Twitter Data Harvesting and Analysis

TEAM 12

May 17, 2022

Name	Student ID	Email	City
Qi Li	1138875	lql4@student.unimelb.edu.au	Melbourne
Yuheng Guo	1113036	yuhengg1@student.unimelb.edu.au	DongYing
Zhaoyang Zhang	1240942	zhaoyangz1@student.unimelb.edu.au	Beijing
Zhangyu Wei	1258372	zhangyuw@student.unimelb.edu.au	Beijing
Xiaohan Ma	1145763	mxm3@student.unimelb.edu.au	RiZhao

Abstract

In our project, we used the Ansible to automatically configure the instances on MRC and collect data from multiple databases including the AURIN, Twitter API, City of Melbourne, Crime Statistics of Australia. The data was analysed inside the couchDB using Map-reduce and we incorporated the couchDB database into the back-end and presented the data visualisation on the front-end.

GitHub link: https://github.com/DoreenML/COMP90024-Assignment2.git

YouTube link:

Analysis and Front-end UI(deployed version):

`https://youtu.be/OdsyBCp4U_M`

Analysis and Front-end UI(Research Presentation):
`https://youtu.be/e1Pj3rdj_RY`
`https://youtu.be/7wF-oYmbCgY`

Cloud Architectures:
`https://youtu.be/5nE3SPhfMw0`
`https://youtu.be/mPg9JhnOxoM`
`https://youtu.be/Pm2XoCHdvBI`
`https://youtu.be/8UTsqlMSsBc`
`https://youtu.be/V6z1-JnERq0`
`https://youtu.be/0UyxxL10_Hc`
`https://youtu.be/nNglE2Xeuf0`
`https://youtu.be/8bJXW_fajg4`
Cloud Architectures overall:
`https://youtu.be/PJLowoLBK1g`

1 Introduction

Question: How many tweets mention Covid-19 or coronavirus and are these clustered in certain areas, e.g. rich vs poor suburbs or in areas where there are more/less hospitals/GP practices etc? (Health indicator)

The spread of viruses and policies such as embargoes implemented by governments to prevent the spread of viruses have a serious impact on our lives. The pandemic has caused the largest unemployment rate in over a decade, as well as a climate of dread, anxiety, and tension in both developed and developing countries.(Higgs et al., 2019) These effects, particularly in the domain of mental health, have not been thoroughly researched and understood. In this situation, understanding and characterizing the incidence of mental illness in the general population during a pandemic demands special attention. Therefore, the specific scenario we have selected for this paper is to study what insight tweets give us regarding the evolution of Australian society's mental health, since the beginning of the pandemic. According to sentiment scores for four regions as presented in tweets, the number of psychological emergency patients obtained from *AURIN*, and thematic modeling to capture the top trends on Twitter to have a better understanding of the aspects of the essence of liveability in Melbourne.(*Aurin Home*, 2022)

Governments at all levels around the world are increasingly aspiring to create "liveable cities" (Giles-Corti et al., 2016). We developed a comprehensive definition of urban livability based on a literature review, which includes "safe, attractive, socially cohesive and inclusive, and environmentally sustainable communities; affordable and diverse housing, education, public open space, local shops, health and community services, leisure and cultural opportunities." There are well-established traditional techniques for studying the liveability of cities currently available. These methods rely significantly on empirical site analysis, which can be time and cost-intensive yet still only gives a partial picture. However, not only in Australia but around the world, people's lives have changed dramatically as a result of the global pandemic and consequent shifts in human behavior. Therefore, as the global epidemic has demonstrated, things can change quickly, making it more vital than ever to develop new methods for correctly and efficiently determining well-being data and reference figures.

Because users may make quick status updates, Twitter has become a powerful tool for measuring and evaluating demographic changes as one of the most frequently used social media platforms.Furthermore, social media platforms such as Twitter have been successfully used in academic studies on urban planning concerns. Due to the massive volume of tweets published by Twitter users and the required tweets to efficiently gather and analyze these tweets, scalable computer processing capacity in the form of cloud computing is frequently required. Our objectives for this project were as follows:

1. cloud-based and highly scalable solutions on *MRC*.
2. Harvest tweets data using Twitter API and save these data to *CouchDB*.
3. Obtain supplementary data from *AURIN* (*SAHEALTH - hospital locations (point) - aurin data*, n.d.), process data based on scenarios, and do data analysis.
4. Design user interface,users can interactively query and extract information about every scenario using a web-based tool.

Section2 provides a detailed introduction to the components of system architecture and an overview of the design process and system deployment. Section3 provides simple user guidelines. Section4 introduces the technologies used in cloud deployment, including *Ansible*, docker, and how these technologies are applied to deployment on *MRC*. Section5 describes the architecture of harvesting Twitter data and the detailed information of data collected from *AURIN*. Section7 introduces the process of data processing and data analysis. This includes using *AURIN* data to support our arguments. In terms of Twitter data, sentiment analysis, hashtag analysis, topic modelling, and word cloud output are included. The results of the data analysis are also explained in detail in this section. Section8 shows the technical details of the front-end visualization of data analysis results. The conclusion of this report is presented in section 9.

2 System architecture and design

Our system architecture is depicted in the diagram below, which clearly depicts the technology stack and the link between software, hardware, and the network. Our main goal is to create a data pipeline that can crawl tweets from Twitter. We manually store the data found from the *AURIN* in a database as well. After that, the tweets are saved to a database. The collected tweets are then analyzed according to our scenario. Then the results are saved to a new database. Then the Twitter data is compared to *AURIN* data. Finally, the user can use the Front-end component to browse the Twitter data analysis results and visualizations.

