Cluster and Cloud Computing

2021S1 Project 2 - Group22

Ran Liang
The University of Melbourne
1162222
ranliang@student.unimelb.edu.au

Xindi Fang
The University of Melbourne
749394
xindif1@student.unimelb.edu.au

Yubo Sun
The University of Melbourne
1048638
yubsun@student.unimelb.edu.au

Yanhao Wang
The University of Melbourne
1142087
yanhaow@student.unimelb.edu.au

Yulun Huang
The University of Melbourne
910398
yulunh1@student.unimelb.edu.au

Abstract

This report is aiming to provide readers a detailed explanation to the cloud based system developed by our group in analysing tweets in various scenarios. In the first section, a simple user guide will be provided, including steps about how to create a ready-to-run system and the usage of the system. In the second section, we will give a detailed explanation of the design and architecture of the system. The pros and cons in relation to the choice behind will also be discussed in the same section. In the third section, a stepped demonstration from twitter harvesting and preprocessing to CouchDB utilization for storage and analysis will be offered. In the fourth section, the report will provide an analysis to all 5 scenarios we choose and the reason behind it. Those 5 scenarios include examining the harvest twitters with respect to language, source, time and testing on the attitudes towards 2 interesting topics, covid vaccine and unemployment. In section 6, we will discuss the difficulties during the developing process of our system and how we deduplicate the tweet. Videos demonstrating the processes in earlier sections will be presented in Section 7 and the git repository storing all the codes for the system will be shown in Section 8.

Key words: Twitter / CouchDB / AURIN / Melbourne Research Cloud/ Docker / Flask

Contents

1. User Guide

In this section, a simple user guide is provided, including steps about how to create a ready-to-run system and the usage of the system.
All the system deployments are accomplished using Ansible. The files and command line executed below are all in path relative to the ./ansible/ directory. Variables are configured in host_vars/instance_config.yaml, including volumes configuration, security groups, instances configuration, and other common variables. The deployment playbooks are organized as follows.

1.1 Create instances

First, we configure the local host and create instances in this step. The following roles are for the local host.

- *openstack-common*: install and update python-pip; install openstacksdk; add group key to default ssh directory on host (install dependencies on the host).
- *openstack-images*: show all available Openstack images.
- *create_instances* (with following tasks included):
  - *openstack-volume*: create volumes on MRC.
  - *openstack-security-group*: create all the security groups we need and the corresponding rules.
  - *openstack-instance*: create instances on MRC and create a file with the IP addresses of the instances created .

1.1.1 How to run

. ./create_instances.sh

Or run the command written in the *create_instances.sh* file directly.

1.2 Config instances

Then we need to configure the created remote instances.

- *set-proxy*: edit proxy settings to allow the instances to have access to external network.
- *setup-docker* (with following tasks included):
  - *common*: install dependencies on the remote host
  - *mount-volumes*: format the volumes with specified file system and mount them to corresponding instances
  - *docker*: create required environment for on remote instances, including install dependencies and add Docker apt repository, etc. Then install and configure Docker.

1.2.1 How to run

. ./config_instances.sh

Or run the command written in the *config_instances.sh* file directly.

1.3 Set up CouchDB cluster

For setting up a CouchDB cluster, first we run the *setup-couchdb* role on all remote instances to start a Docker container with a CouchDB. Then on the masternode specified before, we run the *setup-couchdb-cluster* role to set up a cluster and all other nodes to the cluster.

1.3.1 How to run

. ./setup_couchdb.sh

Or run the command written in the *setup_couchdb.sh* file directly.

1.4 Deploy twitter harvester

For deployment of twitter harvester, we first install dependencies required such as tweepy and couchdb. Then we copy the harvester scripts to remote instances and execute.

1.4.1 How to run

. ./ deploy_harvester.sh

Or run the command written in the *deploy_harvester.sh* file directly.

1.5 Frontend deployment

To deploy the frontend, we first copy the files we need to remote instances and then install dependencies required such as flask and gunicorn. Then enable the web server.

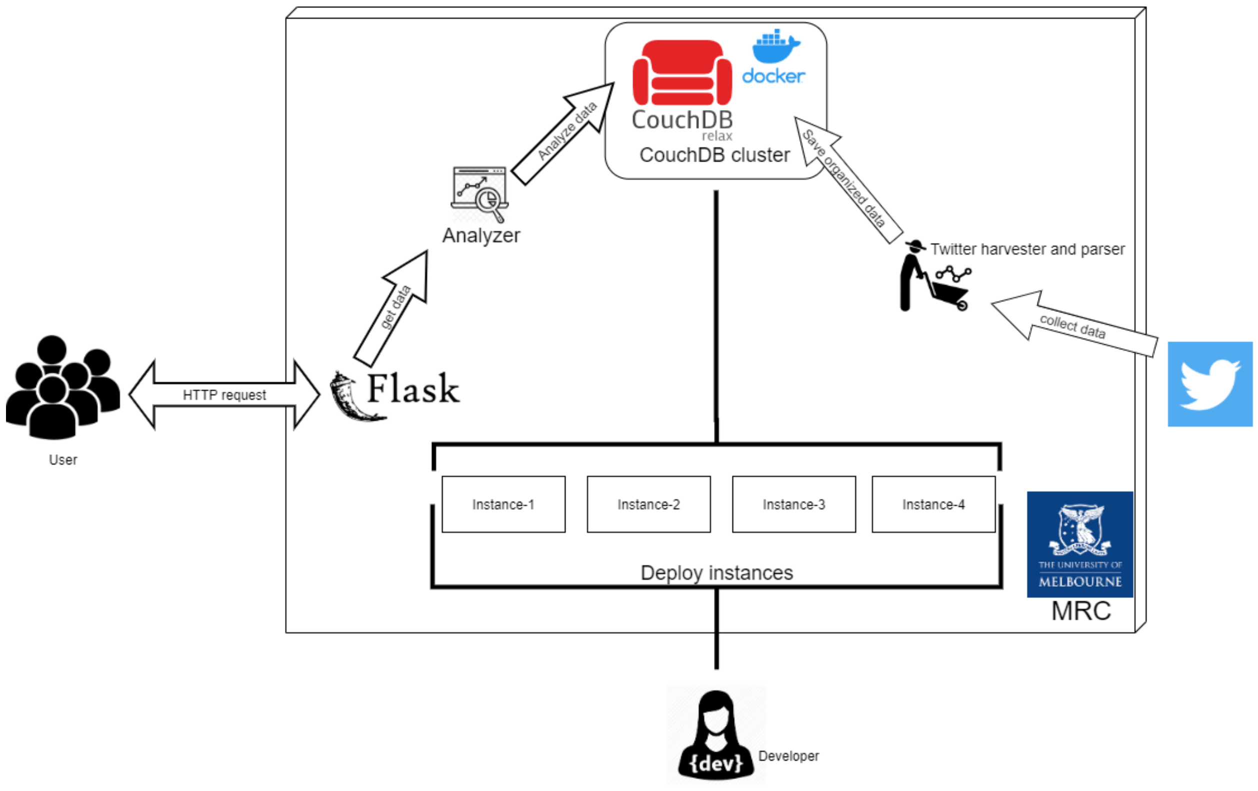
1.5.1 How to run

. ./ deploy_frontend.sh

Or run the command written in the *deploy_frontend.sh* file directly.

2. System Architecture And Design

2.1 System Architecture

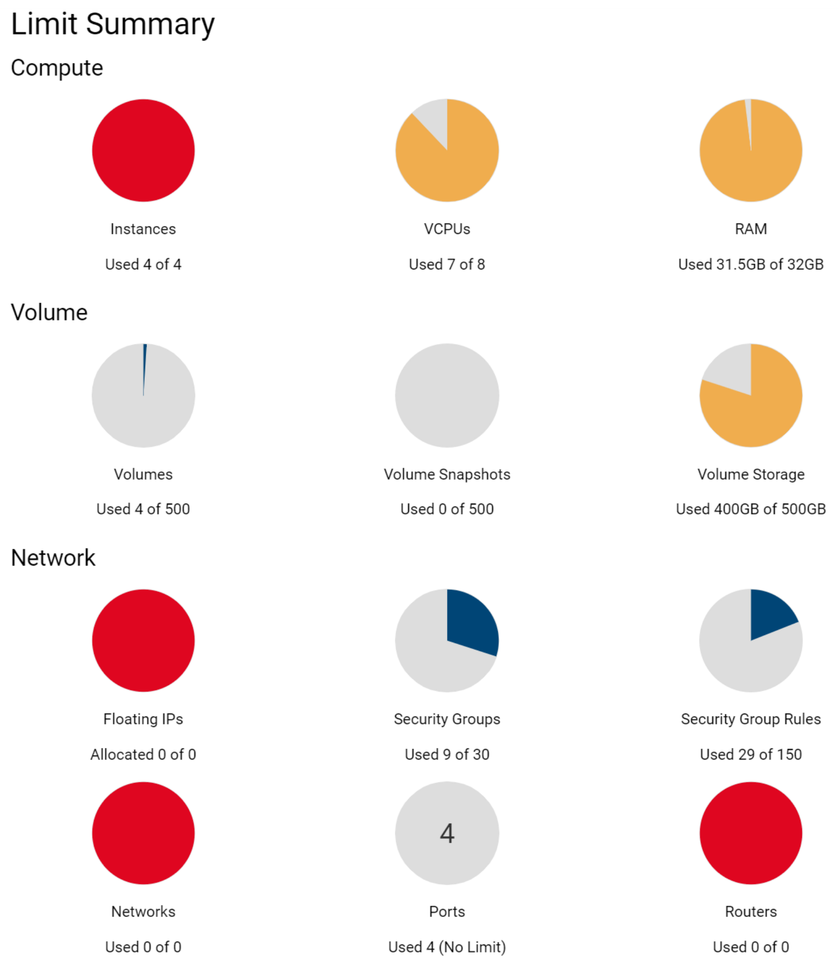


This is an illustration of our system architecture. We deployed four instances on the MRC. A docker container with a CouchDB is deployed on each of the four instances. A CouchDB cluster is set up on these four nodes with *instance-1* as the masternode. We have a twitter harvester to collect data using Twitter API, parse the data received and save them to the database. There is also an analyzer to retrieve the data needed from the database. The frontend Flask server would get data from the analyzer and display the data to the web page.
All the applications are deployed on the remote instances, but this figure is simplified for illustration purposes.

2.2 Melbourne Research Cloud (MRC)

The Melbourne Research Cloud is an on-demand computing resource provided to researchers at the University of Melbourne. It provides similar functionality to commercial cloud providers such as Amazon Web Services, Microsoft Azure and Google Cloud Platform.(Documentation, 2021) The MRC provides Infrastructure-as-a-Service(IaaS) cloud computing services to the researchers, as well as access to a robust set of on-demand virtualized computing resources (Server,RAM, storage etc). This service allows users to access scalable computational power with no cost.

