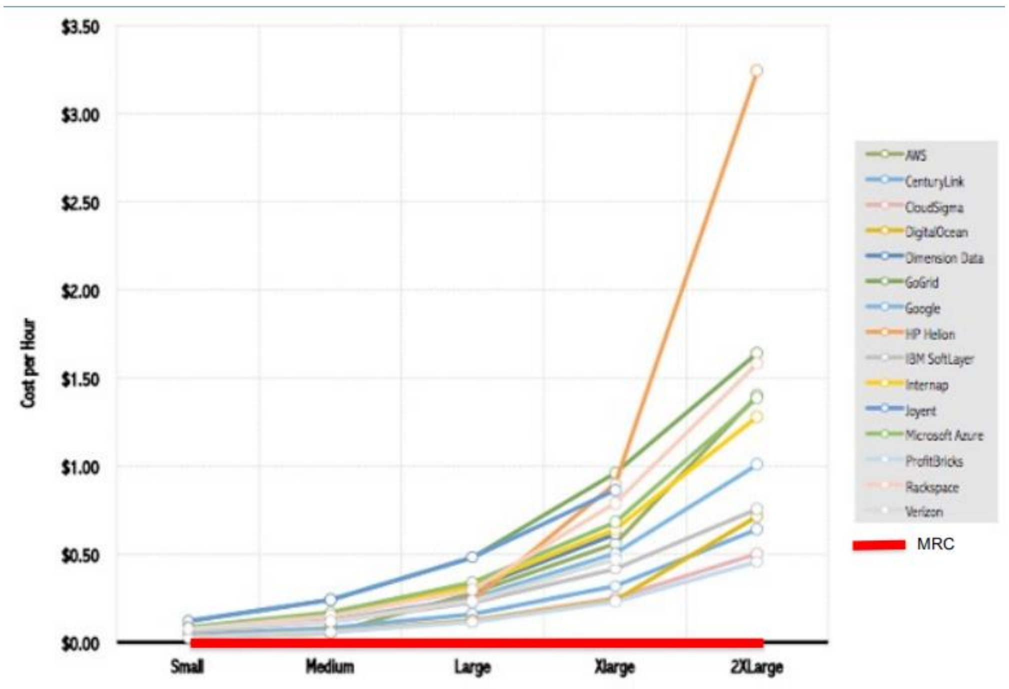


Figure 3 Cost Comparison of Cloud models

2.3.2.2 Speed

With MRC, users can provision small or vast amounts of resources in a matter of minutes, testing new ideas quickly or scaling proven ones even quicker. Because the University of Melbourne has its own data center in which cloud devices are located, it can achieve high-speed transmission between instances.

2.3.2.3 Availability

Through things like multizone regions, the availability and resiliency of cloud applications can exceed traditional approaches. Everyone can use MRC. We can immediately build the servers we need without having to plan, purchase, maintain, or dispose of cloud hardware equipment. MRC is available all the time, anywhere in the world where there is a network. In addition, MRC has also published official tutorials to provide convenience for all users and guide you on how to use the cloud, which makes the use of the cloud easier to use.

2.3.2.4 Scale

With seemingly limitless capacity and the ability to scale resources either automatically or with some supervision, it’s simple to go from one instance of an application or workload to many.

We can allocate or adjust resources according to our own needs according to the limited available usage. We can run scripts through ssh to adjust instances. We can use Ansible to facilitate cloud deployment and set the required openstack content and its dependencies with remote installation into the cloud environment.

2.3.2.5 Latency and performance

Given the broad geographic footprint of most IaaS providers, it’s easy to put apps and services closers to your users, reducing latency and improving performance.

For MRC, it has superior performance and there is seldom error in the possess of using. Since MRC is a private cloud model, private clouds are usually deployed in the firewall of the organization's intranet to ensure efficiency and good network performance. Therefore, compared with the public cloud deployment model, MRC has more powerful cloud computing.

2.3.2.6 Security

In MRC, the allocated cloud resources belong to a client, which is our group. Therefore, we can configure the cloud infrastructure and systems ourselves to provide a high level of security. For example, we can log in to the cloud instance through an encrypted key pair, or we can customize instances and data volumes.

2.4 AURIN

Australian Urban Research Infrastructure Network (AURIN) is a EIF/NCRIS federally funded project whose lead agent is University of Melbourne. AURIN Clouds use severs purchased (VMware) for production system. It aimed to establish an e-Infrastructure for Urban and Built Environment Researchers. AURIN has some good features:

- Distributed completely with heterogeneous datasets
- Data interrogation services
- Security (unit level data, health data, commercial data)
- Online analysis tools
- Collaboration

We download our needed data from AURIN to analyze our scenarios.

2.5 Twitter API

Our twitter data is from the twitter API. The Twitter API provides the tools we need to contribute to, engage with, and analyze the conversation happening on Twitter. We apply to be a developer of twitter and use it to harvest the needed data for analysis of our scenarios alongside with the data from AURIN.

2.6 Front-end

2.6.1 Flask templates - HTML

HTML is a standard markup language used to create web pages; it is the cornerstone of building websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It is often used with CSS and JavaScript to build websites.

HTML in Flask takes jianjia2 as template, imports the variable from back-end and put them in specific places in HTML, All HTML files are placed in the templates file to call.

2.6.2 Flask statics - JavaScript, CSS

JavaScript is a dynamic scripting language based on prototype programming and multi-paradigm. It is used to add various dynamic functions to web pages in this project. CSS is used to specify the layout of a web page. Both are put in statics of flask to make a link with HTML.

2.7 Back-end

2.7.1 Language - python

Python and Java have always been among the best in various popular programming languages. Although java is rigorous, robust, not error-prone, API reusability is stronger, but the amount of code is large and complex, and flexibility is not high.

We choose python because its syntax is simple and beautiful, powerful, the standard library and third-party libraries are very powerful. More importantly, It is suitable for processing cloud computing data for this project.

2.7.2 Flask

Flask is a lightweight web application framework written in Python, using Jinja2 as a template. Because we chose to use python language, flask is the first choice as a back--end frame compared with Django for some reasons:

- We only need to develop a lightweight website or specific micro services (API), we simply don't need the large and comprehensive components and functions that Django comes with.
- Flask has flask-restful package to provide the restful design which is needed in this project.
- Because this project uses a non-relational database. For Django, it has a built-in ORM. For NoSQL, there must be a large part of the functions that cannot be used. Flask has great flexibility in the database and does not affect our work.

2.7.3 Restful API

RESTful architecture is currently the most popular Internet software architecture. It has a clear structure, conforms to standards, is easy to understand, and is easy to expand.

Flask-restful is an extension of flask. It can help to uniformly set resources to the format of URI. The client uses the URI to request and call data from the server. Representational State Transfer means that the presentation layer of the data is changed to json or html, etc. In this project, we convert the data of couchDB processed by MapReduce into HTML and add it to the resource that sets the URI. If the client uses the URI, it will return the corresponding HTML.

2.8 Database

Database is a warehouse that organizes, stores and manages data according to the data structure. Database is mainly divided into relational database and non-relational database.

Compared with relational databases such as MySQL and Oracle that use relational models to organize data, it emphasizes Key-Value storage. Distributed NoSQL has advantages in storage speed and flexibility and is suitable for cloud computing in this project. In NoSQL, we choose CouchDB as the database for this project.

2.8.1 CouchDB

CouchDB is a document-oriented distributed database. CouchDB supports REST API, which allows users to use JavaScript to operate the CouchDB database and can also use JavaScript to write query statements. And CouchDB has MapReduce function to design and process the documents. It uses MapReduce and HTTP as API.

2.8.2 Pros and Cons

Pros:

- Document databases are more convenient and perform better than relational databases in this project.
- Supporting REST API means system with AJAX and CouchDB is How simple and convenient it will be.
- The Map/Reduce feature in CouchDB generates key/value pairs, which can be efficiently searched by keys, and data can be partitioned on multiple nodes without the need to query each node separately.

Cons:

CouchDB reads and writes a large amount of data very slowly, and in the process of updating data, the volume of the data file will become larger and larger, because the data is appended to the end of the file, the old data is not deleted, so you must prepare a larger amount for CouchDB In addition, the compact or replication process should be run regularly to remove old data through compression or copying.