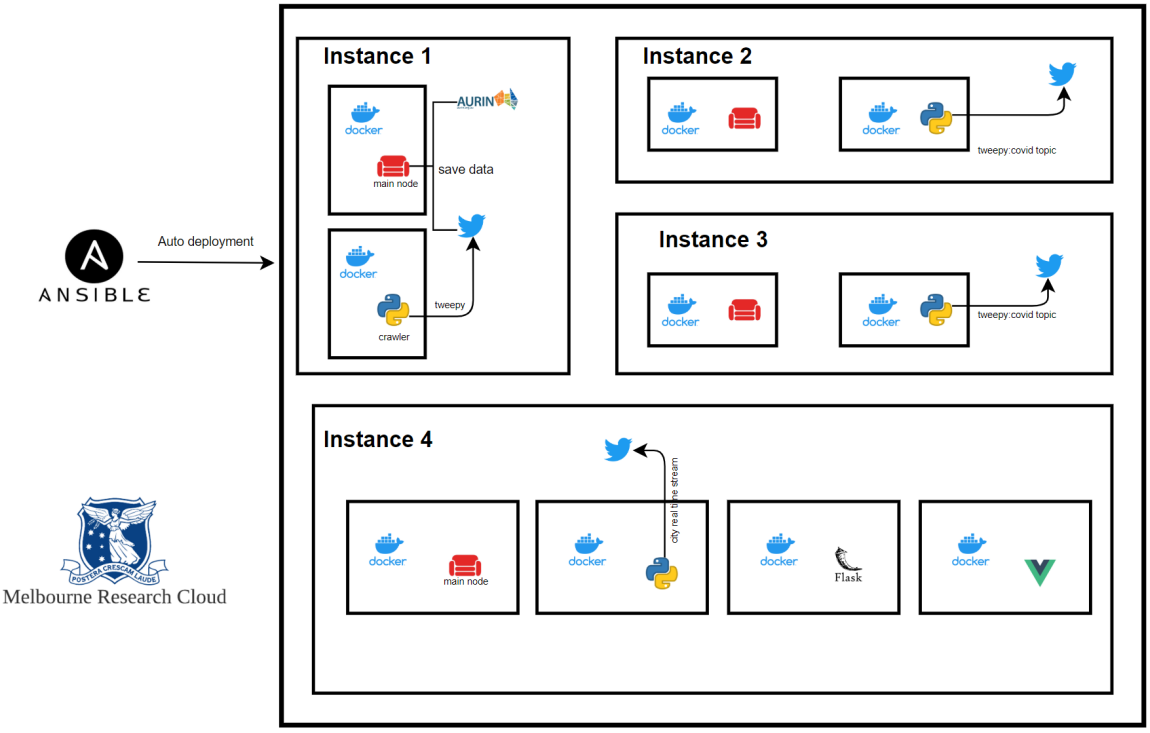

Figure 1: System Architecture

2.1 Functionality

2.1.1 Cloud

The system architecture of this report is based on cloud infrastructure which offers on-demand computing services to researchers. In this project, we use the University of Melbourne Research Cloud (*MRC*) to help to collect, store, analyze data and host the components for the web service. Every group is allocated the following hardware configurations: 4 instances, 8 virtual CPUs, 36GB RAM, and 500GB Volume Storage.

We deploy 4 instances on *MRC*. We deploy the tweet harvester in instance 1 to instance 4 because we have very heavy data collection tasks. Therefore, instance 1 to instance 4 are conducting Twitter Harvester and is part of the *couchDB*. Instance 4 conducts the role of Web Server. The detail of the configuration is shown in figure 2.

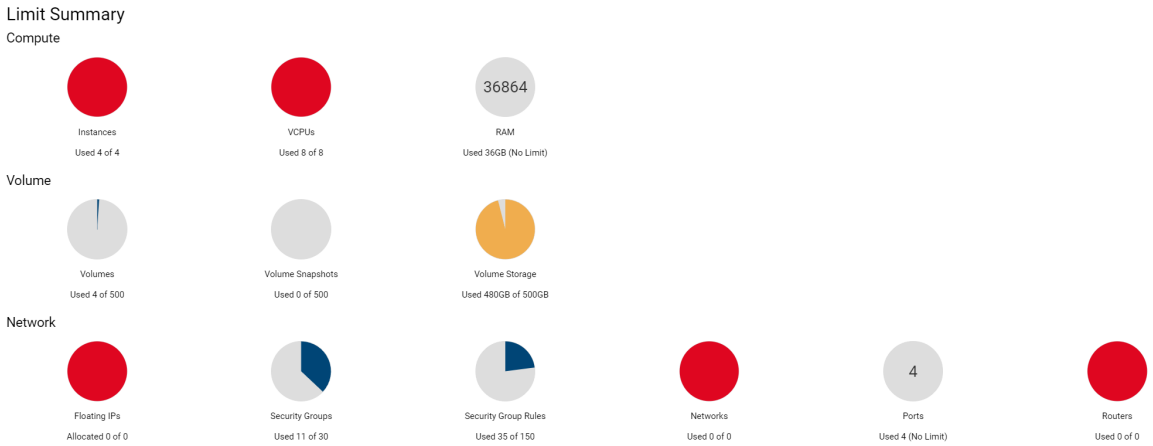

Figure 2: Melbourne Research Cloud resource usage

2.1.2 Twitter harvester

We need to collect a vast volume of Twitter data to get compelling analytical results. We need to use the Twitter harvesting function to get such a vast amount of data from a Twitter application with so many users and real-time updates. Pre-processing the data is required for harvesting: selecting the tweets about a certain topic, deleting duplicate tweets, and so on.(Banda et al., 2021). Different researchers have conducted various experiments on the analysis of Twitter. We could use Twitter to analyze random pulled Twitter information as well.

2.1.3 Database

We need to store the obtained data in a database because the tweet data is collected continuously throughout time. In this project, we use Apache *CouchDB* to store data in JSON-based document formats. Apache *CouchDB* is a kind of *NoSQL* database, which is used to achieve eventual consistency, high availability, and partition tolerance. We chose clustered mode for *CouchDB* because it enhances database dependability via internal balancing, sharing, and replicating the database using internal HTTP calls. A cluster is built to manage distributed data when data is harvested from different nodes. Furthermore, the database should react to requests for data retrieval from the web server at any moment.

2.1.4 Web Server

The web server should simultaneously receive requests from multiple clients, get requested data from the database, and visualize the data.In this project, we use *Vue.js*,*Node.js*,*High-chart*,*E-chart* etc dependencies to achieve an excellent visual front-end framework.We use flask to develop the back-end server.

2.2 Scalability

Scalability is an advantage of *MRC* when the load of our application is too high. We may construct and administer our own dynamic and scalable environment in the assigned cloud environment. We can use server images to build server instance images and replicate instances, but we'll need to create as many servers as we require. We may also develop a Docker container image and re-deploy the containers to different ports. We can execute many applications in various Docker containers on the same server in our project. Moreover, we can use an *Ansible* script to deploy the environment of the

instances and arrange resource allocation to satisfy the cloud's requirements.

2.3 Securiry

MRC is an Open Stack-based private cloud. The University of Melbourne makes each request through firewall and proxy servers, which improve the level of security. Our instances are only accessible from within a private network when they are launched. We needed to log into the University of Melbourne Virtual Private Network to access the servers. Furthermore, all server connections are made via the secure shell protocol (SSH). The private key is given to us when the server is first launched and is used every time we need to access it. As a result, only people with a private key can access the server. This ensures a certain level of protection. We can also create our security group which improves the security of each instance.