2.2.1 Resource Allocation


Limit Summary

Compute

Instances
Used 4 of 4

VCPUs
Used 7 of 8

RAM
Used 31.5GB of 32GB

Volume

Volumes
Used 4 of 500

Volume Snapshots
Used 0 of 500

Volume Storage
Used 400GB of 500GB

Network

Floating IPs
Allocated 0 of 0

Security Groups
Used 9 of 30

Security Group Rules
Used 29 of 150

4
Ports
Used 4 (No Limit)

Networks
Used 0 of 0

Routers
Used 0 of 0

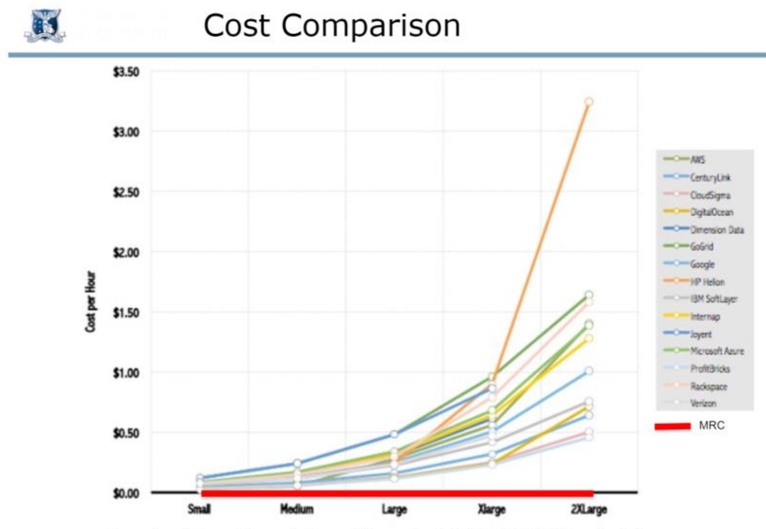
For this project we are allowed to create 4 servers(instances) with 8 virtual CPUs and 32 GB RAM on the Melbourne Research Cloud. Furthermore, we are given a volume storage capacity of 500Gb and full control in the security groups. The team chose to use 4 instances with 3 of them using 2 vCPUs and 9 Gb RAM and 1 using 1 VCPU with 4.5 Gb RAM. Each instance is allocated with 100 Gb of the volume storage.

2.2.2 Advantages of using MRC

1. Security

One of the advantages of using MRC is that it can ensure high level security. As is known to all, the security problem is always an important concern in the IT industry. If the infrastructure of our system is not safe, some malicious operations might be done by some attacker. The MRC not only provides users with data protection stated in their official website but also can allow users to customize their own safety countermeasures. For example, users can create their own key-pair to log into the cloud instance. Besides, users need to set security groups to determine which ports are open to the public.

2. Costs



See also https://www.infoworld.com/article/3237566/cloud-pricing-comparison-aws-vs-azure-vs-google-vs-ibm.html

The cost is one major advantage for using MRC. While using MRC,we don't need to worry about the resources and the load balance as well as the cost of getting the resources. Moreover, compared to public commercial cloud such as Google Cloud or AWS, MRC provides a scalable service to researchers with no usage cost for the cloud services.

3. Control and Customization

Using MRC, we have control over the OS, middleware and the runtime environment as running cloud services. Furthermore, MRC allows the group to manage security group details as well. During the development, we can control the sharing and collaboration, to avoid potential failure, snapshots and backup services are also available

4. Availability & Convenience

MRC has 24/7 accessibility. Users are able to connect to the MRC via VPN to the University of Melbourne regardless where they actually are. MRC provides detailed tutorials for all OS users and a guide for accessing the services. Deployment can be accessed using Ansible script without manual adjustment. Snapshot volume can recover potential failures.

5. Scalability

With MRC, users can customize a lot of functions. First of all, users can create and manage their own scalability environments by allocating resources on the cloud. What's more, users can install any applications on the docker and use ansible to automatically deploy the environment.

2.2.3 Disadvantages of using MRC

1. Background Knowledge and Management Cost

As MRC uses Infrastructure as a Service, users need to build their own cloud infrastructure before deploying their software or web page. Which requires lots of extra knowledge. If the user is unfamiliar with setting up the OS, a potential management cost may be needed to maintain the server.

2. Availability

Since MRC is not a commercial cloud, the availability of the services may not be as strong as commercial products like AWS and Microsoft Azure. Due to the limitation of budget and the design of the user group, MRC can provide relatively cheap services with basic availability. During the project, the team found that the cloud was sometimes not stable for connection and queries. Also, if anytime a hardware failure occurs, the user has no access to the physical data centre thus not able to fix them immediately. Hardware failure may cause server down for a long time while we report the issue and wait for a fixture.

3. Security

The login of the service is based on the unimelb student account, which may not be safe at all. Given that the student id password can be changed with student id, name and birthday provided, there could be a leak of password. Since the user has full access to the instances and the security group, the attacker can even delete the instances in the group.

2.2.4 MRC Summary

The above sections are the advantages and disadvantages of MRC. Although it does have some drawbacks, its benefits are far outweighed by its disadvantages. In short, the MRC ensures users with on-demand computing resources and other useful functions.

2.3 Front-end and Back-end Component

The front-end receives Json data through Asynchronous JavaScript and XML (Ajax) technique and the data is displayed on the website page by Apache Echarts library. The communication between the website pages and the database is implemented by the Flask framework.

2.3.1 Ajax

Ajax is a set of client-side web development technology that integrates multiple technologies. the front-end requests to call the function under the specified route of the back-end through the Ajax function request and return the required data. The main reason that we choose this technique is that it can maintain data without updating the entire page when our requirement is to get data for a part of our page. This allows Web applications to respond to user actions more quickly and avoid sending unaltered information on the network. In this project, we need to change the scenarios to load new analysis figures, and we do not need to reload the rest of the page besides the figure that we want to update. Also, Ajax does not require any browser plug-ins, which means that we do not need to worry about the dependencies on the different platforms.

2.3.2 Apache Echarts

Apache ECharts is a free, powerful charting and visualization library offering an easy way of adding intuitive, interactive, and highly customizable charts on our pages. Firstly, it is convenient that users can easily interact with the result figures by showing the detail of the results. When users move the computer cursor on the figures, it will pop up the meaning of that point or zone. With this lightweight library, we try to display different types of dynamic diagrams, including line charts, bar charts, pie charts, word cloud charts and maps. This library can even help us to implement a timeline function to make our figures change over time. Secondly, there are many sub-level libraries in this open-source library, and it is friendly for us who do not have adequate developing experience to perform animation of the results. However, this also limits our innovation since we can only achieve the functionalities that we want to show based on the existing functions in the library.

2.3.3 Flask

Flask is a lightweight WSGI web application framework which can be implemented in Python. It is useful for those of us who are familiar with Python to get started quickly. Under the framework of Flask, the back-end can quickly receive the query from the backend and request data from the databases. When the back-end receives the requested data, it will send them to the front-end in Json type. The main advantage of Flask is that it is a lightweight and modular design, so we can easily convert it into the web framework you need. Then, it is simpler to deploy than Django which is a traditional web application framework. Because of the elegant interface of API, there are fewer barriers for communication between front-end, back-end and database. Although Flask's community is still small compared to Django, we can still easily find answers to the problems when encountering difficulties.

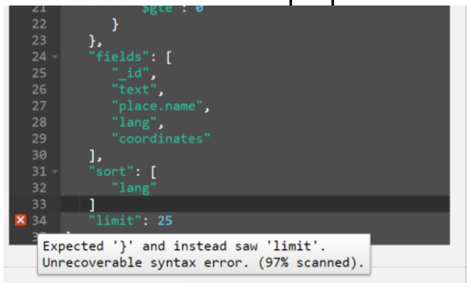
2.4 Database Component

2.4.1 CouchDB

CouchDB is an open source NoSQL document database that collects and stores data in JSON-based formats. Unlike relational databases(eg. mySQL), CouchDB uses a schema-free data model, which simplifies record management across various computing platforms(Education, 2021). The ReSTful API used by CouchDB for accessing data can easily be visualized by the team using either mango query or MapReduce views.

2.4.2 CouchDB pros

1. Since data is stored in document based, Only the file structure needs to be maintained. Before storing the data, there is no need to create schemas or fields, as long as the format is JSON-based, the database can simply save them.
2. CouchDB provides convenient HTTP APIs for visualizing dataset with code testing functionality. The mango query for GET data and the map and reduce function can be tested in the API with proper error message output and explanations (eg. syntax errors).



```
            "$gte": 0
        }
    },
    "fields": [
        "_id",
        "text",
        "place.name",
        "lang",
        "coordinates"
    ],
    "sort": [
        "lang"
    ]
    "limit": 25
```

```
Expected '}' and instead saw 'limit'.
Unrecoverable syntax error. (97% scanned).
```

3. The built-in JavaScript Query Server allows the team to run loops and statements with external constants and variables. When we design the Map function we are able to call all basic functions in JavaScript.
4. The built-in reduce functions are implemented in Erlang and run inside CouchDB which can be much faster than equivalent JavaScript functions.
5. CouchDB can be conveniently set up and used with cluster and GUI. For data accessing, back-end applications only need an appropriate port number with HTTP calls to the node IPaddress, all clusters would work properly.

6. All errors are sent back with JSON format to the applications that are easily readable. The developers have full control over the file conflicts resolutions and the team can resolve them properly.
7. Unique index given to all documents when it's put into the database, Since all indexes “_id” are managed by the system, we can retrieve with the index efficiently. Also, the generated revision number allows us to update certain documents with records which simplifies the work.

2.4.3 CouchDB cons

1. Supported documents for couchDB are hard to find. Mango query does not come with lots of examples and mango query is not a popular SQL language in the industry.
2. Documents database does not check the validation of fields, when applying map function to the database, often we get no document found without any error message. Creating views of the database in Python required the team to write JavaScript in Python IDE which is not sufficient as well.
3. CouchDB only checks the validation of the file but not the content. In the project we have to deal with duplicated tweets by writing external scripts (Explanation in 5.1.1).
4. The Python couchDB is not supporting temporary views anymore.
5. Unlike RDBMS, query needs to read every document which makes the process very expensive compared to SQL databases.
6. The Python package of CouchDB is out of support, there are mistakes and lack of examples in the documentation.
7. Reduce function is restricted by the built-in methods. When the team is trying to modify it to a form of what we need, there are bugs such as “null” occurring when grouping the documents. Also, there is not much support for this function on the internet.

2.5 Containerization

In this section, we will talk about the advantages of containerization, and specifically, docker container, to show the reason why we choose to use this technique and how it is used in this project.
A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. Docker container can be lightweight, standalone, and would include everything needed to run an application. The reason to use docker container is clear. When deploying applications, containerization makes it easier for programmers to organize all the packages and dependencies, and the potential different results running on different infrastructure. This ensures that all the software will work uniformly despite the difference between instances.
The reason why we choose docker is also quite simple. First, it is a recommendation from the specification and it is relatively simple to learn. The documentation for docker is very complete and organized. There are also a lot of online tutorials and resources for learning docker.

3. Data Delivery

In this section, The process of data collection and preprocessing would be carried out. The current system collects and analyses data from AURIN(Australia's Spatial Intelligence Network) and Twitter.

3.1 AURIN Data Collection

To achieve comparsion of data from twitter and government websites, the team imports data about the level of unemployment and language spoken at home from AURIN via the AURIN portal(https://portal.aurin.org.au/). This data originally belongs to Australian Department of Education and the Australia Bureau of Statistics. AURIN has 5217 data records in Research Data Australia, which involve 97 different groups("Research Data Australia", 2021). Using the portal the team is available to select data easily using the data browser and export it into the JSON form which can be stored in the CouchDB database. The data collected from ARUIN are first downloaded to local, clean and preprocessed using Python and stored into the cloud database on MRC which can deliver to all users via front-end web page. Since AURIN collects data from different organizations, the format of files is likely to be different in each dataset, therefore the team needs to process them manually.

3.2 Twitter Data Collection

For this project, we have created four instances which are used to collect twitter data parallely and simultaneously. Four access tokens are used to retrieve tweets from Twitter. After receiving the newly obtained tweets, those tweets will be preprocessed and sent to the couchDB for permanent storage and later analysis.

3.3 Twitter Data Pre-processing and Analysis

After receiving the tweets from Twitter, we preprocess each tweet to keep some of the fields. This modification will lead to improved efficiency and less resource usage. First of all, tweets are filtered by the information that's relevant in the project, such as text, sending place, source(device), language using and the unique identifier, these information are combined with the sentimental analysis by TextBlob and stored into the CouchDB database.