3 Twitter Harvest Strategy

We mainly use Twitter Search API (Twitter API, 2021) and tweepy (Twitter API v1.1 Reference, 2021) library to realize our Twitter harvester functions. The search function is defined as follows:

```
API.search(q, *, geocode, lang, locale, result_type, count, until, since_id,
max_id, include_entities)
```

, where returns a collection of relevant Tweets matching a specified query.

3.1 Search Tweets Containing Specific Keywords

By modifying the `q` parameter in the `API.search()`, we can search for Tweets containing specific keywords. For example, `q = "AFL"` is to search for Tweets containing "AFL" keyword.

3.2 Search for Tweets in Specific Regions

The tweety library supports searching tweets by users located within a given radius of the given latitude/longitude by modifying "geocode" field. Therefore, by consulting the relevant information, we get the geographic location data of six major cities and six States.

Table 1 Coordinates and Radius of Main Cities

City	latitide, longitude, radius
Melbourne	-37.813611, 144.963056, 60km
Sydney	-33.865143, 151.209900, 63km
Canberra	-35.282001, 149.128998, 16km
Adelaide	-34.928497, 138.600739, 32km
Brisbane	-27.469770, 153.025131, 71km
Perth	-31.950527, 115.860458, 45km

Table 2 Coordinates and Radius of Main States

State	latitide,longitude,radius
NSW	-31.253218,146.921097,500km
QLD	-20.917574,142.702789,740km
TAS	-41.454521,145.970673,150km
WA	-27.672817,121.628311,900km
VIC	-37.471310,144.785156,270km
SA	-30.000233,136.209152,560km

Through these data, we can get the tweets of corresponding regions.

3.3 The Strategy to Get A Mass of Data

Twitter API limits the free developer account to only return up to one hundred query results per call. In order to solve this limitation to obtain more data, we adopt the following strategies:

`API.search()` is able to return only statuses with an ID less than (that is, older than) or equal to the specified ID by specifying the `max_id` parameter.

Besides, the order of the one hundred data returned by `API.search()` is from newest to oldest, that is, the _id field is from large to small. Each time we record the smallest `_id` (that is, the _id of hundredth data) and then make `max_id = _id- 1`. Then loop the above steps so that we can solve the problem that only one hundred pieces of data are returned in a single query.

3.4 The Strategy to Avoid Duplication

Twitter API for free accounts can only search for data within 7 days. If all the data for the past 7 days are already gathered, the API will return duplicate data. We handle this situation by the following strategy. We will judge whether there is a duplicate _id field every time before we store the data into the database. If it does, it will prove that all the data from the past 7 days are already acquired and then the loop will be terminated. In this way, we can avoid storing duplicate data.

3.5 The Strategy to Avoid Being Blacklisted by Twitter

Two methods are adopted to avoid being blacklisted by Twitter due to short-term high-frequency searches:

(1) 15 mins break for every 900 `search API` calls.

(2) Run the Twitter harvester on 3 nodes separately, and for each node, search for tweets in different regions to avoid duplication.

4 System Functionality

In this section, the content contains the functional achievements, and four scenarios included the result of these scenarios. The content of each scenario describes the reason for choosing the scenario, data diagrams and analyse.

4.1 Functional achievements

1. Twitter harvest: The main data is collected and processed by this function.
2. The deployment of data warehouse, CouchDB.
3. Using the MapReduce function provided by CouchDB to process data so that it implements RESTful.
4. Using ansible to auto-build whole cloud infrastructure.
5. The frontend visualizes the processed data and makes data easily analyzed.

4.2 Cities to be presented

As Figure 4 showed, almost all data are placed at five cities, which includes Greater Melbourne, Greater Sydney, Greater Brisbane, Greater Adelaide, Canberra, since the majority of Australians live in these cities. The data from these cities is representative.

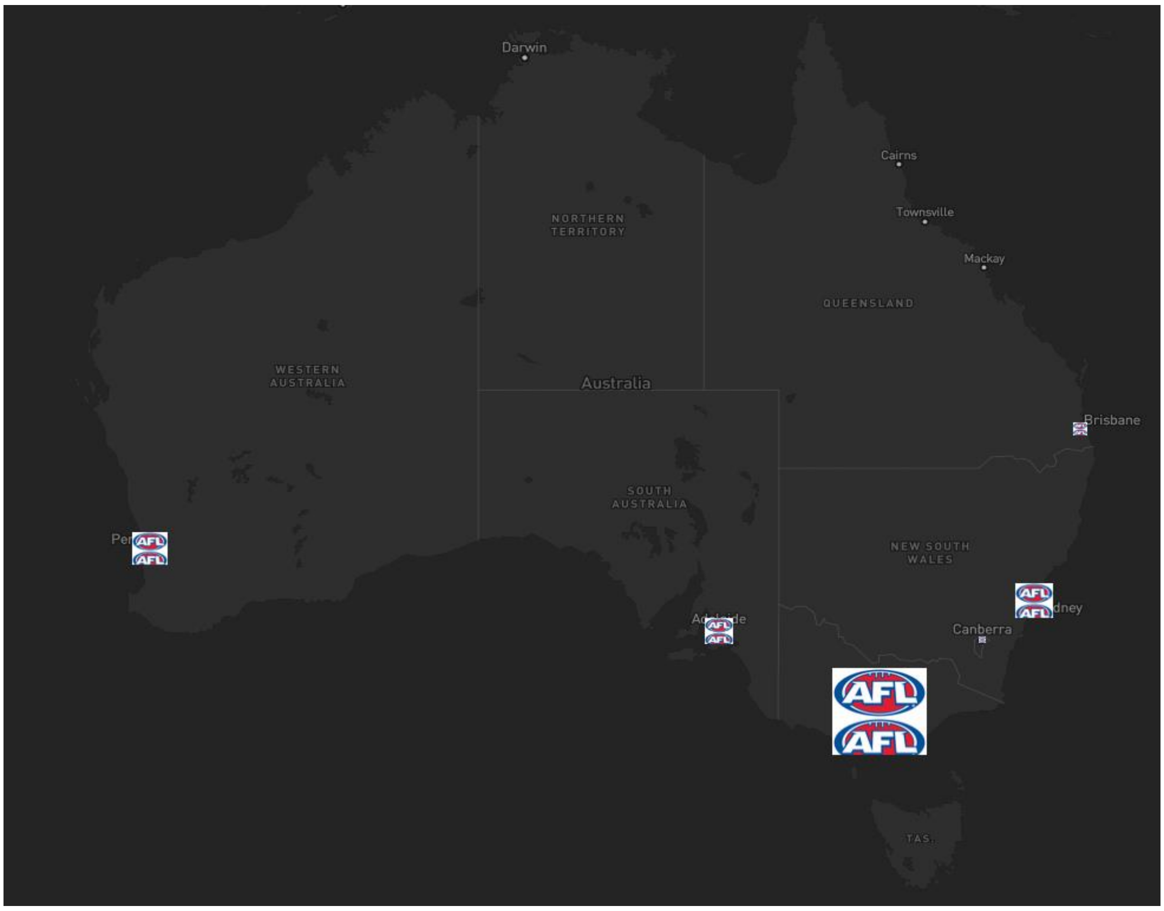
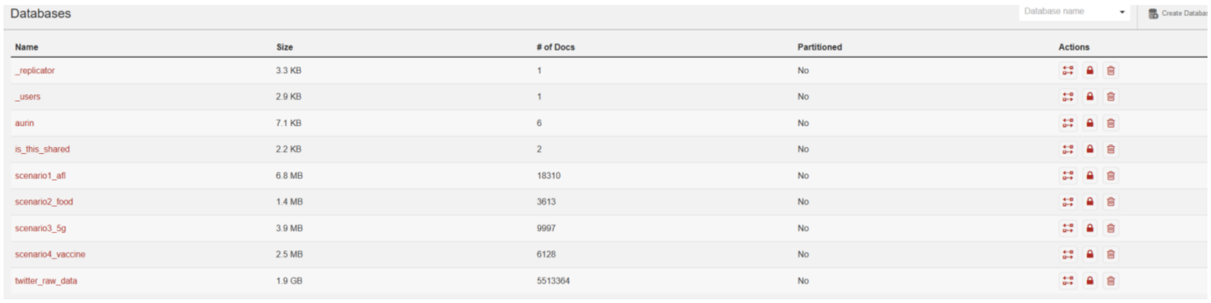


Figure 4 Greater Cities

4.3 Data stored



Name	Size	# of Docs	Partitioned	Actions
_replicator	3.3 KB	1	No	
_users	2.9 KB	1	No	
aurin	7.1 KB	6	No	
is_this_shared	2.2 KB	2	No	
scenario1_afl	6.8 MB	18310	No	
scenario2_food	1.4 MB	3613	No	
scenario3_5g	3.9 MB	9997	No	
scenario4_vaccine	2.5 MB	6128	No	
twitter_raw_data	1.9 GB	5513364	No	

Figure 5 Database

As you can see from Figure 5, the total size of data we collected is about 2GB, including 1.9GB of raw data from Twitter (consists of latest Twitter data gathered by Twitter harvesters and old Twitter data from "bigTwitter.json") . The data of scenarios are respectively 6.8MB,1.4MB,3.9MB and 2.5MB. The aurin data is 7.1KB.