2.4 Fault Tolerance

We must ensure that our system is fault-tolerant, meaning that our system must continue to function properly in the event of a failure of one or more components. Fault tolerance of the system can be achieved by having the system restart from the point of failure. When an instance loss occurs, we choose to restart the instance. However, in the process, the Twitter harvest will also be restarted. In this case, there would be duplicate tweet harvesting. To handle this error, we will use an integer representation of the unique identifier provided by the Twitter API. We store the id of the last tweet fetched for each account in *couchDB*. Therefore, each time the harvest is restarted, it will start harvesting tweets from the last saved tweet id of that particular Twitter account. When data loss occurs, we need to spend a lot of resources and time on rebuilding the crashed database. To fix the problem, attach the same volume to the restarted instance and restore the backup data. Our data will be moved to another *CouchDB* database if the volume crashes as well. As a result, the data can be reconstructed.

3 User Guide

The development of the system can be divided into five parts:

3.1 Develop the Instance

Firstly, we created the four instances on the UniMelb Research Cloud. We need a security rule for these instances that can manage instances, for example, define a port for ssh to access. In addition, a volume also needs to be defined on the Research Cloud as storage. Then, creating a common role as install-based packages. We have created the four roles in the file named create_instance.yaml.
For the four instances, Instance 1 is used to develop CouchDB data. Instance 2 and instance 3 are used to harvest Covid-19 tweets data. Instance 4 is used to develop the back-end and front-end. Docker is a virtual environment. We open the docker first and run instructions on it.
How to Run
1. Run the *run_create_instance.sh*
2. Run the *run_install_instance.sh*
3. Run the *run_develop_instance.sh*
These shell scripts will run the three .yaml file ansible-playbook to manage the cloud infrastructure.

3.2 Configure the Instance

Secondly, configure the instances that have been created. We can add the *"config/mrcSSH.pem"* in to the configure file to access the Melbourne Research Cloud. They are adding proxies for four instances to ensure they can access the Internet and they can also be accessed by the outside environment. In this step, we also set up and clone the repository /home/ubuntu/COMP90024/COMP90024-Assignment2 for each instance. Then mount the volume that was created in the previous part.

We created the four roles in the file run_install_dependencies.yaml.
How to Run
./run_install_dependencies.sh
The shell script will run the "run_install_dependencies.yaml" ansible playbook to manage the cloud infrastructure.

3.3 Develop CouchDB Harvest Data

Install CouchDB first, then allocate four instances. Instance 1 is used to build a CouchDB cluster as a DB Master and build two images for the harvester; the two images are streaming and search. Then construct DB Slave as two nodes in instance 2 and instance 3. When running run_develop_instance.yaml in the first step, the CouchDB has been developed because run the constant.yaml in the host_vars folder. After developing these, we can achieve harvest tweets data and transmit data.
How to Run
1. *cd CouchDB*
2. *python dataUpload.py*
3. *cd Harvester*
4. Open every search file in the Harvest and run the python file to crawl data.

3.4 Develop Back-End Server

In this step, we use Flask as the back-end server and import two " jonsonfy " libraries and the Flask Cross-Origin Resource Sharing (CORS) libraries for connection. We discussed this part detailed in section 8.1.2
How to Run
1. Open the BackEnd folder – *cd BackEnd*
2. *python BackEnd.py*

3.5 Develop Front-End Server

How to Run
Open the front-end directory firstly.
1. Project Set up
yarn install
2. Compiles and minifies for production
yarn build
3. using the command
npm run serve to run.
We discussed this part detailed in the section 8.1.2

4 Cloud Infrastructure

Clouds are hosting the full computational workload in this project. Cloud infrastructure can help with data collection, storage, large-scale data analytics, web-service implementation, and hosting web-services. We use the University of Melbourne Research Cloud infrastructure, *Ansible*, and Docker to implement the architecture and deployment of this project.

4.1 Melbourne University Research Cloud(*MRC*)

4.1.1 Functionality

Students and researchers at the University of Melbourne can use *MRC* which provides on-demand and Infrastructure-as-a-Service (*IaaS*) cloud services. As their workload expands, the researchers will have

easy access to scalable computing capacity. The *MRC* provides about 20000 virtual cores in a variety of *VM* sizes. The *MRC* is built on the Open-Stack open-source implementation platform. Because our project is based on *MRC*. We will discuss the advantages and disadvantages of *MRC*.

4.1.2 Advantages

1. **Cost saving-** We no longer need to spend money on hardware installation, and we no longer need to worry about hardware setup, software patches, and other time-consuming IT management tasks.

2. **Faster-** We can provide massive amounts of computing resources in minutes and eliminate the stress of capacity planning. For example, we can create several instances in just a few minutes to help us access, store and analyze data.

3. **More Security-** Provides a wide range of policies, technologies, and controls to improve overall security that helps protect data, applications, and infrastructure from potential threats.

4. **Scalability-** Resource Provisioning You can pre-provision the number of resources based on actual needs. You can immediately scale up or down these resources to expand or reduce capacity as business needs change.

5. **Flexibility-** The hardware and other resources can be customized easily by the development team with the *MRC*. Each instance can be configured as required.

4.1.3 Shortcomings

1. **Limitations in available resources-** The project's resource allocation does not allow for the provision of computing instances with public IP addresses. The deployed infrastructure can only be used within the university network, thus limiting its potential user base. And if users are located overseas, this distance can cause communication delays.

2. **Security-** Each instance on the *MRC* is secured by a key-value pair. If you leak your key, then others can access your cloud and delete your instances on the cloud. In this case, you may lead to data loss and cause security issues.

4.2 *Ansible*

4.2.1 Functionality

Ansible is an open source configuration management and application deployment tool based on python that combines many operations and maintenance tools. *Ansible* automates multi-layer deployments of complex systems with machine-readable configuration files, allowing system integrity to be maintained over time.*Ansible*'s modules contain configuration information for remote emulation systems. It uses a manifest file to store all receivers that need to be configured or initialized. *Ansible* does not require a client to be installed on any remote machine; it is executed securely via SSH by default. The detail of the ansible flow chart is shown in figure 3 and figure 4.