In this project, we are aiming for the data from only 8 greater cities across Australia, which are Adelaide, Brisbane, Canberra, Gold Coast, Melbourne, Newcastle, Perth and Sydney. As each tweet contains the place information in the city level, A simple filter is taken while retrieving the views. For the result to be pin out on the map, we manually store the central location of these chosen cities and combine them with our dataset from CouchDB.

The tweets are filtered by five different scenarios which are language, source, time, vaccine and unemployment related. Language, source and time are originally contained in all tweets which can be easily grouped out using MapReduce built-in views in the database. To get the related tweets for vaccine and unemployment rate, the team decided to use a relevant word dictionary. All tweets that contain words in the dictionary are considered to be relevant to the given topics. As the team wants to study if there is any relationship between these five scenarios and the polarity of tweets, we set up the MapReduce function for calculating the polarity and subjectivity of the tweets in different scenarios. Map functions are designed to contain a key array including the name of the city, date of sending and the related keyword (different language in the language study, mentioned vaccine related keyword in vaccine study etc.). The team has designed a custom Reduce function which can sum the polarity, count the number and get the average polarity. Data is then passed to the backend Flask and then to the visualization in frontend. Detailed explanation of the scenarios will be carried out on section 4.4.

3.4 Data Storage in CouchDB

In the system, the team has decided to format all relevant data into the database with proper analysis by re-producing data for different scenarios' results using both Python programming and MapReduce views.

3.4.1 MapReduce (Views)

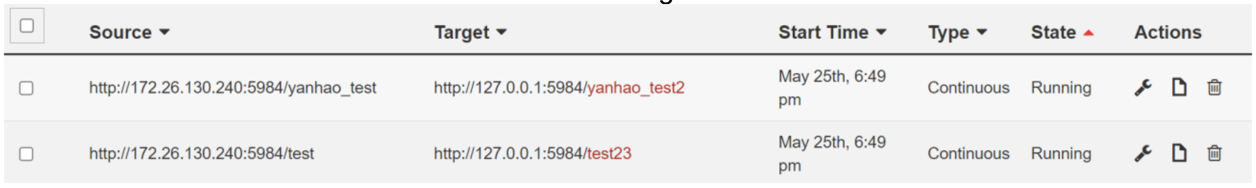
CouchDB provides a default MapReduce functionality. By creating and using views, we can filter the tweet data with modification such as count or simple calculations.In this project we've created multiple views in the no duplicate dataset, each view can extract certain data fields that the team is interested in for different scenarios. The backend application in the Python program will use the couchdb Python package to curl data and transform it into a required JSON format by the front-end.

3.4.2 CouchDB Load Balance

The team has deployed the cluster database on all four instances on MRC. In this design, we can access any of the nodes with consistent data output. Therefore, we are able to utilise the processing power in each node while querying specific information for the web page visualization. To achieve this, we separated the requests in each page on the web application into different servers while querying the data by using different host addresses of CouchDB nodes. By doing this, the speed of retrieving data has increased and the retrieving time for the graphs in the page is significantly improved.

3.4.3 Continuous Back Up （Replication）

To avoid potential failure on the instances that may cause loss of information, the CouchDB database with all tweets' information is set to be continuously backed up with a local database on one of our team member's PC. With this information, we can rebuild the CouchDB server with consistent results even if all the instances are being reset.



☐	Source	Target	Start Time	Type	State	Actions
☐	http://172.26.130.240:5984/yanhao_test	http://127.0.0.1:5984/yanhao_test2	May 25th, 6:49 pm	Continuous	Running	
☐	http://172.26.130.240:5984/test	http://127.0.0.1:5984/test23	May 25th, 6:49 pm	Continuous	Running	

4. System Functionality:

This section of our report will illustrate the function of our system by clearly discussing 5 scenarios included and clarify the reason for choosing this scenario, each scenario will be supported with the plots and result we got on it.

4.1 Functionalities Achieved:

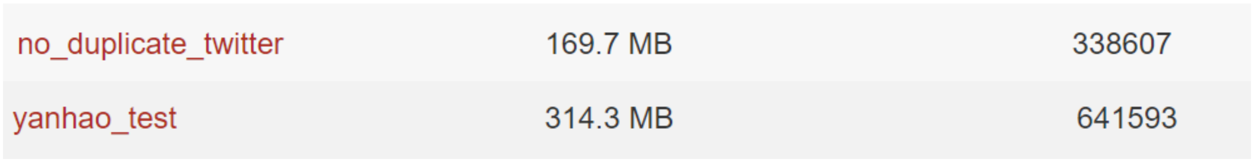
1. Develop Cloud infrastructure for supporting the following autonomy and data analysis
2. Automatically harvest twitter data from social media
3. Automatically deploy the CouchDB for data storage and analysis
4. Automatically preprocess data before embedding them to couchDB
5. Use Mapreduce function in CouchDB to do analysis for 5 different scenarios and compare the result to data on Aurin
6. Display our analyzed result on front end with graphs and map

4.2 Scenarios for analysis

1. How do different language tweets correlate to language spoken at home?
2. What is the difference between iPhone and Android users towards twittering?
3. What is the pattern of people accessing tweets in a 24 hours period?
4. What are the trends of attention and attitudes of people towards Covid vaccines?
5. What are the trends of attention and attitudes of people towards unemployment?
6. What is the correlation of people talking about unemployment on tweet and the unemployment and youth unemployment rate?

4.3 Data and Region

4.3.1 Data:

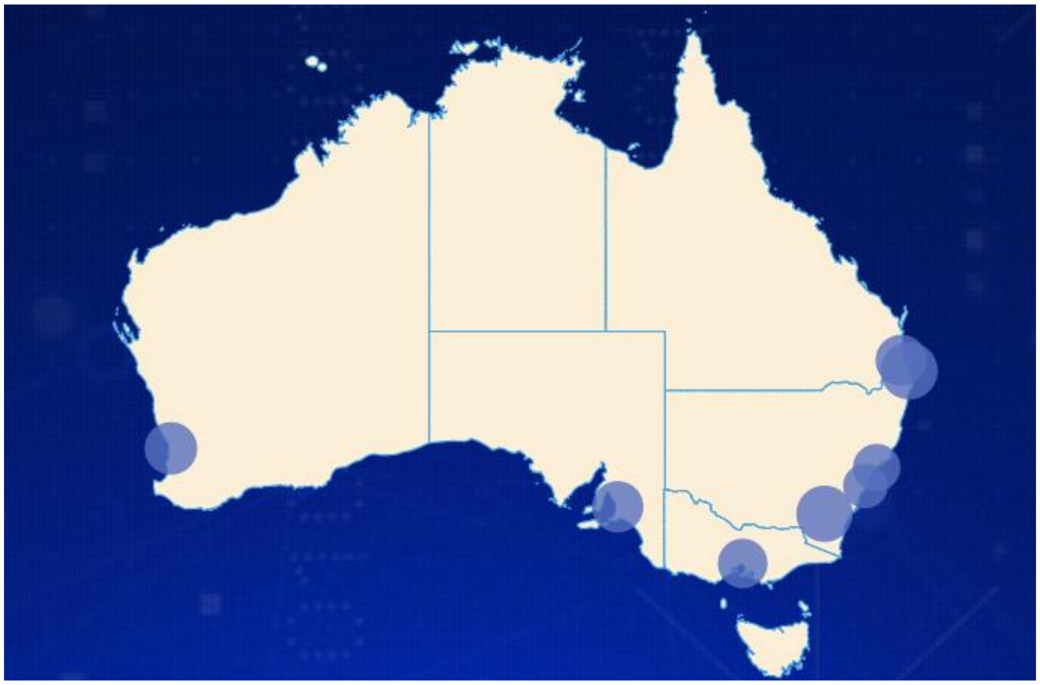


no_duplicate_twitter	169.7 MB	338607
yanhao_test	314.3 MB	641593

As shown in figure, the total number of twitter we collected is 641593; the number of twitter left after deduplication is 338607.

4.3.2 Focused Region:

In most of the above scenarios, our analysis will consider its correlation to the region in map. The region we considered includes 'Sydney', ‘Melbourne', 'Brisbane', 'Perth', 'Adelaide', ’Gold Coast', ' Canberra' and 'Newcastle’ (spotted in graph below), which are the top 8 largest cities in Australia. Among all twitter data collected, these 8 cities are also the top cities in twitter count with more than 2000 twitters each. This minimum twitter amount requirement for data analysis is to deteriorate the impact of outlier and noise in our collected data.



4.4 Scenarios

4.4.1 Sentiment score of twit vs Various languages

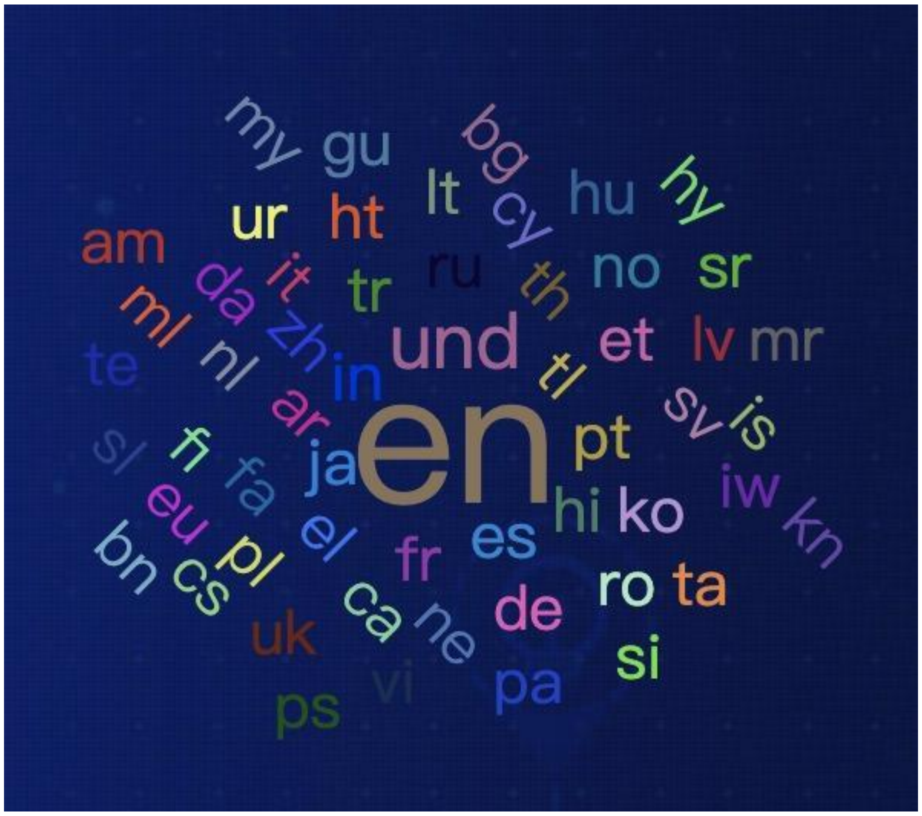
4.4.1.1 Topic Focus

This topic focuses on the relationship between language and the polarity of related tweets. The plan reveals the polarity of people who speak different languages in Australia's 8 largest cities and the overall polarity of people who speak different languages throughout Australia