4.4 Scenarios

The scenarios we present are:

1. Correlation between amount of twitters about "AFL"(the heats of "AFL") and the economy level, age distribution of citizen.
2. Correlation between the popularity of different food types and the economy level, age distribution of citizen.
3. Correlation between the views of the 5G and the level of education, age distribution of citizen.

4. Correlation between the views of the vaccine and the economy level, the level of education, age distribution of citizen.

4.4.1 The AFL related information analysis

4.4.1.1 Scenario description

This scenario focuses on analysing the relation between the heat of the "AFL" and the features of the citizen, which includes the age distribution and economy level in Australia. The cities include Greater Melbourne, Greater Sydney, Greater Brisbane, Greater Adelaide, and Canberra.

4.4.1.2 Reason why scenario chosen

The different group of citizens have different favour of sports. We are curious about how specific features of people influence the selection of sports. And AFL is an interesting sport that deserves analysis.

4.4.1.3 Graphical result

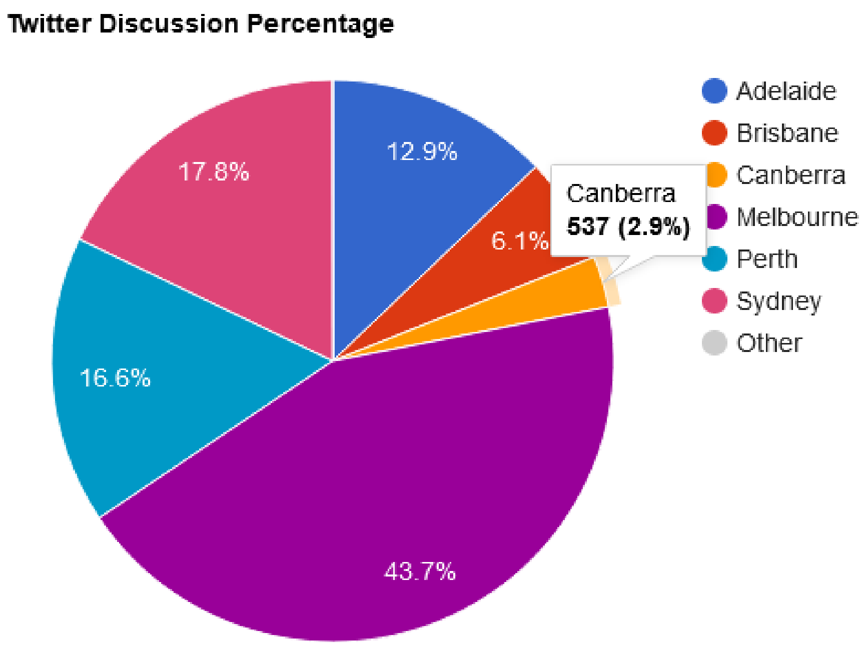


Figure 6 Twitter Discussion Percentage

Figure 6 shows the Twitter discussion percentage of six main cities, which are Adelaide, Brisbane, Canberra, Melbourne, Perth and Sydney. According to this data, we can figure out the heat of the topic "AFL" in the main part of Australia. As we can see, 43.7 percentage of the "AFL" related Twitter are from Greater Melbourne, which is twice as much as in other cities. The second hot area is Greater Sydney, occupying 17.8 percentage of whole "AFL" Twitter, followed closely by Greater Perth and Greater Adelaide. The total heat of "AFL" from these three cities is slightly higher than from Greater Melbourne. It is apparent that the citizens in Greater Brisbane and Greater Canberra show little interested in "AFL" with 6.1 per cent and 2.9 per cent of "AFL" related to Twitter.

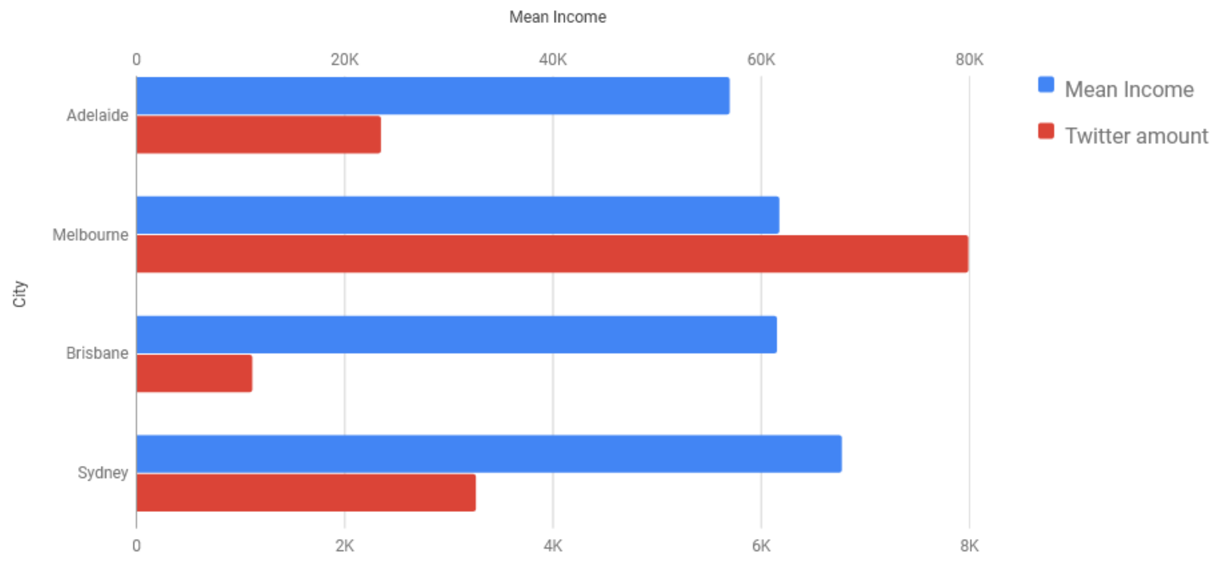


Figure 7 Mean Income and Twitter Amount

In order to figure out the correlation between income and the heat of "AFL", we put the income and "AFL" related twitter amount into a single bar chart. The blue bar indicates the mean income of the city, and the red bar indicates the twitter amount. It is easy to find that the heat of "AFL" in different cities are completely inconsistent while the mean income of these cities almost the same. We are happy to say that these cities have a fairly balanced level of economic development, but this is bad news for our analysis since it means that mean income has no relationship with the heat of "AFL".

Now let us focus on the age distribution. We can intuitively infer that the age distribution of cities can have an impact on the heat of "AFL" since people from different age groups should have a distinct preference for sports. For example, young people prefer confrontational sports while elderly people think sports not intense are more interesting.