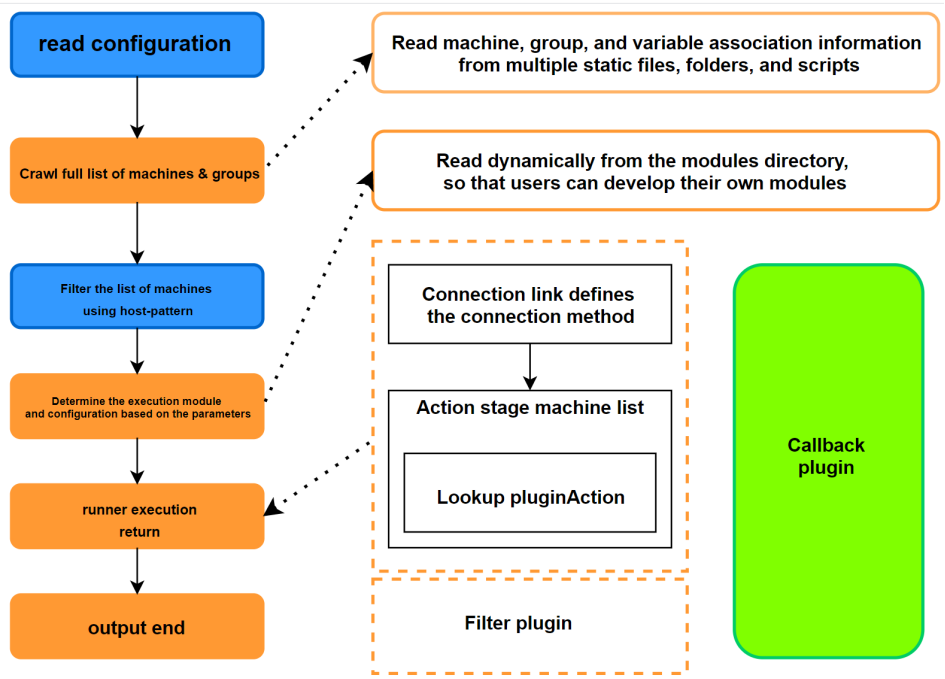

Figure 3: *Ansible* Flow Chart

In our project, we use *Ansible* to automatically configure and deploy multiple instances. It builds a blueprint of automation tasks for instance deployment on *MRC* server, which assigns the following config. It uses SSH to connect to the server and run the configured tasks, and uses the following file to control the process:

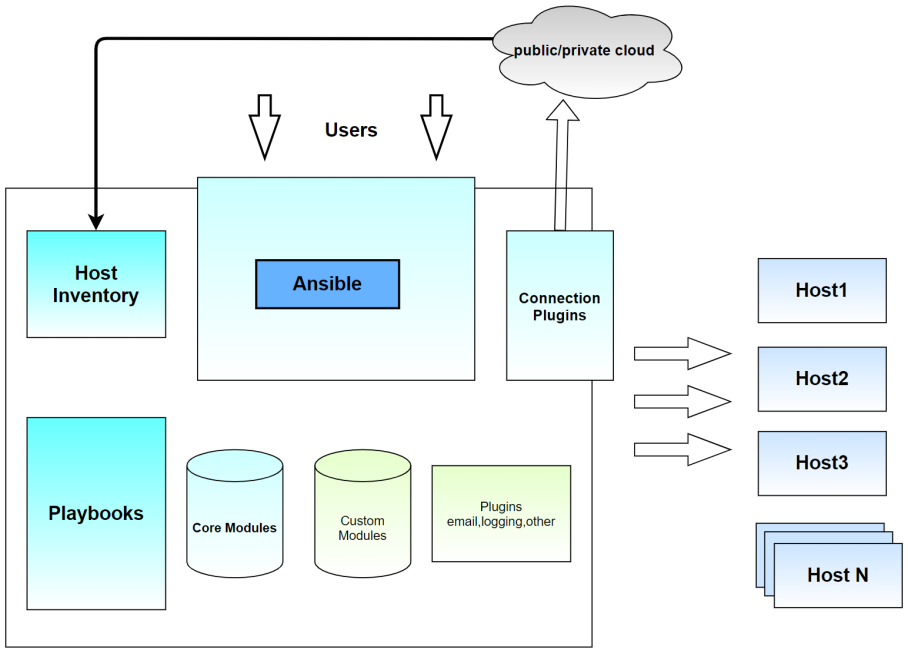

Figure 4: Ansible Flow Chart

Inventory files: Used to define the servers that will be managed. This file is named hosts.in the

host's file, we can define some of the servers to be managed.
Playbooks: The Playbook can run multiple tasks and offers some more advanced features. Let's move the above tasks into a playbook. In ansible, both plays (playbooks) and roles (roles) are defined using Yaml files.
Handlers: The handler is the same as a task (it can do anything a task can do), but only runs when another task calls it.
Role: Save the role to Ansible's public registry, and create the base directory structure for the role locally.

The most significant file is the Playbook, which orchestrates how the other files listed above are used. The playbook organizes the flow of all the other sections, preserving the order of calling tasks. The host variables are also linked to each of the files in this section. We can't construct instances without first setting up the commons and volumes. The structure of the ansible playbook is shown in figure 5.

```
- hosts: localhost
  vars:
    ansible_python_interpreter: /usr/bin/python3
  vars_files:
    - host_vars/deploy.yaml
  gather_facts: true

  roles:
    - role: openstack-common
    - role: openstack-images
    - role: openstack-volume
    - role: openstack-security-group
    - role: openstack-instance
```

Figure 5: ansible playbook

openstack-common: This role sets the required software dependencies for the local host.

openstack-volume: To minimize data loss on our system, "cloud-volume" mounts two volumes, one for the general docker system and the other for Couch. The general docker system and the other is the Couch-DB data folder. The purpose of this is to have persistent data storage in case the Docker container fails.

openstack-security-group: The purpose of this role is to dynamically create security groups for SSH access to the instance and to open the required ports for the Couch-DB cluster.

openstack-instance: This role creates Ubuntu virtual machines and connects them to matching volume groups.

4.2.2 Advantages

Ansible is lightweight. There is no need to install the agent on the client. Only one update on the operating mechanism can be done when updating. Additionally, batch task execution can be written as a script, and it can be executed without distributing to a remote location. Ansible is written in python, and simpler to maintain. We can deploy the entire set of instances for our project using only Ansible playbooks and the OpenStack API. In addition, Ansible follows a known file structure that is

easy to understand. Ansible can support the following scenarios:

1. Application code automation deployment.
2. System management configuration automation
3. Support for continuous delivery automation.

4.3 Containerisation

4.3.1 Functionality

Containerization technology is sharing hardware resources and operating systems with our host, allowing dynamic allocation of resources. Containerization includes the application and all its dependent packages but shares the kernel with other containers. Containers run as separate processes in the user-space within the host operating system. Containerization technology is a way to virtualize your operating system, allowing you to run applications and their dependencies in processes where resources are isolated. By using containerization technology, we can easily package an application's code, configuration, and dependencies into easy-to-use building blocks that achieve many goals, such as environmental consistency, operational efficiency, developer productivity, and version control. Containerization technology can help ensure that applications are deployed quickly, reliably, and consistently, independent of the deployment environment in between. Containerization technology also gives us more granular control over resources, making our infrastructure more efficient.