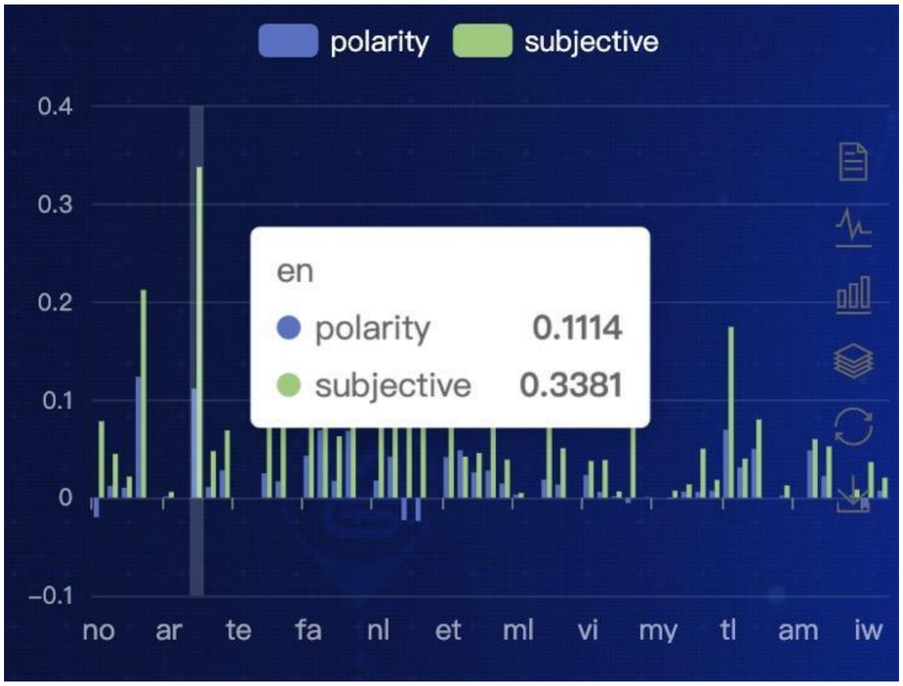
4.4.1.2 Reason to Choose

The time period for the outbreak of the new crown virus varies around the world. Twitter users are spread all over the world and people speak different languages. Collecting their tweets and calculating their polarity based on their tweets, we can see the differences in the attitude and mood of people facing the epidemic in different countries.

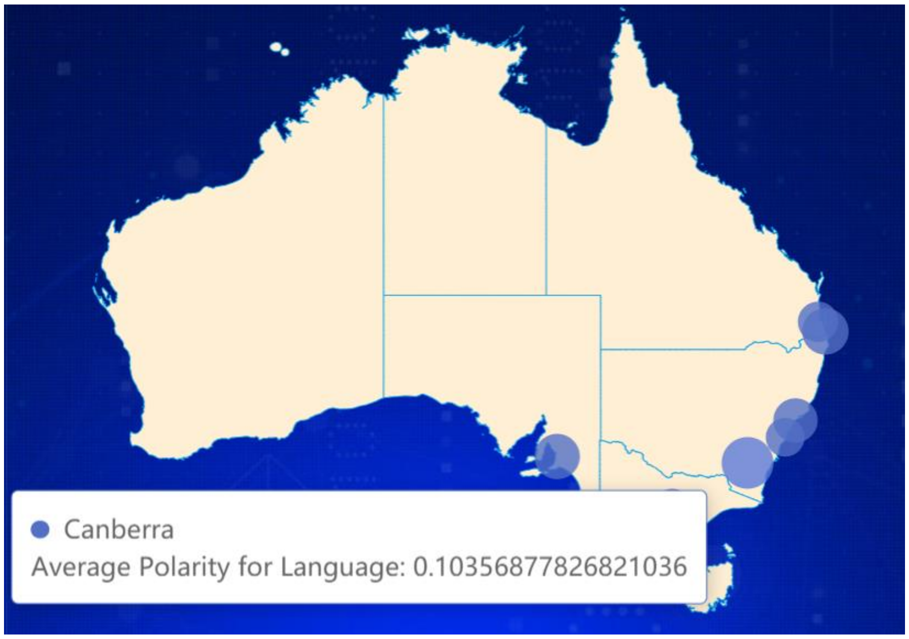
4.4.1.3 Comprehensive analysis



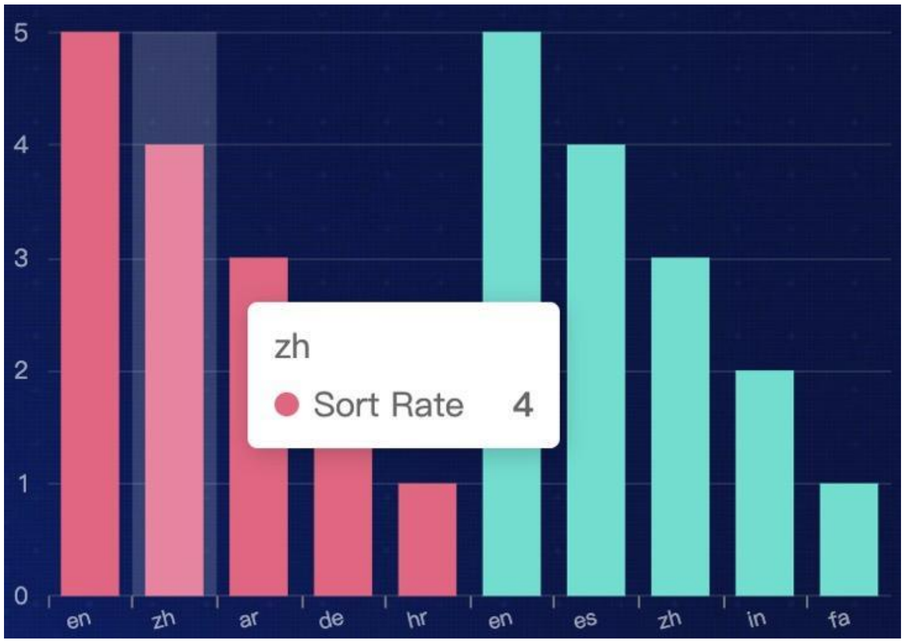
We first counted the number of people who speak various languages on all tweets we fetched, and made a word cloud map. As is shown in the picture, The number of English-speaking Twitter users is the largest among all the tweets we get.



Because the twitter data are unbalanced, for example, there are relatively more English-speaking Twitter users, so the average score is calculated in the analysis. By calculating the average score which divides the total polarity of people who speak this language by the number of people who speak this language, twitter data reflects that the average polarity of English speakers is the best.



We calculate the sum of the polarity of speaking various languages in this city divided by the number of people speaking various languages in this place. We can see that people in Canberra and Gold Coast achieve the highest polarity score. Compared with other cities, the pace of life in these two cities is relatively slower and life will be relatively comfortable, so it can be reflected in twitter.



We also count the top five of the total number of people who speak a language in each city from the tweets and compare it with the aurin data. We can clearly see that the aurin data and the results reflected by twitter are different. It may be affected by the epidemic and the policies of each country are different. Many people have left Australia, so the data will change. Compared with the aurin data, we found that some countries have very few people in Australia, but they like to tweet and vice versa.

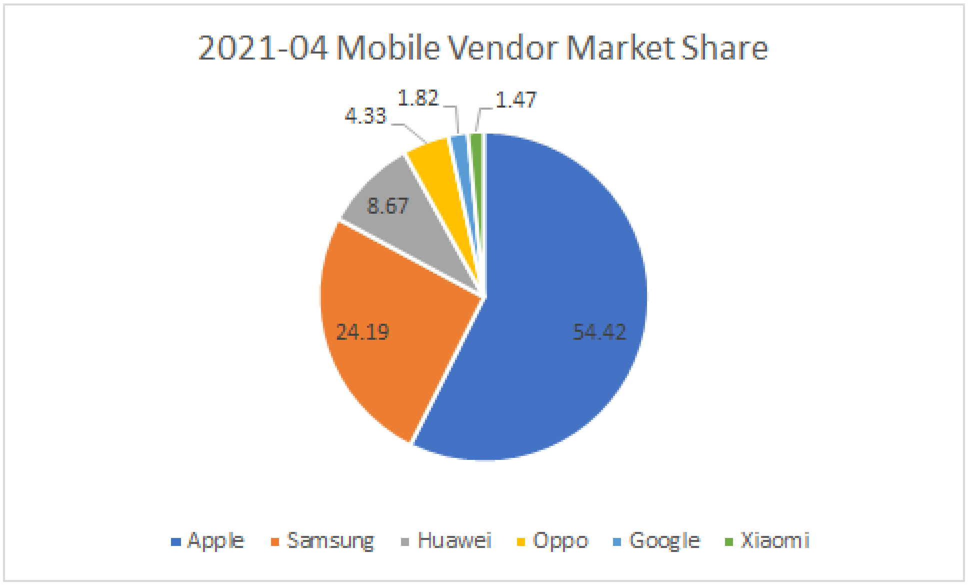
4.4.3 Difference between iPhone and Android users towards twittering

4.4.3.1 Topic focus

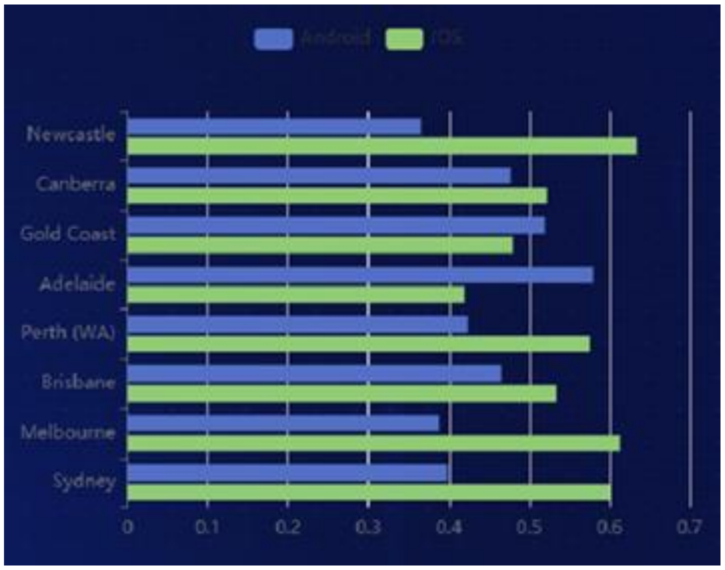
This topic is focusing on analyzing the difference in social media attachment between the 2 largest groups of source users iPhone and Android by using the counts and sentiment score of tweets and comparing them with respect to time, region.

4.4.3.2 Reason to choose

The debate between Android and iPhone (2 largest mobile vendors) will never stop. In this scenario, we are aiming to examine the difference between those 2 group users towards social media (tweet). By using the Mapreduce function on Couchdb, we could easily group data by their source. The count and average sentiment score could be computed directly from the reduce function. According to the data below, we could see that the iPhone takes 54.42% of the mobile market and thus Android should take the rest 45.58% approximately. By assuming the direct positive correlation of the number of mobile owners to the number of tweets, we could infer the proportion of users in each city by count. Relating to time, we could obtain the pattern for iPhone and Android users to access twitter in a day. With the sentiment score, we could see the optimism of people using iPhone or Android when posting tweets. This may reflect the happiness of users for iPhone and Android in each city.