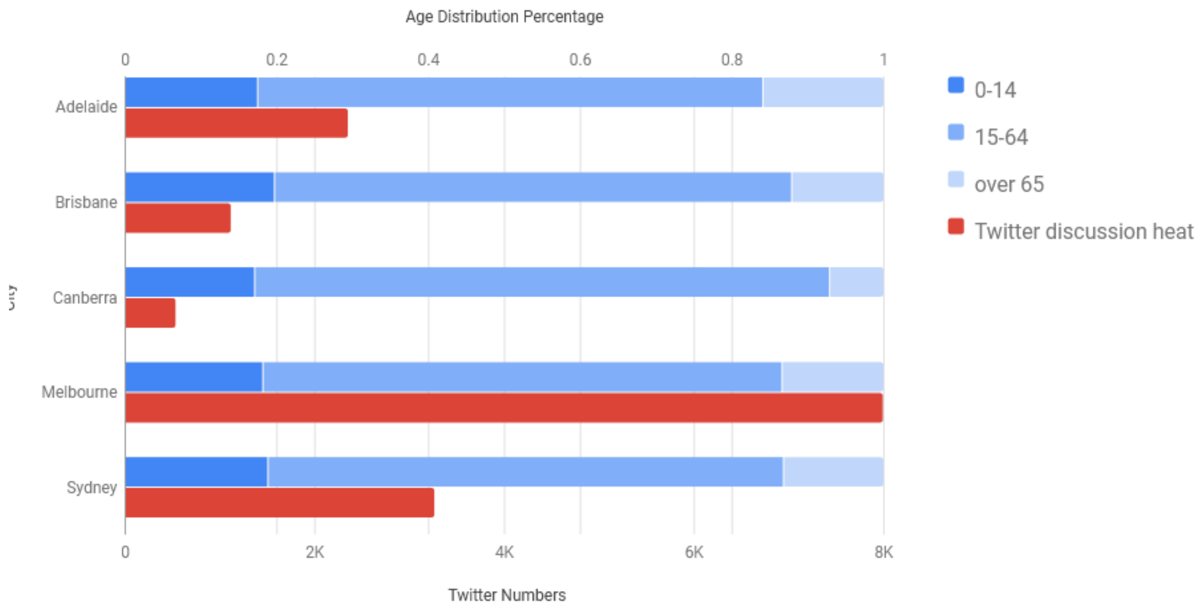


Figure 8 Age Distribution Percentage and Heat of AFL

Figure 8 shows the age distribution and heat of "AFL" in different cities. To present the age distribution, we use the percentage form. As we can see, the age distribution of these cities may be slightly different, but they all follow the same structure, that is, the age group of 15 to 64 is the main part of age distribution with about 70 per cent of people in this range, the age group of 0 to 14 and the age group over 65 occupy the 20 per cent and 10 per cent respectively. This is in line with people's perception that young people account for the majority. However, the same age distribution of these cities makes this information useless, which is a similar situation when we analyze the correlation between income and heat of "AFL". But we cannot hastily assert that the age distribution is independent with the heat of "AFL". We suppose that the heat of "AFL" may be related to the number of people in the age group 15 to 64.

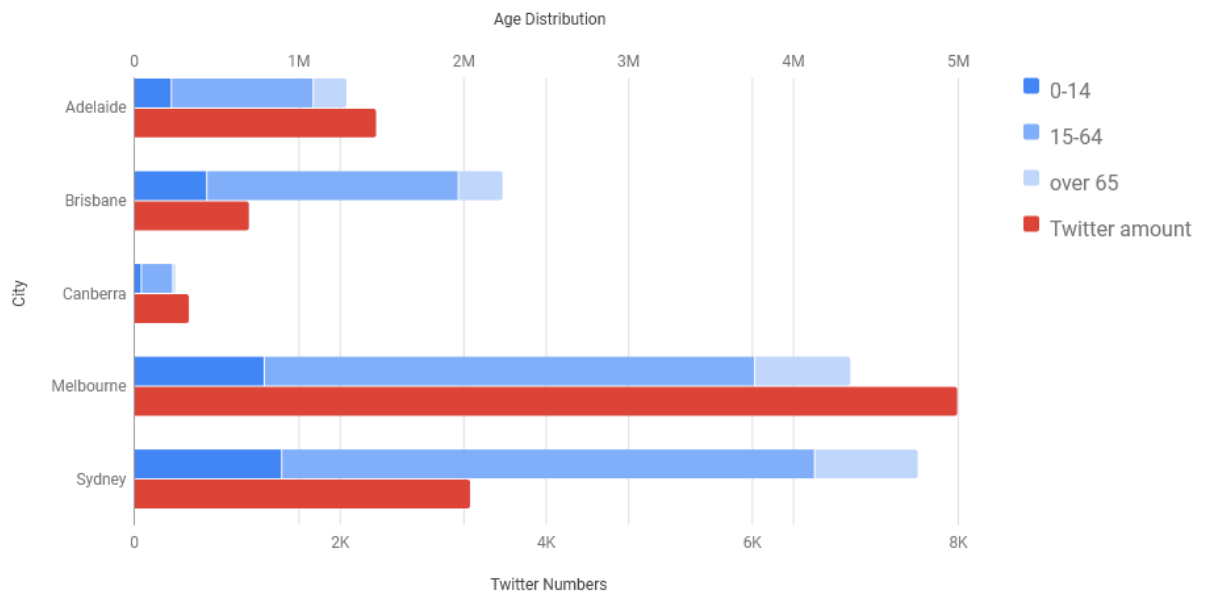


Figure 9 Age Distribution and Heat of AFL

This time the age distribution in the Figure 9 are not in the manner of percentage but the explicit number. The bar chart illustrates that the tendency that the more people from age group 15 to 64, the higher heat of "AFL", that is, there is basically a positive correlation between heat and the number of people from age group 15 to 64 excepting the Greater Brisbane and the Greater Sydney. So, we can ensure that one of parameters influcing heat of "AFL" is number of people from age group 15 to 64. In the future, we may try to figure out all factors which have impact of heat of "AFL". One factor we may explore is region.

4.4.2 The views on the 5G

4.4.2.1 Scenario description

As a new technology, we are desired to know about the people's views of 5G which is closely related to people's lives. We compare the sentiment of 5G and features of people in Australia.

4.4.2.2 Reason why scenario chosen

As a new generation of communication technology, 5G will greatly change people's daily life. There is no doubt that 5G will provide a lot of convenience for people. Also, different people may have distinct views of 5G. For example, young people always tend to use new things while older people hold opposite opinions due to the cost of study. We are interested in the correlation between the views of 5G(negative, positive or neutral) and features of people.

4.4.2.3 Graphical result

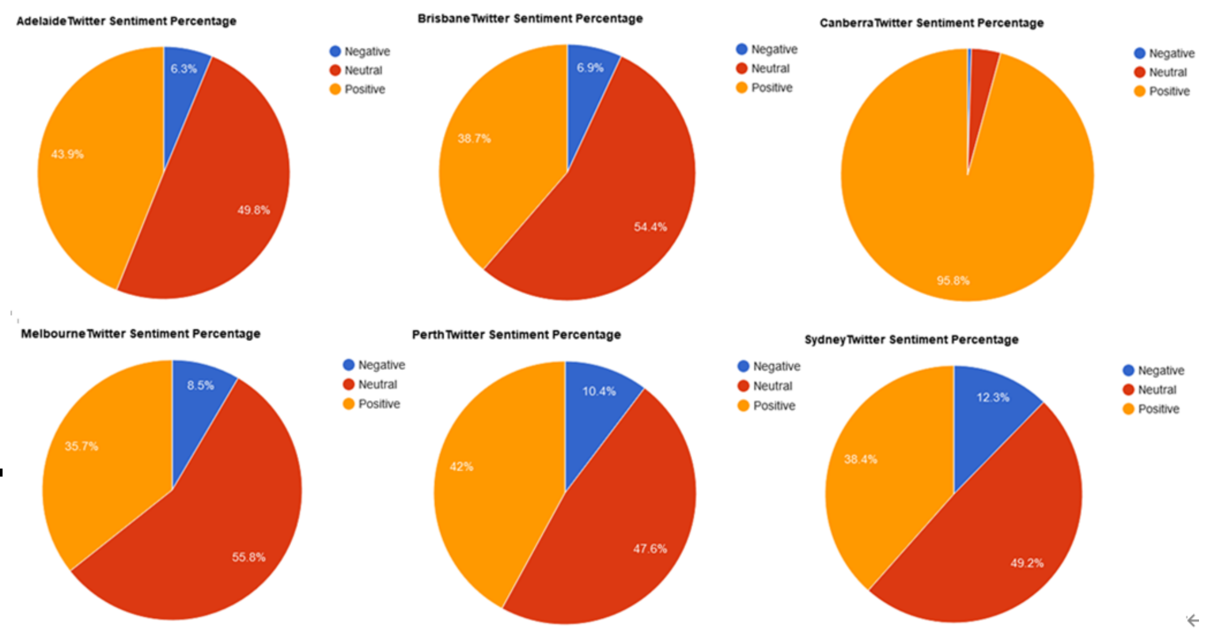


Figure 10 Twitter Sentiment Percentage

Figure 10 shows that in all cities, the people who hold negative views on 5G are the minority. The difference in these cities is the percentage of positive and neutral views on 5G. As figure 7 shown, the neutral views in these six cities are all higher than positive views, and the difference is the degree of neutral views exceeding the positive views. Firstly, we exclude Canberra, where the positive views are in the ascendancy, which makes the data of Canberra like an outlier. In Sydney and in Melbourne, the difference between neutral views and positive views are greatest, where the percentage of neutral views is higher than positive views, about 11%. For that, in Adelaide, Brisbane and Perth, the difference is about 6%. Generally speaking, the people in these six cities hold relatively active views.