Docker is a wrapper for Linux containers that provides a simple interface for working with them. It is the most widely used Linux Containerization solution at the moment. Another virtualization technology developed by Linux is the Linux container. In simple terms, a Linux container isolates a process rather than simulating an entire operating system. It's like putting a protective cover on top of a normal process. All resources touched by the process inside the container are virtual, thereby isolating them from the underlying system. Docker bundles an application's dependencies and the application itself into a single file. By running this file, a virtual container is created. Programs run in this virtual container as if they were running on a real physical machine. With Docker, you don't have to worry about the environment. Overall, Docker's interface is quite simple, and users can easily create and use containers and put their applications into them. Containers can also be versioned, copied, shared, and modified, just like managing regular code.

4.3.2 Advantages

Docker interacts with the kernel with little or no performance loss, so it requires very few resources. Docker's architecture can share a common kernel with a shared application library, which takes up very little memory, making docker very lightweight. Docker achieves high availability through rapid redeployment. And, docker has sophisticated assurance mechanisms to ensure business continuity and recoverability. Docker documents the container build process in a Docker file for fast distribution and rapid deployment in a cluster.

4.3.3 Shortcomings

One of the shortcomings of docker is security problems. Docker's tenant root is the same as the host root. Once the user inside the container is elevated from normal user privileges to root privileges, it directly has the root privileges of the host and can then perform unlimited operations. Therefore, the security of docker is weak.

The other shortcoming of docker is manageability. Docker's centralized management tools are not yet mature.

5 Data Collection

5.1 Tweet Harvesting

Twitter is a popular social network in which users share messages called tweets. Twitter allows us to mine any user's data using the Twitter API or Tweepy.(*Twitter API documentation — docs — twitter developer platform*, n.d.) The structure of the tweet harvester is shown in the figure 6. The data will be extracted from the user's tweets. The first thing to do is to get the Twitter developer's user key, user key, access key, and access key from each user easily. These keys will help the API to authenticate. The API class provides access to the entire Twitter RESTapi method, each of which can accept various parameters and return a response. The API class provides access to the entire Twitter RESTapi method, each of which can accept various parameters and return a response. Tweepy also supports long links to live messages. The data we used for the number of increases or decreases is from the official website of the Victoria COVID-19 website. This website is a dynamic website with changing COVID-19 cases. In our project, the implementation of the back-end enabled this feature and our front-end website always reports the newest information about the COVID-19 cases on the interactive map. (*Victorian covid-19 data*, n.d.)

5.2 Other Data

The resource of our data mainly come from three different databases. The first database we used in our project is the AURIN dataset, in which we collect the polygon information of Victoria city. (*Aurin Home*, 2022) The second database we use extensively is the "City of Melbourne" database, in which we collected the house and camera data. (*City of melbourne - open data portal: Tyler Data Insights*, n.d.) Another individual dataset we used is the crime dataset which is come from the crime statistics Agency of Victoria. (Crime Statistics Agency Victoria, 2022). This report implements Tweepy API to harvest tweets from given big cities in Australia and focus on each region's tweets in Melbourne. The script we used is deployed in MRC, and the architecture shown in figure 1. The main reason we used this architecture is that the API limited the number of access times, but not limited the number of tweets we can crawl from a specific Twitter user. We adjust our plan to focus on those cities' users who once cared about covid and mental relevant tweet content.

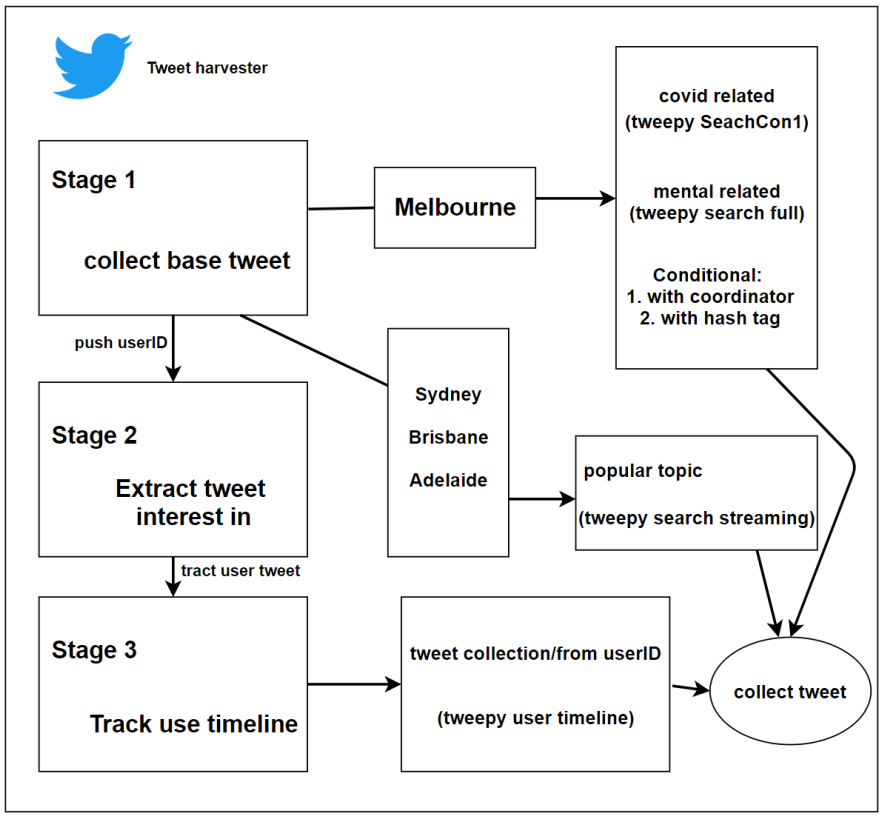

Figure 6: The stage we adpated to harvest millions of tweets

The harvester plan we adopted consisted of three stages. First, deploying a tweeter crawler on MRC to harvest tweets content we are interested in as more as possible and store it on the CouchDB server, including Covid-19, sentiment related, and that tweet has coordinates. The harvester also crawled the real-time tweets from other big cities. The total number of tweets we crawled is over 100,000. Secondly, we filter and extract the users we are interested in, and use a timeline API to backtrack those users' Twitter history. Finally, we track the user's tweets and harvest more than 10 million tweets. We take advantage of other data sources, including reference covid labeled tweets. The detail of the code can be shown in figure 7.

```
if countRequestNum % 25 == 0:
    print("rank:", rank, ", i", i)
    limits = api.rate._limit_status()
    timesRemain = limits['resource']['statuses']['/statuses/lookup']['remaining']
    if timesRemian <= 50:
        userTimeLine = limits['resource']['statuses']['/statuses/lookup']['remaining']
        userTime = time,localtime()
        currentTime = time.localtime()
        sleepTime = time.mktime(userTime) - time.mktime(currentTime)
        print("rank sleep for: ", sleepTime, "Seconds")
        time.sleep(sleepTime)
```

```
from mpi4py import MPI
def crawlTSVProcess(given_tsv, api, rank, aipLength):
    retrieveList = []
    countRequestNum = 0
    for i, row in enumerate(given_tsv):
        if i% apiLength != rank or i ==0:
```

```
            continue
```

Figure 7: Crawl Tweets Method

5.3 Issues & Challenges

1. The main challenge we meet in this part is that the Tweepy API strictly limits the number of tweets we can crawl in 15 minutes, the solution we provide is to run crawler in multi-thread where each thread uses an individual access token. We also use rate_limit_status method to check the remaining time to early stop. The second challenge is that there is a large amount of Twitter that we lacks coordinated information. To handle this problem, we apply an academic twitter development account.