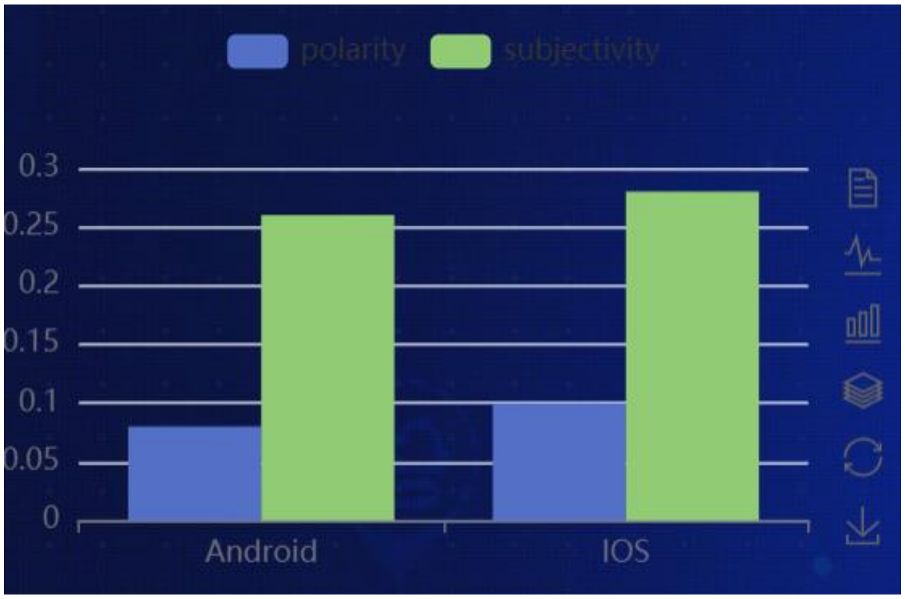


4.4.3.3 Cities Preference over iPhone and Android

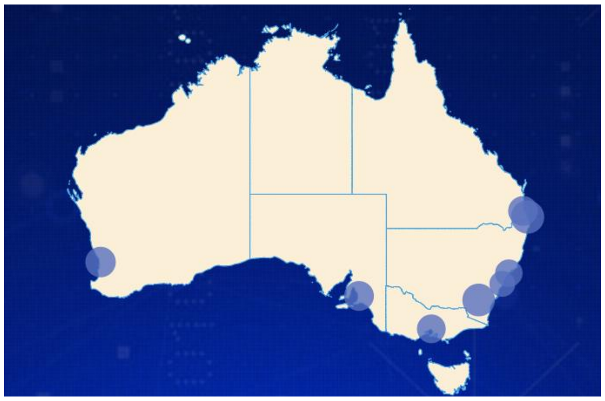


First of all, our system will give a plot to show the proportion of tweets sent by iPhone source or Android source for each city. By assuming the similar sending pattern for Android and iPhone users, we could use this data to infer the preference of people in that region towards those 2 brands of mobiles. For Newcastle, Sydney, Melbourne and Perth (WA), iPhone defeats Android to be the one dominating the tweets. For Canberra and Brisbane,the number of tweets from iPhone is also slightly higher than for Android. However, in Adelaide, the preference for Android takes over the dominant position in the market; similar to Adelaide, Gold Coast also enjoys a higher proportion of tweets from Android users. Overall, Android and iPhone users constitute more than 90% of tweets in each city.

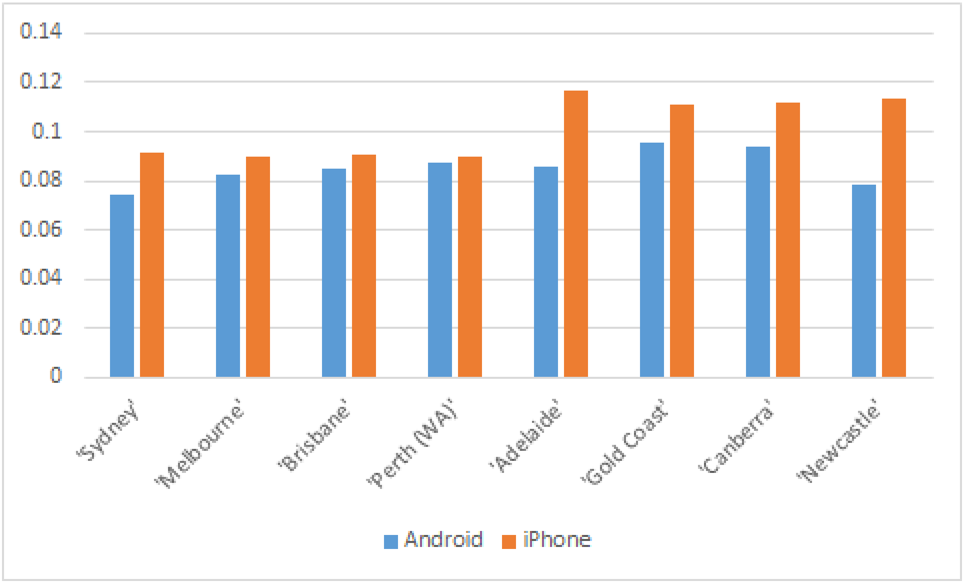
4.4.3.4 Difference in Sentiment Score for iPhone and Android users (iPhone wins)hi



The system will provide an overall analysis to the sentiment score of iPhone and Android tweets. Based on the plot, we could observe that the iPhone slightly outperforms Android in both polarity and subjectivity. The following graph gives a more detailed analysis to the performance in each region.

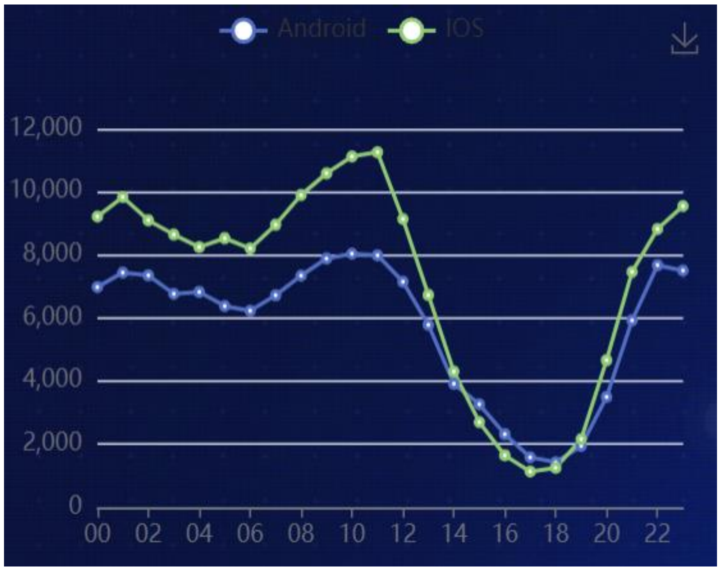
	
Android	iPhone

By selecting spotted points on a graph, we could see the average sentiment of that city for Android or iPhone. If we compare the average sentiment score by city, we could find that for all 8 largest cities in Australia the average sentiment score for tweets sent by iPhone users is higher than it for Android users.



To make it more readable for comparison (the same information could also infer from our map). The happiest people among all are the iPhone users in Adelaide, whereas the most unhappy people are the Android users in Sydney. The largest difference between iPhone and Android users is in Newcastle. Except for Sydney, the top 4 largest cities have relatively lower happiness difference between iPhone and Android users than the 4th to 8th largest city in Australia. Overall, according to our analysis, iPhone users tend to be more optimistic than Android users in posting tweets online. It just shows some correlation between source and happiness, but not suggesting the causality of the happiness to source. People should be careful when using this relationship.

4.4.3.5 Difference in time spending on twitter for iPhone and Android



The final analysis given in this part corresponds to the analysis in time spending pattern of iPhone and Android users. Based on our plot, the times for iPhone and Android users seems trending simultaneously. The correlation between the number of tweets for each over between 2 sources is 0.988. The only variation in trend is the relatively fewer afternoon tweets from iPhone, which may imply the iPhone users are less likely to post tweets in the afternoon (from 11:00 to 20:00).

Overall, there is no evidence showing that iPhone and Android users have different time patterns in accessing social media.

4.4.4 Twitter analysis in accordance with Time

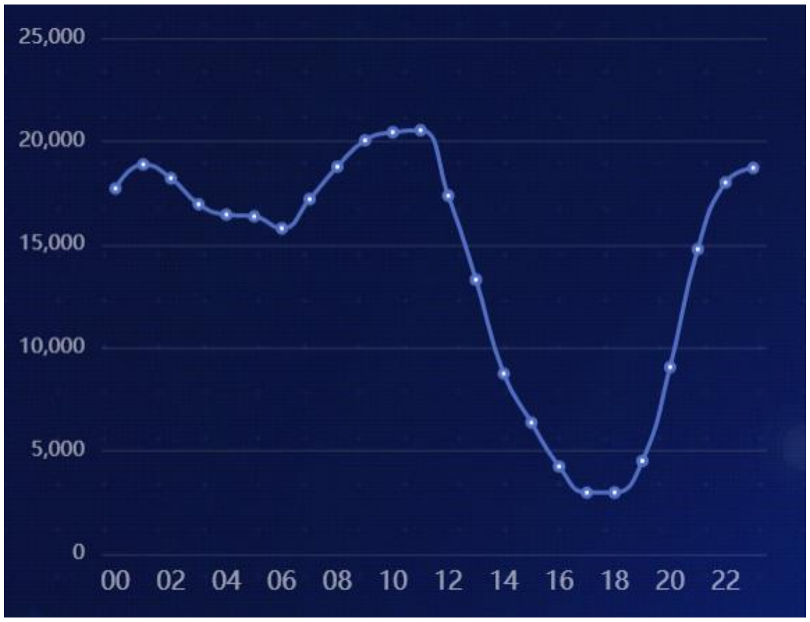
4.4.4.1 Topic focus

This topic focuses on analyzing the trends and pattern of people in sending tweets in count and sentiment score of tweeters over 24 hours for eight largest cities and for all collected data.

4.4.4.2 Reason to choose

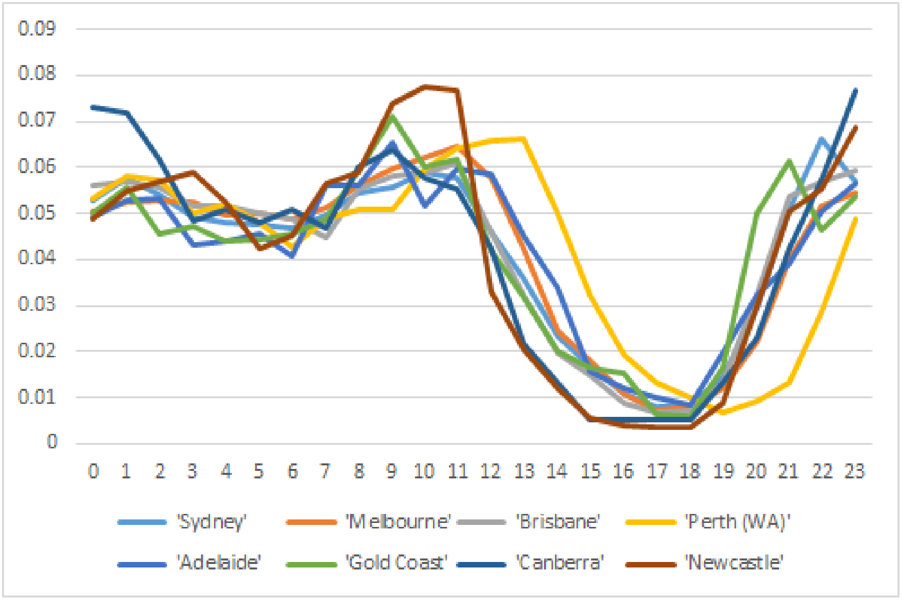
In this section, we would like to examine the pattern of people posting tweets over a continuous variable time. By counting the number of tweets in each hour, we could see the tendency of people accessing twitter. The low tendency may mean people are busy with other stuff or falling asleep. In this way, we may have a guess on the time people put down their smartphones at night. Sleep is essential to people's health. “Report to the sleep health foundation 2016 sleep health survey of Australian adults” (Adams, R et. al,2016) suggests that not enough sleep will lead to low performance at work. According to the study, teenagers will have a longer sleeping time if they stay away from smart-phones before going to bed (Bartel, K, Richardson, C & Gradisar M,2018). Besides this, our study will also include the sentiment analysis associated with tweets which indicates change in happiness over the day.