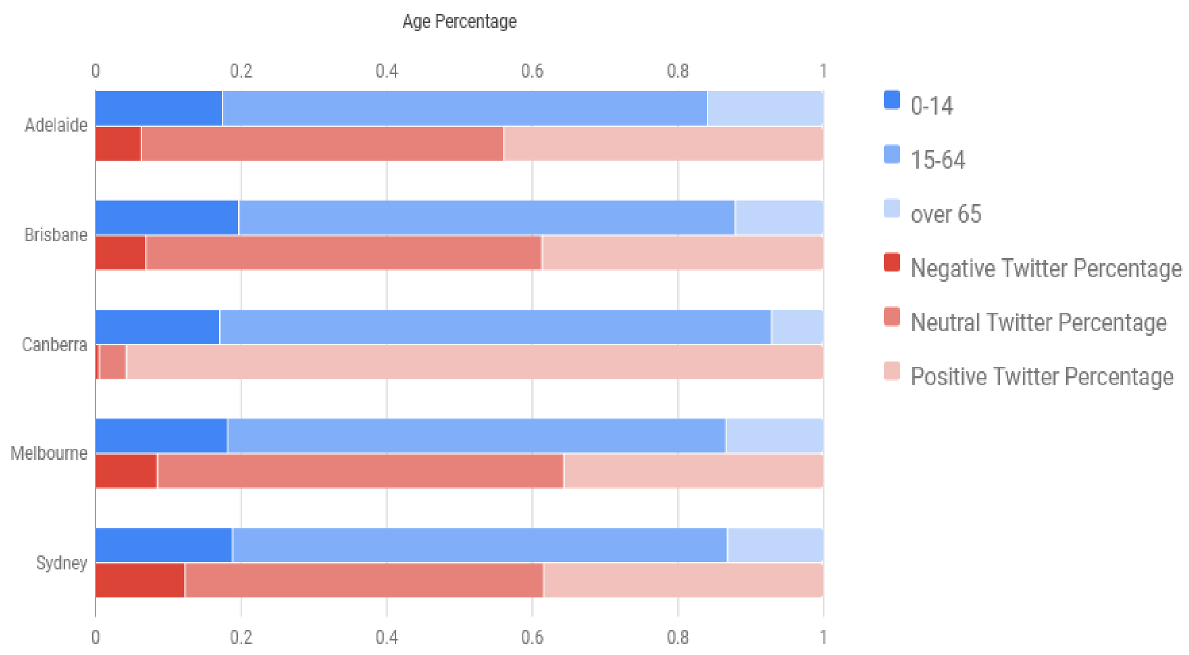


Figure 11 Age Percentage and Sentiment Twitters

As Figure 11 shown, almost all people in Canberra, where the people from age group 15 to 64 is 76% holds positive views on 5G. While the percentage of people from age group 15 to 64 are all 68% in other cities and the positive views in these cities are lower than in Canberra. So there may be a positive correlation between the percentage of people from the age group 15 to 64 and positive views on 5G.

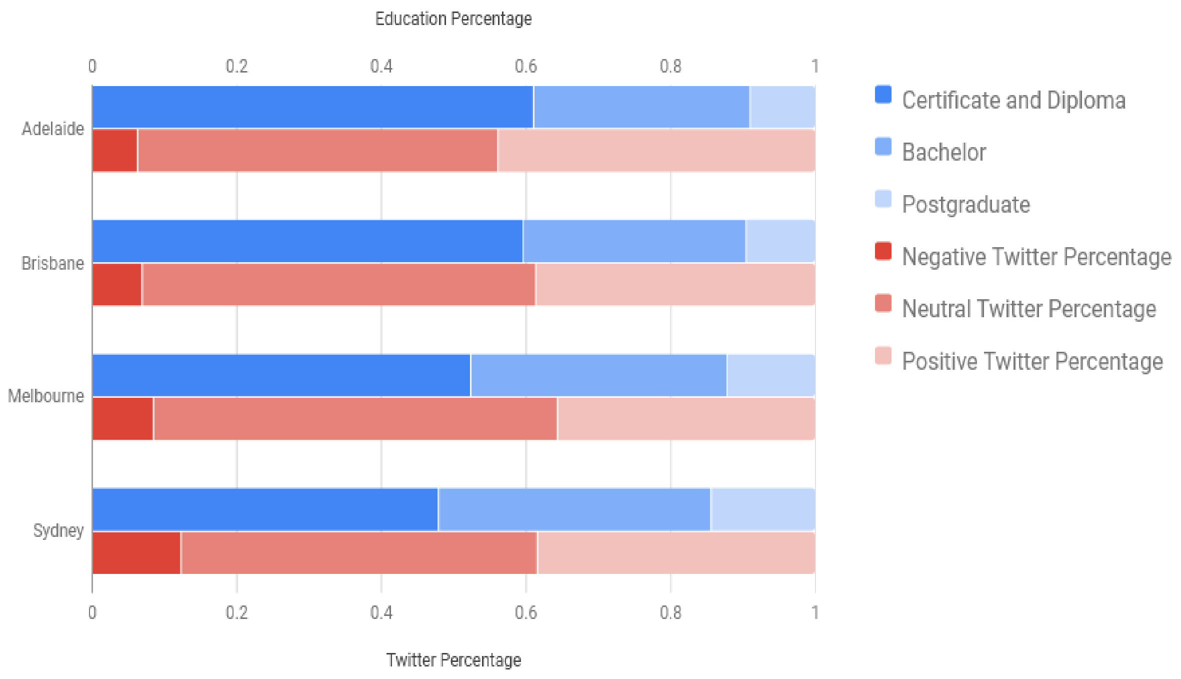


Figure 12 Level of Education and Sentiment

From Figure 12, we can easily find that there is a strict correlation between the level of education and negative views. Here we define the Certificate and Diploma as low level of education and Bachelor and Postgraduate as high level of education. As figure 9 shown, in the cities with the lower level of education, people prefer to hold negative views on 5G. In other words, people with a high level of education tend to hold active views on 5G. This conclusion is in line with the intuition that people with a high level of education is willing to accept new technology.

4.4.3 The views on vaccine

4.4.3.1 Scenario description

This scenario focus on the views of vaccine at the background of COVID-19. We try to explore the relationship between the opinions of vaccine and features of people.

4.4.3.2 Reason why scenario chosen

As we have known, the Covid-19 spread rapidly around the world in the last year, which has a great impact on people's daily life and causes a lot of deaths. Countries with high levels of medical care have invested huge sums in the development of the vaccine. However, not everyone has confidence in the effect of the vaccine. May some people think that vaccine can not protect them from the virus and even may hurt their health. So, it is necessary to analyze the correlation between the opinions of vaccine and features of people, which may help government encourage people to get the vaccine.