2. In addition, to obtain a larger number of Twitter data, we found a new dataset at Banda et al. (2021). In this paper, the authors provided Twitter data from 1 January 2020 to 27 June 2021 for researching about Covid-19, then we can use ID as a key field to crawl related data about emotional and mental information.

6 CouchDB

6.0.1 Functionality

CouchDB is a NoSQL development tool for constructing a database with JSON document formats, and it can transmit data for applications by JavaScript and Mongo from the backend. In addition, CouchDB provides some functions, for example, document designs and cloud multi-database. There are three characteristics of CouchDB:

1. CouchDB is a distributed database. The storage system can distribute the N physical nodes, then it synchronizes the data read/write consistency between their nodes.

2. CouchDB is a document-oriented database and a semi-structured database for storage. It is more convenient and has better performance than relational databases.

3. CouchDB supports RestFul API. Users can operate the database by JavaScript, and they can write JS code to query.

6.1 Advantages

According to CouchDB characteristics, CouchDB is suitable for these aspects of our project:

1. **High Store Performance-** Compared with a traditional relational database, CouchDB stores data with N nodes. In our project, extensive Twitter data need to be stored, and the distributed database can write and read consistently with high performance.

2. **Wide Storage Field-** CouchDB stores data in JSON format; therefore, harvesting data from Twitter API can contain wide keys and fields. This method is convenient to store in the database.

3. **Simple to Use-** CouchDB provides a concise server "$http://localhost:5984/$" , then some functions can help design views and options in the visual interface directly. We can also write a view to show results in this project before analyzing the data.

4. **High Accessibility-** CouchDB can be accessed by RESTFul API and can achieve "create", "Update", "Delete" and "Read". it can run and be installed on the various operating systems, which will support development teams when teammates use different operating systems.

6.2 Database Structure

In this project, we created multi-database for different data clarification, include twitter data, house data and crime data (Crime Statistics Agency Victoria, 2022) etc. The detail of the couchDB structure can be shown in figure 8.

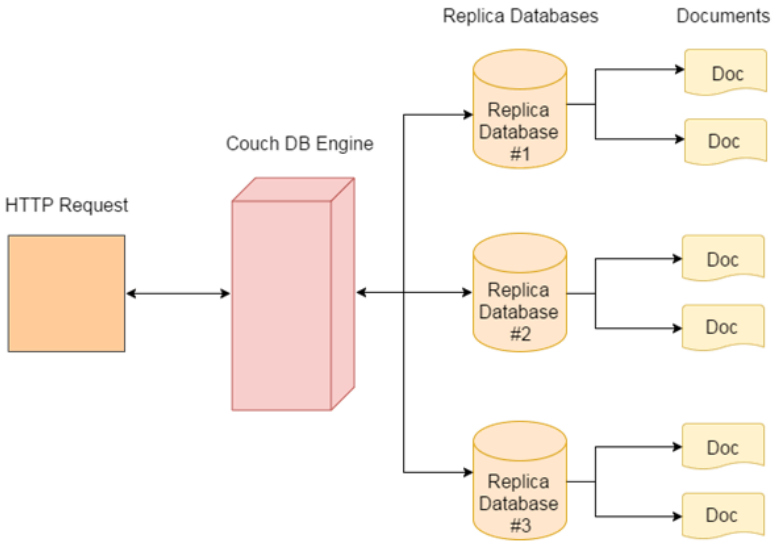

Figure 8: CouchDB Structure

The architecture of CouchDB is described below:

1. **CouchDB Engine-** The CouchDB Engine is based on two B-tree, they are by_id_index (The key is documents' id) and by_seqnum_index (The key is sequence number). The engine is reliable for Cloud and Multi-database.
2. **HTTP Request-** Through Get, Post, PUT to operate documents in database. "HTTP Request" also allows users to access **View** and **MapReduce**.
3. **Replica Database-** It is used to replica data to other database(Local or Remote).
4. **Doc-** The document is written with JSON format, it is used to store data.

6.3 Implementation

6.3.1 Views

CouchDB can design *views* in the visual interface with JavaScript code. Then we can filter data in JSON documents according to the relationship between keys and values. This operation can build dynamic views and index effective information, which will not influence the basic structure and data in the database. At last, building views can help users operate in these designed documents in the management platform.

6.3.2 MapReduce

MapReduce is a significant characteristic in database design. In CouchDB, it provides an Apache MapReduce, which will perform "sum", count" and "sort". Then, the function allows us to count values, sum tweets and sort dates. In the Fauxton, the website also gives us visual options can extract values. After creating MapReduce, users can connect "database" and "views" with python code to extract data, and the connection detail can be shown in figure 9.

```
# default couch
adminName = "admin"
adminPasswd = "Password"
```

```
    url = "172.26.130.135:5984/"
    # define couch
    couch_url = "http://" + adminName + ":" + adminPasswd + "@" + url
    couch = couchdb.Server(couch_url)
    datasetName = "dataset_Name"
    designName = "design_Name"
    viewName = "view_Name"
    db = couch[datasetName]
    view = db.view(designName + "/" + viewName, group=True, reduce=True)
```

Figure 9: Connect CouchDB and Views

For the **MapReduce** and**views** in this project, there is an example in figure 10:

```
# read sentiment relevant tweet data
Melbourne_save_list = read_json(JSON_FILE_NLP_BASE + "Melbourne_mental_suburb.json")
db = retrieve_couchdb(couch, "melbourne_mental_data")

# create view
mapFunction = '''function (doc) {
  if (doc.hashtag.length != 0){
    var volLabel = 0
    if (doc.rt*5 + doc.like >= 50){
      volLabel = 1
    }
    for (let i = 0; i < doc.hashtag.length; i++){
      emit([volLabel, doc.sentiment_polarity, doc.sentiment_subjectivity, doc.hashtag[i]], 1);
    }
  }
}'''

createView(db, "backend", "view_by_mental", mapFunction)
threadSaveMethod(Melbourne_save_list, db)
```

Figure 10: MapReduce Example

In this program, we need find the total **hashtags** of each places. However, in the JSON data, there are several tags in the list of the "hashtags" key. We must split the list and extract each tags from it. Then, we use a **for loop** to achieve this goal in the view. Then we use MapReduce function to count the number of these hashtags according to the same keys.