4.4.4.3 Count twit vs Time

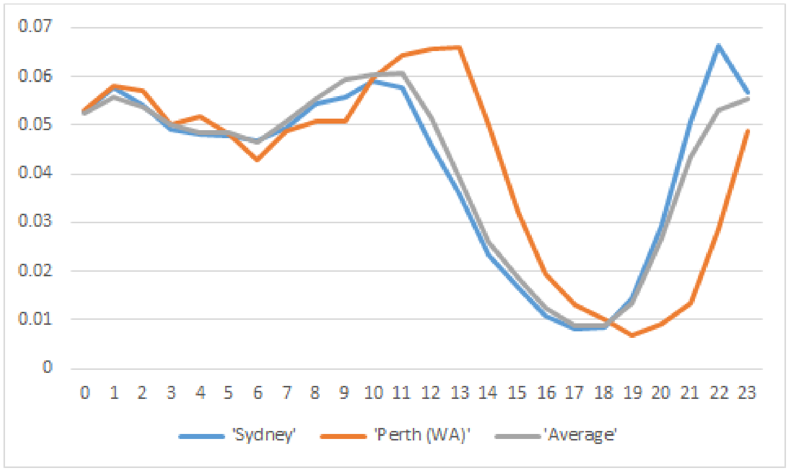


From the above graph we could see most of the tweets are concentrated in the morning and night. The U shaped part of the graph shows that people have a really low tendency to post tweets from 13:00 to 20:00. This may be explained by the fewer accessibility to smart phones due to work and study.

From the graph, we could observe 2 drops in the number of tweets posted at night. One is 23:00 to 0:00, the other is 1:00 to 2:00 and the decreasing trend continues after 2. These 2 drops the 2 time spot when most people leave social media and go to bed. Normally speaking, the higher the drop from 23:00 to 0:00, the healthier the sleeping habit of people. In the following plot, we could examine the pattern in the number of tweets for 8 cities we focused on. (This plot is different from what we showed in the front end. We get 24 plots to represent the hourly proportion instead. This plot is the summary of those 24 plots):

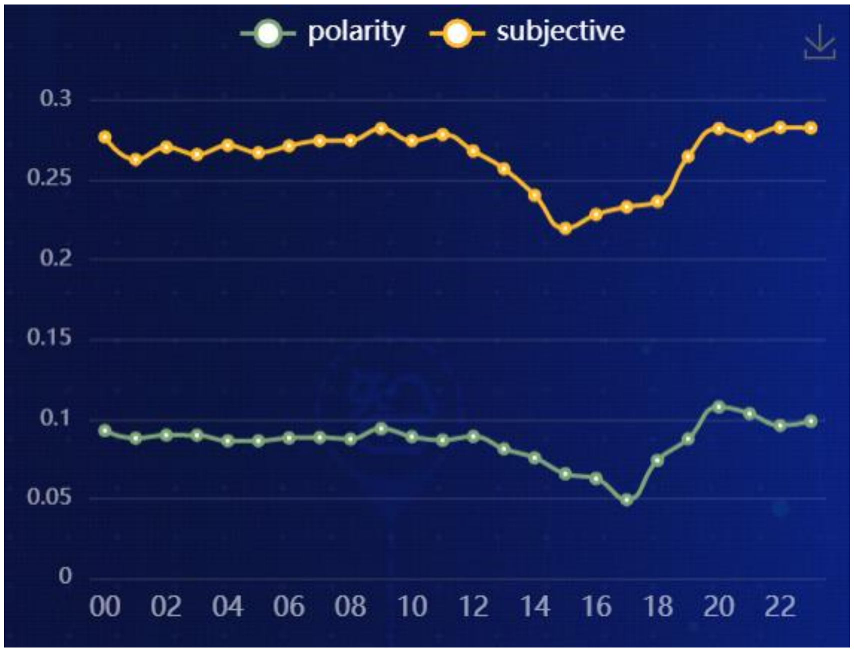


Based on our plot, we could see that most of them have a similar pattern as the graph for the whole of Australia. The y_axis of the plot is the ratio of tweets collected in that hour to the total number of tweets collected in that place. After testing the correlation of other cities to Sydney, we could find that the correlation value for most cities is above 0.9. The only city which has a less than 0.7 correlation to Sydney is Perth.



According to the plot, U_shape in Perth starts at 13:00, which is about 2 hours later compared with other cities. This is because the time zone in Perth is (GMT+8), which has a 2 hours time difference compared to other cities. If we remove these 2 hour differences, we could observe a similar pattern for all 8 cities. Therefore, we could conclude that there is no significant difference in the pattern of sending tweets over the 8 largest cities.

4.4.4.4 Sentiment Score vs Time



First, the graph shows a high correlation between the trends of polarity and subjectiveness of tweets over the day. The polarity line shows the trend in happiness. The happiness of people stays relatively stable in the morning. However, after 12:00, we could observe a clear decreasing trend in the optimism of tweets. However, the happiness of people bounced back after 17:00 and hit the highest at 20:00. Combine with the count plot vs time. We could see that the tweets in the afternoon are both fewer in counts and not happy compared with tweets arriving other times in the day.

Our system also provides plots showing the happiness of people in 8 largest cities individually. We will not discuss too much about this here.

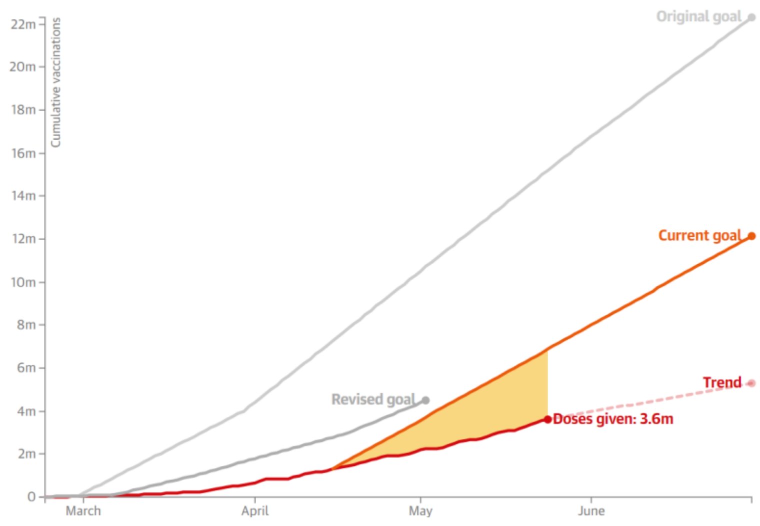
4.4.5 The trend of topics and attitudes towards COVID-19 vaccine

4.4.5.1 Topic focus

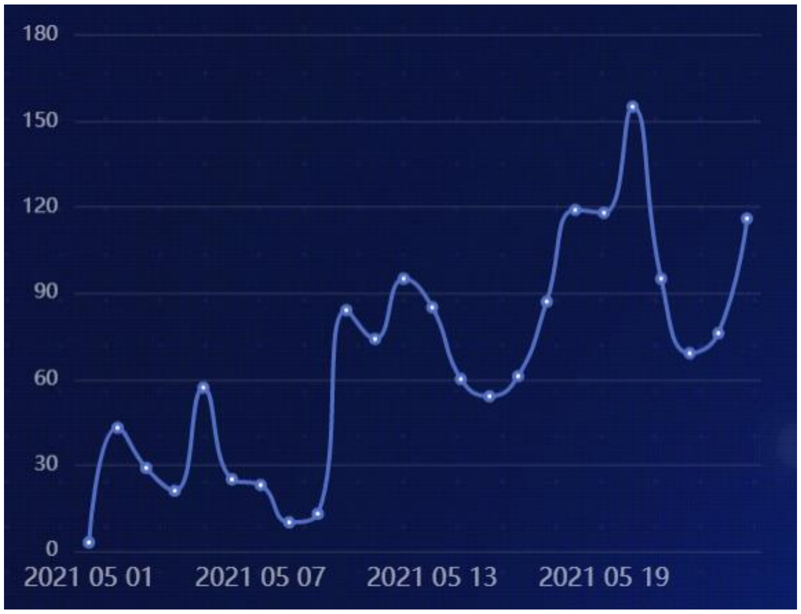
This topic is focusing on explaining trending attitudes of people’s tweets towards COVID-19 vaccine. Our system will provide analysis on both the number of tweets harvested and the sentiment score of them over a 24 days period (from 1st of May to 24th of May). Our system will also show the correlation of it towards the “Household Impacts of COVID-19 Survey” constructed in April 2021.

4.4.5.2 Reason to choose

Covid-19 vaccine has already been in plan since late February. However, the dosage taken is way behind the original goal and current goal. Our system uses the Map function filtering out the COVID_19 vaccine related tweets, and the Reduce function summarizing them. With this, we could investigate trends in the focus of the topic and the attitudes towards it by using harvested data. With this kind of information, it may help to predict the doses given trend in the future.



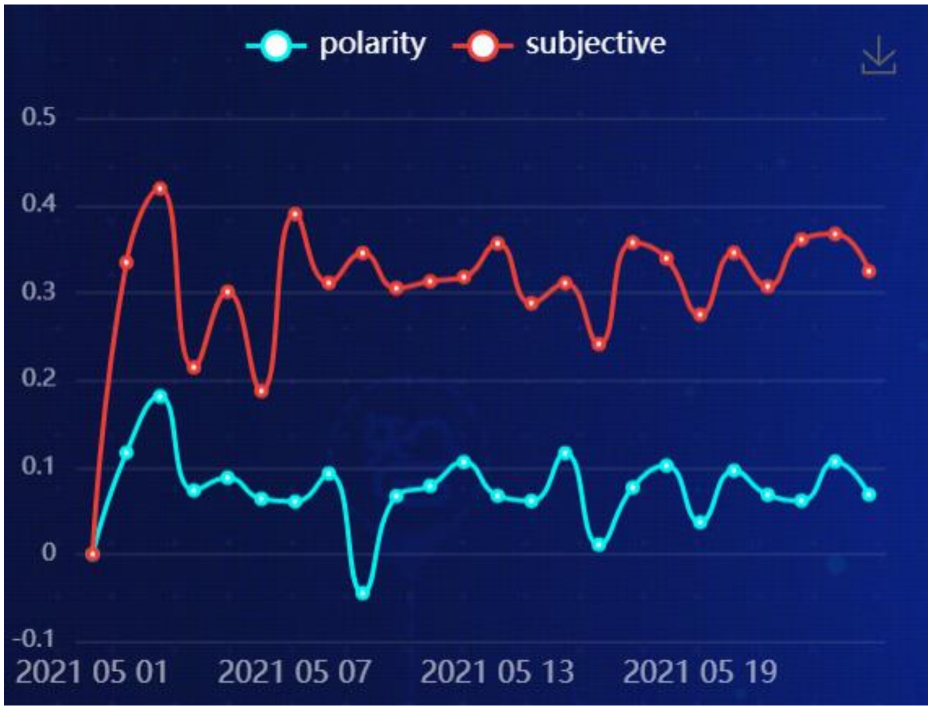
4.4.5.3 Attention to Covid-19 vaccine related topics.