4.4.3.3 Graphical result

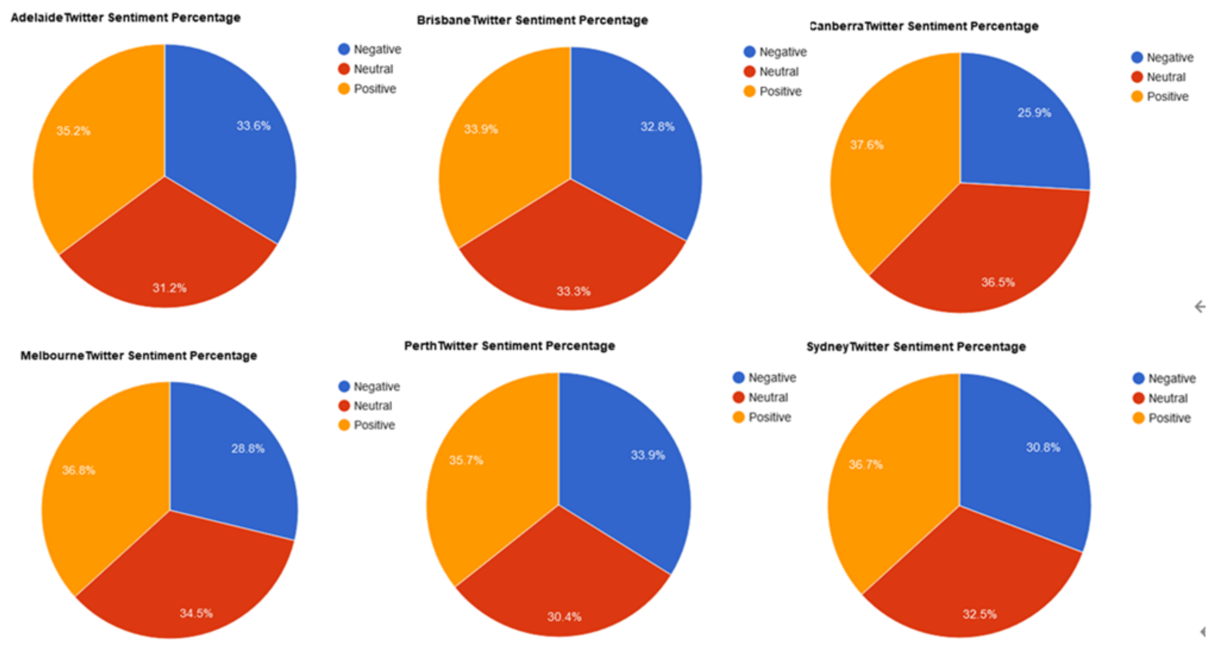


Figure 13 Percentage of Sentiment of Vaccine Twitter

The pie charts from Figure 13 show the percentage of Covid-19 vaccine sentiment twitter in different cities. The sentiment can be divided into three-part, positive, neutral and negative, respectively marked with the different colour yellow, red and blue. It is easy to find out that in Adelaide, Brisbane and Perth, the people hold balance views on the vaccine, that is, the percentage of sentiment in this cities are slightly different, three sentiment degree are almost all about 33 per cent. The data in Brisbane is especially balancing while in Adelaide and Perth, the positive views take the first place, followed by the negative views, and the neutral views take the last place. In Greater Melbourne and Greater Sydney, the positive opinions are majority which apparently higher than the other two views, which means the people in these two cities may be active to take the vaccine. In Canberra, the number of people who holds positive and neutral views are similar and are greatly higher than negative. The negative views proportion are smallest in these six cities.

So how the factors like income, age distribution and level of education influence the sentiment?

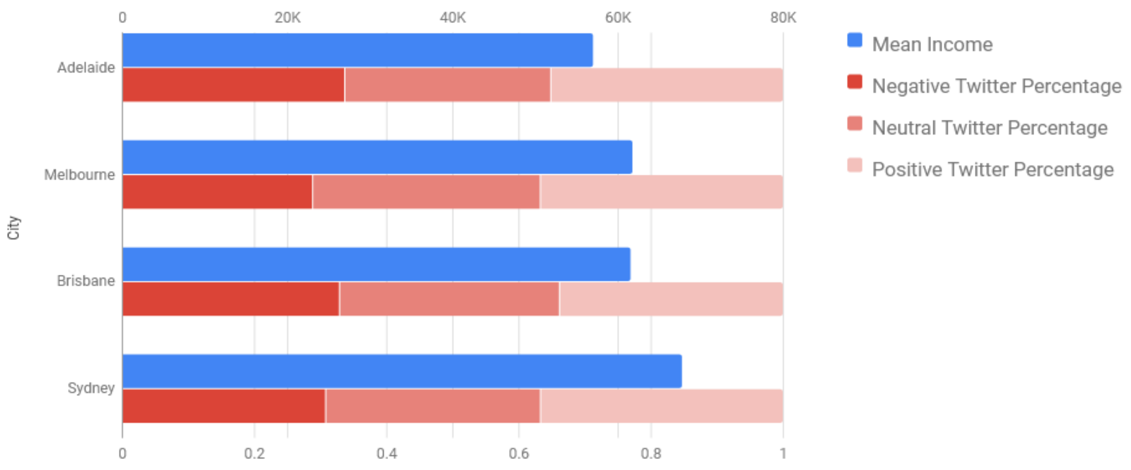


Figure 14 Income and Percentage of Different Views

From Figure 14, we can find that with the first two places in the mean income, Sydney and Melbourne holds the positive views as the majority. And in Adelaide and Brisbane, where the income of Brisbane is slightly higher than that of Adelaide, the negative views in Adelaide is more than in Brisbane. So, we can make the preliminary conclusion that the higher income, much more active in views of the vaccine.

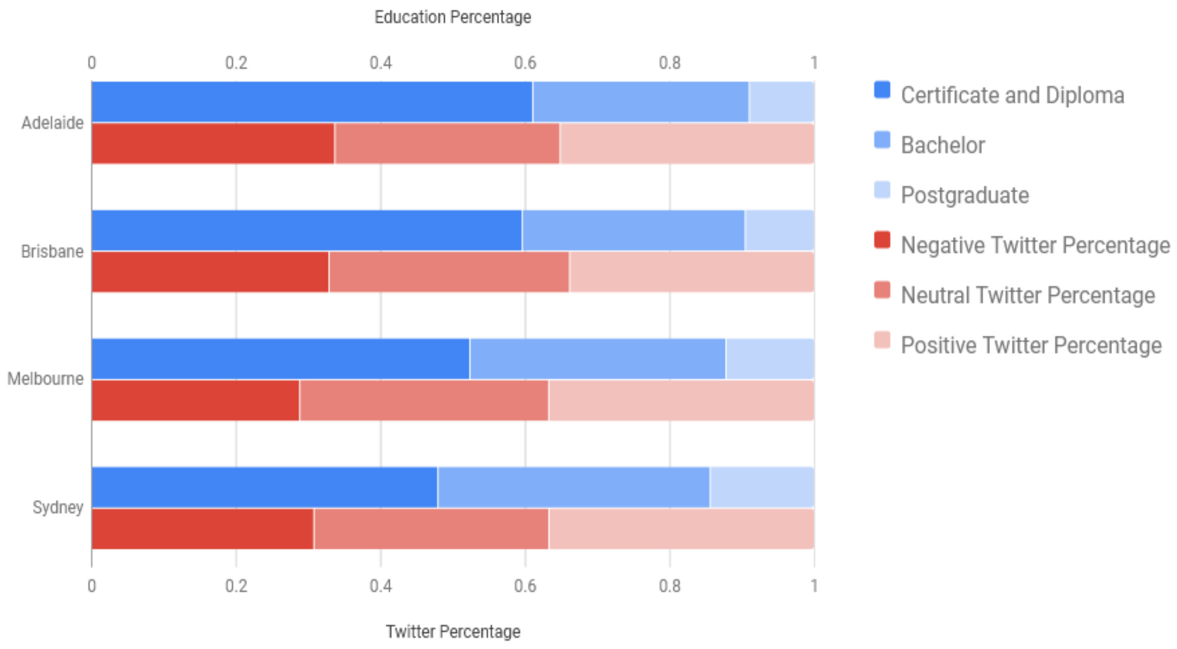


Figure 15 Educational Level and Percentage of Different Views

Figure 15 shows the relationship between the level of education and the sentiment of the vaccine. The Melbourne and Sydney, whose number of well-educated people (Bachelor and Postgraduate) takes first two places hold the positive views as the majority. The people in Adelaide hold more negative views than in Brisbane, where the educational level is better. (The number of Certificate and Diploma in Brisbane is lower than in Adelaide). So, we can say that there is a positive correlation between the level of education and views on the vaccine.

4.4.4 The preference of food type analysis

4.4.4.1 Scenario description

In this scenario, we determined to analyze the popularity of different food types and their relationship with the income level and age distribution of citizen in the main part of Australia. The data relating to the popularity of food is collected from Twitter, and the features of citizen come from Aurin.

4.4.4.2 Reason why scenario chosen

The reason why we choose this scenario comes from the facts that a great amount of residents comes from different countries. Naturally, these people should have diverse food preference. Besides, features such as level of education may impact the food selection since, intuitively, people with the high level of education prefer to select healthy food.