6.4 Issues & Challenges

There were two challenges about Couch-DB:

1. When transmitting data to Couch-DB, we cannot count value of duplicate key in database, then we found that using MapReduce to groupby. At last, we get results without duplicate data.
2. CouchDB faced crashes and slow operations when we loaded data to the database in this process, so we needed to try many times.

7 Data Analysis

7.1 Overview of scenario

When it comes to entertainment and happiness, Melbourne offers a plethora of amazing clubs and bars that are suitable for a social drink, social gatherings, and late-night clubbing sessions. Melbourne also

has the most live music venues in Australia, with local and international artists performing anything from hard rock and techno to jazz, and avant-garde shows, all of which, Melbourne appears to have everything, whether it's relaxing, dining, people watching, or shopping, in a colorful and cultural heritage atmosphere. Melbourne's entertainment scene is flourishing. People are asking why Melbourne doesn't just do these cultural shows a little more naturally as the yearly Melbourne Festival continues to impress.

Concerning the business objective, the team conducted several data analyses to provide evidence that Melbourne is one of the most livable countries with top-notch mental health in Australia and around the world.

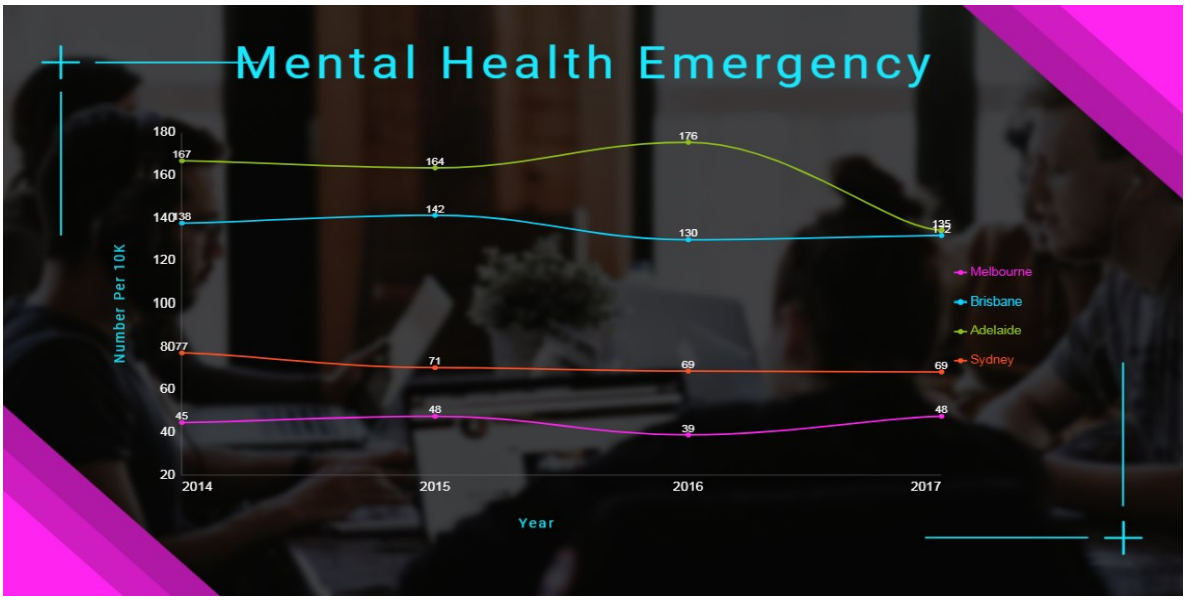

Figure 11: Mental Health Emergency

The above chart figure 11 shows the number of emergency department visits to the psychiatric department in four cities of Melbourne, Sydney, Brisbane, and Adelaide from 2014 to 2017. It can be clearly observed from the pictures that the number of psychological patients received in Melbourne is relatively small. This demonstrates the positive and optimistic attitude that Melbourne residents maintain in their daily lives. It can also show that residents do not have too much pressure on material conditions, thus this is one of evidence that proves the livability of Melbourne.

Even during the time of COVID, people in Melbourne seem to be relatively less affected by the pandemic. Livability of a city is very strongly correlated with the mentality of its citizens. One method to determine how healthy the mentality of the citizens is by examining the sentiments of messages people post on social media. Through sentiment analysis of Tweets and other Twitter postings, it has shown that residents of Melbourne recovered from the afflictions from pandemic rather quickly, have extremely strong mental health, positive mentality, and have easily transitioned back to their normal lives as time goes on.

7.2 Details of analysis

The presentation of the data analysis consisted of an interactive COVID-19 map data analysis inside