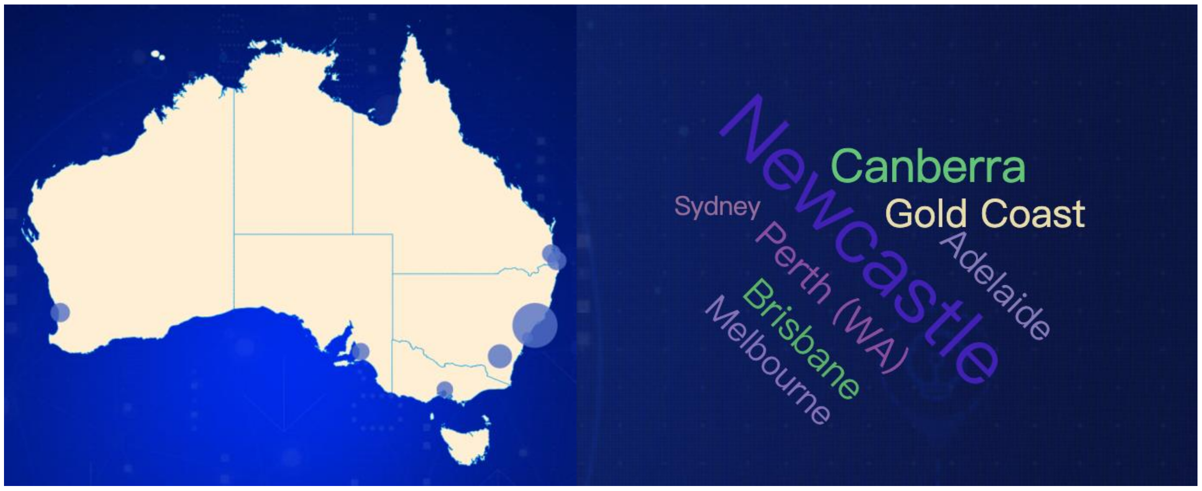
Generally, we use the number of tweets relating to this topic representing the attention of people towards this topic. Based on the plot we could observe cycles of people' attention towards the topic. From the graph, we could see a gap between the number of tweets found before and after 9th of May. On 10th May 2021, the flight from India resumes which makes Australia under threats of covid again. After that, the number of identified cases rises all over the place. According to this, we could see that the number of tweets hikes up when people feel that they are more endangered.

However, there are also cyclics showing that the attention to the topic strikes when new updates come out, but it would fade away in 1 or 2 days.

4.4.5.4 Attitudes to Covid-19 vaccine related topic.



According to our plot, the sentiment score of tweets toward Covid vaccine is relatively stable. Though it fluctuates a bit over time, it will normally be above 0. The only day which shows a negative polarity rate within the inspection period is 9th May. On 9th of May, Morrison announced that the travel restriction will still be strict for a considerable period of time. It is similar to saying that though people are vaccine protected, they are still not allowed for international travel. However, the attitude towards covid vaccine goes back quickly when people know that the flights from India resume. Overall we could get a slightly positive correlation between the average sentiment score of people and the attention of the topic. As mentioned in earlier cases, people's attention to this topic booms when 'bad' news updates. The attention to the topic shows the concern of people towards COVID_19. This positive correlation explains when new cases are identified or there are potentials for new cases to be discovered, people are more willing to be protected by vaccines; they tend to be more positive towards the vaccine.

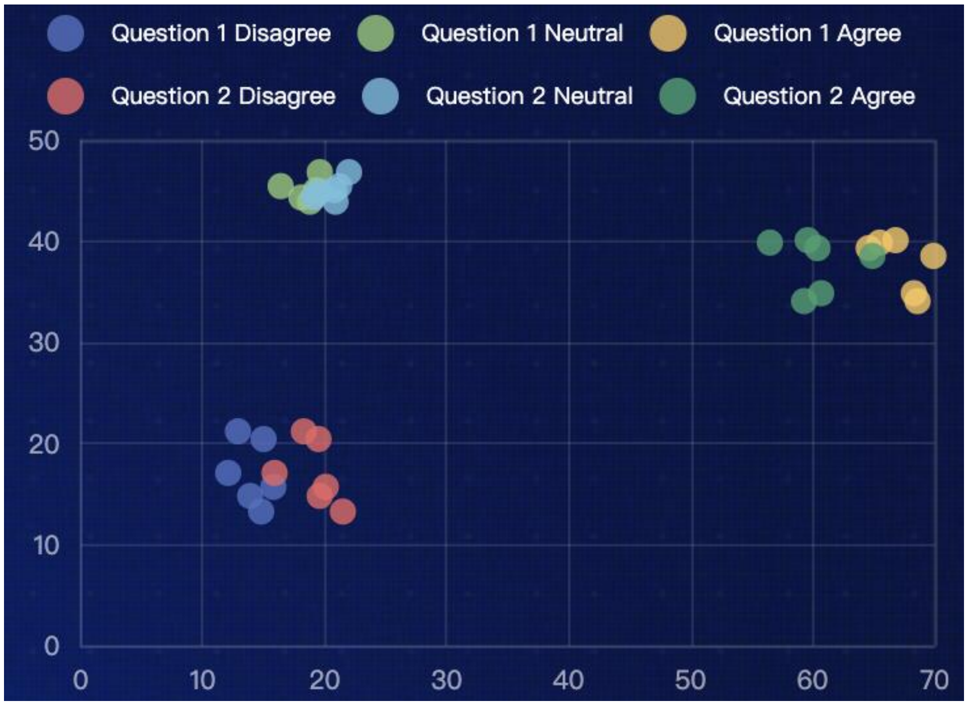


The above figures demonstrate the attitudes of tweets towards COVID-19 vaccine related topics in each city. From the graph, Newcastle is much more optimistic than all the other cities. Tweets from Canberra are also slightly happier than other cities followed by Perth and Golden Coast.

However, the average sentiment score for tweets from Melbourne and Sydney is relatively low. A guess for the reason behind may be: people in Sydney and Melbourne have more exposure to the rest of the world, the concern of the effectiveness of the vaccine towards varied types of Coronavirus coming from various parts of the world may be stronger.

However, we could also see an interesting fact that all other cities except Sydney have an above normal optimism in topics relating to covid vaccine.

4.4.5.5 Comparison of the sentiment score to the Survey Response



The above result gives a scatter plot of the response to the government survey to the sentiment score we have.

The first question from the Household Impacts of COVID-19 Survey is “**When a COVID-19 vaccine becomes available and is recommended for me, I will get it”.**

The second question from the Household Impacts of COVID-19 Survey is “**I will try to get a COVID-19 vaccination as soon as it is available to me.”**

For each question, we got 3 possible responses, and Australian Bureau of Statistics records the proportion of the response according to state. Therefore, for the 8 largest cities, we have overall 6 different states (VIC,NSW,SA,WA,QLD,ACT).

Comparing these 2 different questions, the first question shows a better correlation to our sentiment score output.

Points on the left bottom and right top corner shows the proportion of the attitude in accordance with the response is similar to what we got from the survey. Overall, the proportion of negative sentiment tweets is similar to people responding disagree in both questions. Due to the high proportion of neutral tweets the proportion of people responding agree is much higher than it for people writing positive tweets. There is no significant difference in response between these 2 questions. The result we got from the tweet is more similar to question 2 as compared to question 1, since the points are closer to the diagonal.

In conclusion, based on the comparison of the response to surveys and tweets harvested, a significant number of people posting neutral tweets will take the vaccine, when COVID-19 vaccine becomes available and is recommended to them, OR will try to get a COVID-19 vaccination as soon as it is available to them.

This shows that the sentiment score of tweets may underestimate the optimism of people towards Covid_19.

4.4.6 Sentiment score in unemployment related tweets with comparison in AURIN

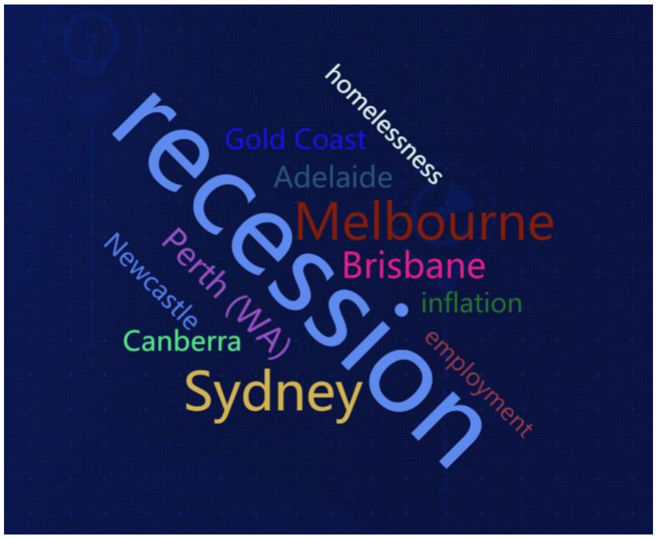
4.4.6.1 Topic focus

This topic mainly focuses on the relationship between the labour force participation and the polarity of related tweets. The scenario first reveals the popularity of unemployment and economic related topics. After that, the team wants to study if the unemployment rate affects the sentiment in tweets for different cities.

4.4.6.2 Reason to choose

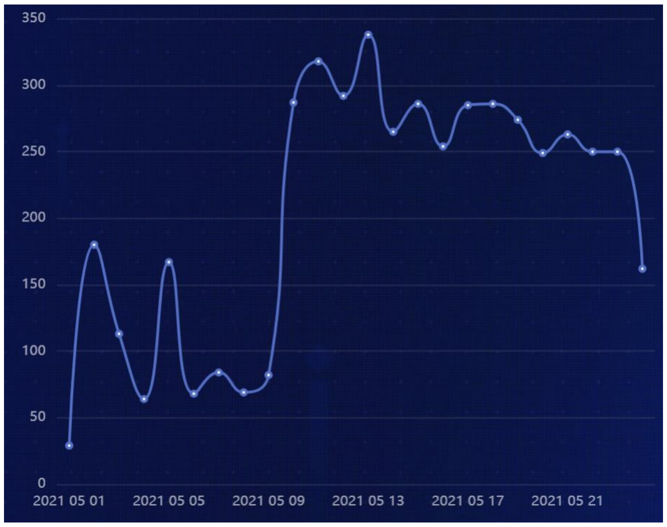
After the pandemic, Australia’s unemployment rate has increased significantly, and the government provides allowance such as Job Keeper and Job Seeker. The government also aims to reduce the unemployment rate back to a normal level in 2021-2022. Currently COVID-19 is found again across Australia, which may cause negative effects on the economy. The scenario wants to study the view of this potential issue in different cities. Results are carried out by using keyword matching in the text in tweets and compared to the unemployment rate measured by the Australia Bureau of Statistics.

4.4.6.3 Hot Topics And Area



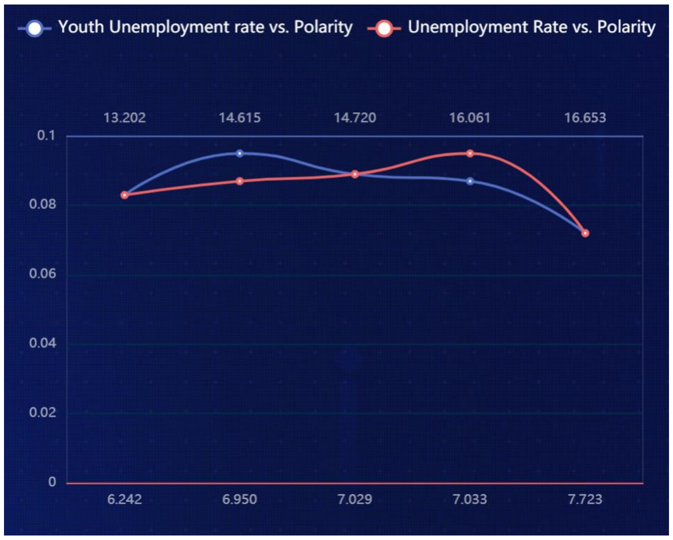
During the measurement, the related word in the dictionary that has been mentioned most of the time is recession in tweets, which is reasonable as a result. Words like inflation, employment and homelessness are frequently mentioned in the tweet as well. These words can reveal the most concerning worry in the society. The other thing shown on the plot is the cities with the largest number of related tweets. There is no surprise that Melbourne and Sydney got the most number of tweets due to the population base.