4.4.4.3 Graphical result

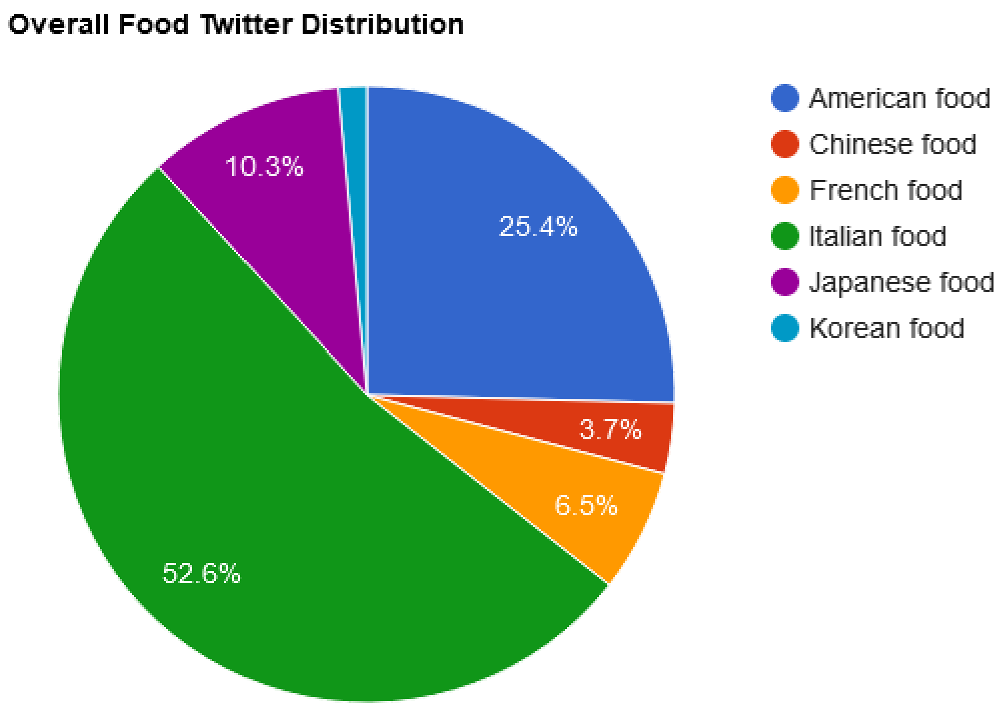


Figure 16 Overall Food Distribution

As Figure 16 showed, the most popular food type is Italian food which is over half of the distribution. And second popular food type is American food which takes 25.4 per cents, followed by Japanese food and French food. It seems that rarely people choose Chinese food and Korean food.

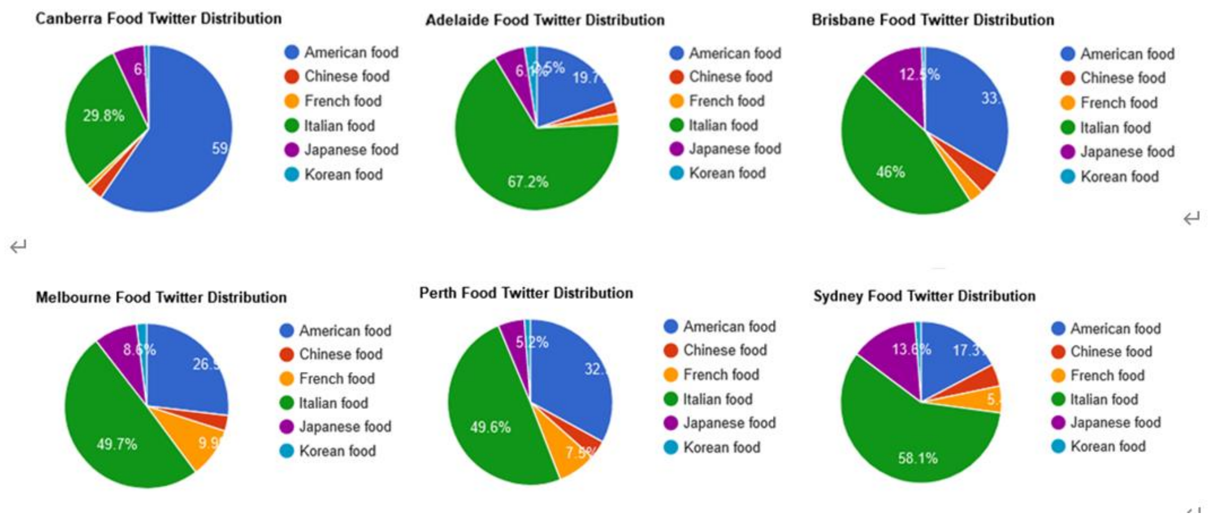


Figure 17 Food Distribution in Different Cities

As pie charts in Figure 17 shown, almost all people in these cities prefer to eat Italian food except Canberra, where American food is the majority. And second popular food is American food in cities except for Canberra. For third place, there are differences in these cities. The people in Melbourne and Perth prefer French food while other cities prefer Japanese food.

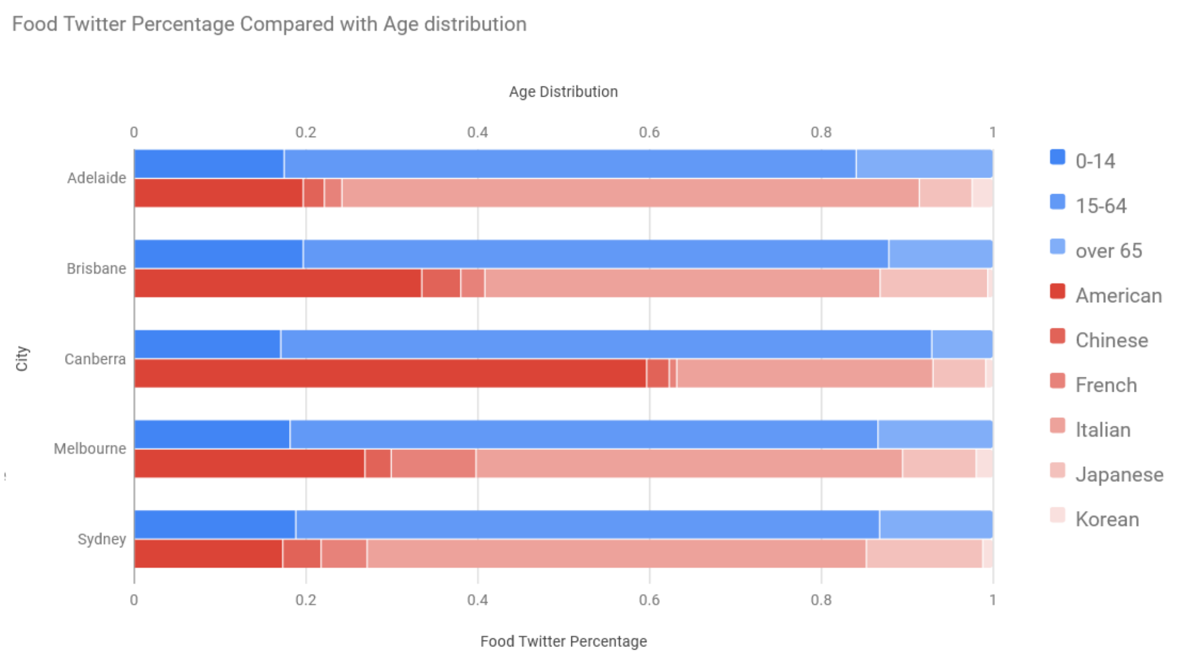


Figure 18 Age Distribution and Food Distribution

Figure 18 shows that for the age group 15 to 64, the percentage of that in Canberra is 76 per cents, which is higher than that in other cities with all about 68 per cents. So, we can infer that the young people may prefer American food. Although people in cities except Canberra treat Italian food as their first choice, the degree of preference are different in these cities. We can see that Adelaide has the highest percentage of people choosing Italian food, which is 67.2 per cents, and Adelaide has the highest percentage of people from the age group over 65, which is 16 per cents, while other cities are all about 13 per cent. So older people may prefer Italian food.

5. User Guide

5.1How to build and deploy the system from scratch

Before running the shell scripts, there are couple preparations must be done first.

1. Go to your MRC dashboard, download the *OpenRC file* by clicking your profile icon and choose download.
2. Rename the downloaded file to "*openrc.sh*" and move it under the root project folder.
3. Navigate to "Key Pairs" under "Project – Compute" and create a new key pair by clicking "Create Key Pair".
4. Fill in key name and choose key type as "SSH Key".
5. Save the downloaded *.pem* file to a directory where you can easily navigate to. (Warning! DO NOT share this key pair with anyone or public it!!!)
6. Open file "*mrc.yaml*" under *./ansible/vars/*, after field "*instance: key_name:*", replace the old key name by the key name you just created.
7. Click "User" on the top right and click "Setting"
8. Navigate to "Reset Password"
9. Click "Reset Password" and write down the new password somewhere safe. You will need to enter this password later when running the shell scripts.
10. *[Optional]* Open "*mrc.yaml*" file again, and decrease the "*vol_size*" if there no 500 GB storage available in your project space.