4.4.6.4 Mention related vs. Date



As shown above, many people start mentioning this topic in the middle of May (With a peak on May 13th that the system streams 338 related tweets). The result is reasonable since with the reappearance of COVID-19 cases across the country, people may worry that there will be another lockdown which may cause recession. However, there may be some bias during early May that the crawler is not working 24 hours during our update of the database and the last day(May 24th) where we stopped our streaming of tweets.

4.4.6.5 City Unemployment Rate vs. Average Polarity



This plot wants to study the relationship between the unemployment rate and the sentimental appearance in different cities. Due to the GCCSA spatial level is not fully suitable for our chosen cities, the data is only compared across Melbourne, Sydney, Adelaide, Brisbane and Perth. By the assumption that higher unemployment rate may affect the popularity of the unemployment topics discussed on Twitter, we test our hypothesis over the overall unemployment rate(All labour force) and the youth unemployment rate(18 - 24 year old labour unemployment rate). The result given above shows that the total unemployment rate in each city is positively correlated with the average polarities. Since most tweets are defined with polarity close to zero (natural tone), the difference of polarity is not significant for each city. We also observed that the youth unemployment rate is negatively correlated with the polarity. We may conclude that the people in the higher age group are more likely to be worried about the unemployment issue.

5 Issues and Challenges

5.1 error handling

5.1.1 remove duplicates

In this part of the report, the reason why repeated tweets are achieved and ways to remove duplicates are explained below.

Although our program makes corresponding judgments when fetching tweets, and checks whether the newly obtained tweet id exists in the database in order to get unique tweets, it is still possible that some repeated tweets will be fetched due to the reason that instances on the MRC fetch tweets simultaneously. Thus, some strategies are used in our program to completely remove the repeated tweets and store non-repeated tweets into CouchDB.

The first strategy is to separate the Australian territory into four parts and use different bounding boxes on the different instances to fetch tweets. This strategy is the main method to eliminate duplicate tweets. But it is also possible that two instances fetch tweets on the same bounding box due to misoperation. In this case, a new strategy is introduced, and it turns out that it can remove duplicates completely. This strategy is written in the removeduplicate.py and the logic behind it is to build a dynamic dictionary to store different tweets by setting the _rev as key and the content of that tweet as the value in that dictionary. Every time when this python file executes completely, it will record how many tweets have currently been processed in order to separate the processed tweets from the newly obtained tweets. In this way, we don't need to traverse the whole table where the raw tweets are stored whenever we want to remove the duplicate. We only need to process the newly obtained tweets and store the unique tweets in the non-repeated table.

It is of vital importance to remove duplicate tweets before analyzing them. Doing so can not only reduce the data storage in the database so as to occupy fewer resources but also reduce the amount of calculation caused by analyzing duplicate data.

5.1.2 Retweet

In the onstatus method, determine whether the tweet is retweeted or not, and get the data when it is not retweeted.

5.1.3 ProtocolError

When receiving too many tweets, the API can not handle that accumulative processing time, leading to the connection broken. This error happens when the program wants to read new tweets while the connection is broken. Thus, the stream is restarted whenever this error occurs.

5.1.4CouchDB Error Handling

The CouchDB database used in the project has been designed to be run on a cluster. A crash of nodes in the server will not affect the availability of retrieving data or views. Compared to single node services, the cluster database provides more reliability.

5.2 Melbourne Research Cloud

Through using the MRC, the biggest challenge is the unstable network connection. The anyconnect VPN from mainland China to the university network is quite unstable and connection constantly got lost, which made it hard to debug the deployment process.

5.3 Twitter

5.3.1 Twitter Mining

The team chose to use tweepy by the official twitter developer access token. As this method is limited by Twitter that we can only retrieve tweets within the past seven days, the team needs to stream tweets across the whole period of the project to get a valid time varite dataset. The query rate is also limited by Twitter and sometimes bugs appear while retrieval as well. Since the method “tweetpy.search” is not able to filter retweets, the team needs to use the “Stream” method for retrieval. Also, twitter has blocked all IP addresses from China even with VPN connections with the University of Melbourne, some of our group members have lost their twitter developer account or even failed to set up for one. In order to get a reasonable amount of tweets for further analysis, the team has decided to run the stream process 24/7 on the MRC server.

5.3.2 Tweet Size

The size of a fully retrieved tweet is found to be around 2.3 kb (tested with 150 tweets), most content such as user’s information and the entities are not useful for our project. In the design of our project, the team needed at least 500,000 tweets for analysis which may cause a huge waste in the volume. Also, as the properties of CouchDB(or other NoSQL document databases) for querying needs to read each document in the database, the time and memory cost of using the raw tweet data would be much higher than the preprocessing.

The solution to overcome this issue is to simplify the tweets between streaming and saving. In the current design, we only save the twitter generated id, created time, source(devices) of sending, text and the place name in cities.

5.3.3 Language Processing Limitation & User Groups

Tweet is a classical example of short text corpus which challenges the team for analysis in a limited amount of information retrieved. When constructing the analysis for different scenarios mentioned in section 4.4, the team has to ensure that the amount of information is representative to cover the 8 largest cities in Australia. Also, some of the topics may not be as popular in the society which needed us to scale up the amount of tweets.

Another limitation is caused by the user group in different social media. In the scenario that analyses the language spoken in different cities compared to the survey done by Australia Bureau of Statistics, due to Twitter being more likely to be used by English speakers, the result may not be as accurate as we expected.

5.3.4 Sentimental Analysis

Since Twitter is a multi-language based social media platform, each tweet may also contain topic or @ information which consider noise to our analysis, the sentimental score for each tweet is quite hard to compute. The team decided to use the TextBlob package with the NLTK(Natural Language Toolkit) library in Python for the tokenization and sentimental calculations. However, since the algorithm of the polarity and subjectivity is not found, we can only assume that the results are correct without proof. Based on the samples that we took out and checked manually, the methods work most of the time.

5.4 Containerization

The biggest obstacle for setting up the CouchDB cluster in a docker container is to get all the parameters right. Several ports should be open such as port 5984 for all HTTP API requests, 4369 for Erlang map since CouchDB uses Erlang-native clustering functionality to achieve a clustered installation. Some of the parameters are related to each other during usage, which causes a very tricky deployment problem. When we try to deploy the CouchDB cluster, a HTTP Error 500 constantly shows up. After a lot of searching and trying, we finally found that the problem is a *NODENAME* parameter setting as the IP of the corresponding instances is required. This is not mentioned in the official documentation and takes a lot of time to locate the issue, which happened to many teams since we got quite a few classmates asking for the solution.

5.5 CouchDB Database

5.5.1 Duplicated Tweets

As mentioned in section 2.5.3, CouchDB cannot perfectly filter the duplicate data. Also, due to the stream method having no control over duplicates, around 40% of the tweets are put into the dataset. The reason for this may be: 1. using multiple tokens to retrieve tweets at the same time or 2. The number of tweets in such areas has not increased significantly for the twitter API to give different tweets when we stream (May due to the time that no user is sending a new tweet at 2-3 a.m. except us).
The solution of this problem is fully explained in section 5.1.1.

5.5.2 The Limited Resource for User

CouchDB is found to be not well supported. When the team is trying to search for tutorials or documentations, often there is no solution available. Especially the Reduce function in the View of a database. There are bugs that some of the results may return Null when we group them by key. Self made code is not supported with error or syntax check, often after waiting for 5 minutes for generating the view, the results reveal that no document was found. Mango query is quite different from the SQL language and requires us to spend more time studying and using it. Some features are not available in the database anymore but the Python couchdb package has never been updated.(e.g. Temporary view is not supported anymore since CouchDB 2.0 which is not even available for downloading, but the couchdb.Database.Query methods can still be called) The limited amount of code explanation and code made us confused about whether the method is not working or we have syntax error in the query.

5.6 Aurin Data Collection

Given AURIN data came from different organizations across Australia, the data format of each dataset may be hugely different. Also, the data are from different spatial levels (e.g. statistical area, local government area etc.) may cause extra work in the searching process. Thus, during AURIN data collection, we have to first find the data in a correct spatial level as we want, and then process them into a form that we need.

5.6.1 Data Searching

As we mentioned in section 3.1, AURIN is a crucial infrastructure with a huge amount of data. Most of them are distributed in different ranges of areas with different spatial level settings. In the current design, we are only interested in the data in the level of 8 greater cities across Australia which is overlapping with the statistical area 2. However, the browser in the AURIN portal is not perfect for filtering the keyword of searching. The team has to spend lots of time looking at the description of the dataset and checking if some key attributes existed.

The other problem we found using AURIN is that most of the data are out of date. As twitter can retrieve information that's more recent, the AURIN dataset may not be as persuasive to the audience. The team also retrieved data directly from the Australia Bureau of Statistics to overcome this issue.

5.6.2 Data Preprocessing

Fields created in the dataset are more likely to be designed for database management instead of human understanding. When the team downloaded the JSON formatted data from AURIN, we often needed the attributes table for understanding. Also, some data come with a smaller spatial level, which requires us to merge them manually using Python programs.

6 Video Link

https://youtube.com/playlist?list=PLTaCIQqMUNXi4Ek6jbj0JMC2ZhHlzFexI

7 GitHub Link

https://github.com/TyrionGump/COMP90024_proj2

8 Team Member Role

Ran Liang	System deployment; related report
Yubo Sun	Front-end application; related report
Yulun Huang	Twitter harvester and data analysis; related report
Xindi Fang	Twitter harvester and data analysis; related report
Yanhao Wang	Twitter harvester and data analysis; related report

Bibliography

Research Data Australia. (2021). Retrieved 22 May 2021, from
https://researchdata.edu.au/contributors/australian-urban-research-infrastructure-network-aurin

Documentation, M. (2021). Melbourne Research Cloud Documentation. Retrieved 24 May 2021, from
https://docs.cloud.unimelb.edu.au/

Nick Evershed, Josh Nicholas, Andy Ball and Mostafa Rachwani, *Covid vaccine Australia data tracker: how the rollout is progressing, tracking daily new coronavirus cases, stats and live data (2021)* Retrieved 24 May 2021
https://www.theguardian.com/australia-news/datablog/ng-interactive/2021/may/24/covid-19-vaccine-rollout-coronavirus-updates-tracker-australia-daily-live-data-stats-update-total-numbers-distribution-progress-schedule-tracking-new-cases-today-statistics-latest-news

Federal court rejects bid to overturn India travel ban – as it happened Retrieved 1oth May
https://www.theguardian.com/australia-news/live/2021/may/10/australian-politics-live-coalition-talks-up-spending-on-infrastructure-and-womens-health-on-budget-eve

What is a Container? | Docker. (2021). Retrieved 24 May 2021, from
https://www.docker.com/resources/what-container

Teenager and Sleep. Better health Channel
https://www.betterhealth.vic.gov.au/health/HealthyLiving/teenagers-and-sleep

Mobile Vendor Market Share in Australia - April 2021
Retrieved from https://gs.statcounter.com/vendor-market-share/mobile/australia

Bartel, K, Richardson, C & Gradisar M 2018, Sleep and mental wellbeing: exploring the links, Victorian Health Promotion Foundation, Melbourne

Adams, R., Appleton, S., Taylor, A., McEvoy, D., & Antic, N. (2016). Report to the sleep health foundation 2016 sleep health survey of Australian adults. *The Adelaide Institute for Sleep Health*.