After that, you can run the shell scripts following the following instructions to run Ansible.

1) cd to the project folder
2) in your terminal, enter "*sh ./run_first.sh*" (or simply double-click the "*run_first.sh*" file)
3) Terminal will ask you to enter password, which is the password from *step 9* previously.
4) Copy the smallest IP address. By smallest we need numerically smallest, for example, 172.0.0.0 < 172.0.0.1
5) Then open file "*couchdb_config.py*" under *./app/backend/*, replace all strings "*172.xxx.xxx.xxx*" with the smallest IP address you just copied.
6) (Make sure you are still at the project root folder in your terminal) enter "*sh ./run_last.sh*" and enter the same password again.
7) When this script finishes, app is up on the cloud and can be accessed by entering the smallest IP address in step 4 followed by ":8003". For example, "172.168.123.43:8003". Make sure you are using The University of Melbourne's network as well.

Note: Here assume the project space in MRC is empty. If not, uncomment the second line in "*run_first.sh*" first before running it.

5.2 Deploy instances

In this step, we will manage the cloud infrastructure, such as creating/deleting volumes, security groups and instances, on the UniMelb Research Cloud. We also will use Ansible scripts to dynamically manage them with very configurable settings.

Table 3 Ansible Playbook outline

Shell script	.yaml	Hosts	var	Role	Description
run_first.sh	main	localhost	mrc.yaml	common	Install required packages Such as pip, onpenstacksdk
				images	Display all available images in MRC
				volumes	Create volumes based on mrc.yaml
				security-groups	Create security groups based on mrc. yaml
				instances	Create instances based on mrc.yaml
				post-instances	Add hosts to Ansible in-memory inventory file

5.2.1 How to run

To deploy instances, simply run "*sh run_first.sh*" under the project root folder. The shell script use Ansible Playbook command to run the *main.yaml* file, which runs on localhost.

If the user wants to delete all instances, volumes and security groups, simply uncomment the second shell command in "*run_first.sh*". As it runs the "*uninstall_server.yaml*", which will make the project space in MRC become empty. Uncomment this line with caution.

5.2.2 How to customize

By default, we create 3 *volumes* with size 180, 150 and 150; we create *security groups* which open port 22, 80, 443, 5984, 5986, 4369, 9100-9200, 8001, 8002, and 8003; we also create 3 *instances*. We use *image ubuntu 20.04 LTS*, use our own key pair and with instance *flavor uom.mse.2c9g*.

If the user wants to customize or scale up this deployment, simply modified the *mrc.yaml* under *./ansible/vars/* . For example, by adding more volumes and instances will create more remote

VMs (if available). The user can also change instance flavor as well. However, the image instance (ubuntu) and availability zone are not advised to be changed.

5.3 Set up instances

In this step, we will configure the instances. Particularly, we will focus on fix proxy settings and install packages. This is an essential step as without it apt-get cannot pull and docker cannot be installed or run properly.

Table 4 Ansible Playbook outline

Shell script	.yaml	Hosts	var	Role	Description
run_first.sh	main	COMP90024	config.yaml	add-proxy	Upload environment proxy file to the remote VM to configure proxy settings.
				mount-volumes	Mount volumes that are created previously to each instance by using file system *xfsprogs.* The mount point is defined in *config.yaml*.
				install-packages	Install required packages (curl, python…) and docker in each instance.
				post-docker-installation	Configure the docker proxy setting and create a user group called docker in each VM for permission arrangements.

5.3.1 How to run

This step will not be run independently. It is dependent on task 1.1. When the user runs "*run_first.sh*", this set-up procedure will be run automatically after instances are created.

5.3.2 How to customize

This step cannot be customized a lot by the user. Most of them cannot be changed. However, the user can add more dependencies in role "install-packages" if required.

Note: remember to copy the smallest IP addresses among the remote VMs to the couchdb_config.py before continue to next step.

5.4 Deploy CouchDB

In this step we will use Ansible to automatically run the CouchDB docker container and set up clusters among 3 nodes.

Table 5 Ansible Playbook outline

Shell script	.yaml	Hosts	var	Role	Description
run_last.sh	deploy	COMP90024	config.yaml	couchdb	- create required directories - upload docker-compose file - run docker-compose file - set up CouchDB cluster based on config.yaml

5.4.1 How to run

By running “run_last.sh”, the first role Ansible will do is to copy the entire app directory to remote. Then it will start to deploy CouchDB specified by role “couchdb”. So this deployment will not run independently. If the user really wants to deploy CouchDB only, then comment out all other roles. So that only 1 role, which is current one, will be run.

5.4.2 How to customize

Many things can be customized, and all of them are listed in the file *config.yaml*. In this file, under the field “couch:”, all required variables are defined here. The user can replace the content as they wish. For example, change version to latest, or change the user and password to something more secure.

However, this is also the least part where the user can scale up. As the current implementation forces there must have only 3 nodes.

5.5 Deploy Twitter harvester

In this step we will use Ansible to automatically run the Twitter Harvester docker container to crawl data and upload to CouchDB.

Table 6 Ansible Playbook outline

Shell script	.yaml	Hosts	var	Role	Description
run_last.sh	deploy	COMP90024	config.yaml	harvester	- run harvester docker container on each VM.

5.5.1 How to run

This is covered in “run_last.sh”, so no need to run it independently.

5.5.2 How to customize

It cannot be customized mostly, as the implementation forces that there must have only 3 nodes. But the user can change the Dockerfiles under *./app/backend/crawler/* as they wish. Current Dockerfiles are the simplest with no extra functionalities except running a python file.

5.6 Deploy App

In this step we will use Ansible to automatically run the Flask docker container and deploy our web application.

Table 7 Ansible Playbook outline

Shell script	.yaml	Hosts	var	Role	Description
run_last.sh	deploy	COMP90024	config.yaml	app	- run Flask docker container on 1 remote only.

5.6.1 How to run

This is covered in “run_last.sh”, so no need to run it independently. And as it depends on CouchDB, copy-directory, it must be run after all of them are finished. Do not run this individually, except CouchDB is already running, and the app directory is completely copied.

5.6.2 How to customize

It cannot be customized mostly, the only the user can customize is again the Dockerfile. Current Dockerfiles are the simplest with no extra functionalities except running a Flask application.

6. Conclusion

In this project, we build an Australian media analytics system using MRC and combining datasets from AURIN to analyze some interesting scenarios about Australian life in major cities. The social media data we collect are from Twitter which is one of the most popular social media platform. Twitter data always contains geographical information, which makes analyzing information according to cities become possible. In the project, we use Ansible and docker to auto-deploy the whole system. That is, use Ansible to create three instances and allocate the resource such as memory or volumes. After resource allocating, we use docker to deploy CouchDB, front-end and back-end. In order to get data from Twitter, we developed the Twitter harvest based on the Twitter API and process the data using some sentiment tools of python. We also get the related information from AURIN which used to compare with Twitter data for analysis. The AURIN data are downloaded from the AURIN portal. The raw data and processed data are stored on CouchDB. For data analysis, we use the MapReduce. Since the system should be high available and can always accessed. We use the clusters CouchDB with one master node and two slave nodes. There is replication in these two slave nodes so that when the master node crash, we can access the service through accessing slave nodes. That is, this cluster CouchDB increases the availability of the system and guarantee the system to be high fault-tolerant. We design the front-end to represent the data, and the design of the back-end follow the RESTful API. By accessing the portal, you can see the whole Australia map and find collected data presented using different size images. Also, you can access the figures to show the information for analysis. But there are still many problems that should be improved. For example, the number of scenarios analyzed is not enough with sufficient data from Twitter and AURIN. And the metrics used to analyze scenarios are not diverse. In the future, we may come up with more scenarios to take full advantage of this mass data. Also, we can improve the design of the backend, making the system more flexible and the front-end UI can be more beautiful.

References

Mell, P., & Grance, T. (2011). *The NIST Definition of Cloud Computing.* Retrieved from National Institute of Standards and Technology: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Twitter API. (2021). Retrieved from Twitter Developer: https://developer.twitter.com/en/docs/twitter-api

Twitter API v1.1 Reference. (2021). Retrieved from Tweepy: https://docs.tweepy.org/en/latest/api.html