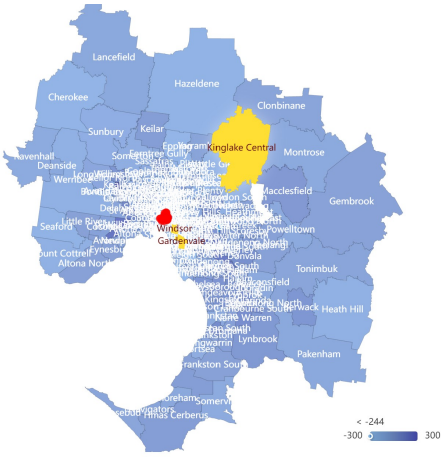

Figure 12: Suburbs COVID case decrease of 244

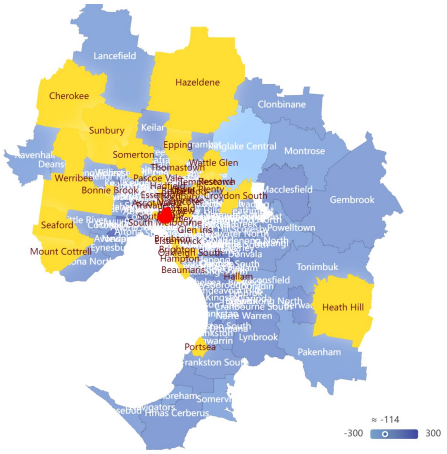

Figure 13: Suburbs COVID case decrease of 114

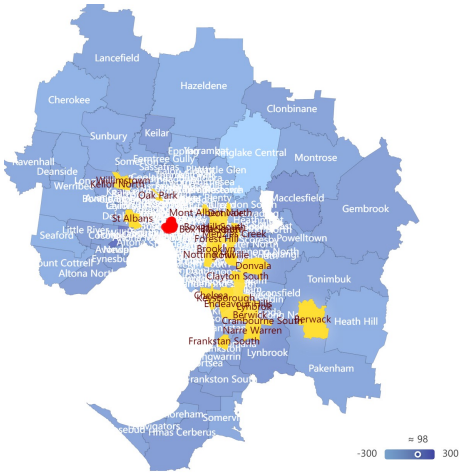

Figure 14: Suburbs COVID case increase of 98

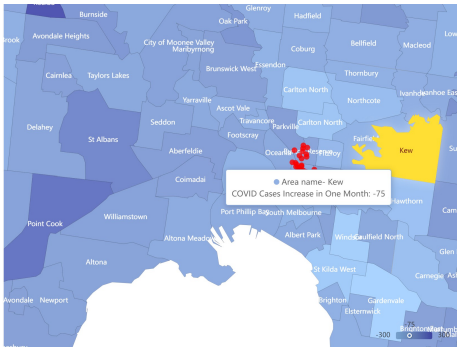

Figure 15: COVID cases notification

of Melbourne city. The interactive map includes 183 suburbs and we picked those many suburbs because these 183 places have important features and the data information is sufficient to do an in-depth analysis. Those suburbs are distributed all around the port and it's also the central business area of Melbourne. The polygons are colored in the sequential color scheme and those shallow blue and deep blue represent the COVID-19 cases updated over the past two months. The deep blue represents there are COVID-19 increases in this suburb and the shallow blue represents there are COVID-19 cases decrease in this area. This method is adopted from the cutting-edge health analysis paper in which the health condition of citizens in America was analyzed in the form of map polygon color distribution.("analyzing$_i$n$_f$ormation$_d$if$_f$usion$_i$n$_a$$_p$ublic$_h$ealth$_c$risis$_u$sing$_g$oogle", n.d.) As shown in Figure 12, the overall distribution of the COVID-19 cases is that the southeast part of the Melbourne has COVID-19 increases since it shows deeper blue and the northwest part of the Melbourne has COVID-19 decreases since the blue are shallower. This pattern also corresponds to the economic and population distribution of Melbourne since the east and south part of Melbourne is the population-intensive area and it's also where the Central Business District is located. The Figures 13, 14, 15 also show three different levels of COVID-19 cases. Figure 12 shows the area where the COVID has a decrease of 244, which is one of the largest decrease areas. This suburb locates in the north part of Melbourne. The figure 13 shows the COVID-19 decrease cases of around 100 and we could observe that those areas are all located in the north and west parts of Melbourne and far from the central business area. According to figure 14, we could observe that the most increased cases, an increase of around 100, happen around the central corner of the port of Melbourne. This observation is reasonable because this area is close to the C.D and the virus could transmit quickly within a month. Figure 15 shows the notification of the interactive map where it shows the number of increases or decreases for each suburb we would like to observe.

The first step in the sentiment to analyze mentality is to import all the necessary packages including Natural Language Toolkit and Gensim etc. Then the associated data are loaded, and the Tweet data are in JSON format. The raw data extracted and used are all the Tweets from Australia between the years 2017 and 2022. The data is divided into two main portions, namely before the pandemic and during the pandemic. Tweets from January 1, 2020, until the day of extraction, are considered during the pandemic, whereas January 1, 2017, up to December 31, 2019, are grouped as before the pandemic. The psychological health condition of the citizen's sentiment is analyzed from the Twitter information and this method is adopted from a similar analysis paper and the sentiment status of users is analyzed from the textual information from Twitter. (Culotta, 2014) In our project, we use the embedding library for the sentiment analysis of the textual information by using the *textblob* library. This library generates the numerical value of the sentiment and subjectivity for each text information.

Because the data is extremely large, it is not feasible to run the model using all the Tweets during the selected period. Instead, a sample of approximately 13 million treated Tweets is randomly selected for the analysis. These 13 million instances are original Tweets but from unique users, to eliminate the issue of duplicated Tweets from the same user. Next, the data are subjected to initial cleaning where missing values have been dropped. After taking out data with missing values, there are approximately 2.5 million Tweets from before the pandemic and 7.7 million Tweets from during the pandemic period.

Next, only the Tweets from selected cities are used for further analysis. The selected cities are Melbourne, Sydney, Brisbane, and Adelaide. Tweets from outside of these four cities are dropped out from the further analysis. From these four cities, there are approximately 1.7 million Tweets from before the pandemic and 4.6 million Tweets during the pandemic.

For the sentiment analysis, it is important to remove emojis and punctuations. A list of punctuations and emojis are first defined, then various links and time information are all removed. The Tweets are tokenized and free of punctuations and unnecessary links.

The following step of data cleaning is to determine whether the Tweets are associated with any mental health or mentality-related connotations. A Mental Health Keyword dictionary is implemented to aid in determining whether a Tweet is related to mental health. If the Tweet contains one or more words

that are found in the Mental Health Keyword dictionary, then that Tweet is useful for further analysis. Those Tweets that do not contain any words found in the Mental Health Keyword dictionary are dropped. It is found that there are about 120 thousand mental health-related Tweets from before the pandemic and approximately 330 thousand mental health-related Tweets from during the pandemic.

Following data cleaning, the next major step is sentiment analysis. In the sentiment analysis, the polarity of the mental health-related Tweets is evaluated. The polarity of a Tweet is a numerical value assigned and determined by a package called TextBlob. TextBlob generates sentiment polarity and subjectivity score for each tweet, every word is assigned to a value, then takes the average. If the polarity of a Tweet is on a scale from above 0 to 1, then that Tweet is associated with a positive tone. If the polarity of a Tweet is below 0 to -1, the tone of that tweet is negative. If a Tweet has a polarity of zero, then it is a neutral Tweet and is not associated with a positive or a negative tone. It is shown that most of the Tweets are neutral-toned Tweets, and they are not used in further analysis. After removing the neutral Tweets, it is shown that there are approximately 205 thousand positive and negative tweets from the four selected cities.

Exploratory data analysis for the sentiment polarity is conducted; Figure 16 and Figure 17 below show the sentiment polarity distributions of Melbourne and Sydney during and before the pandemic. Before the pandemic hits, the polarity distribution is not normally distributed. There is a skewness towards the left and there are more positive sentiments than negative sentiments. During the pandemic, as expected, there has been a small rise in the negative sentiments, but the overall polarity shape is still very similar to before COVID times, with most of the sentiments having a positive polarity.

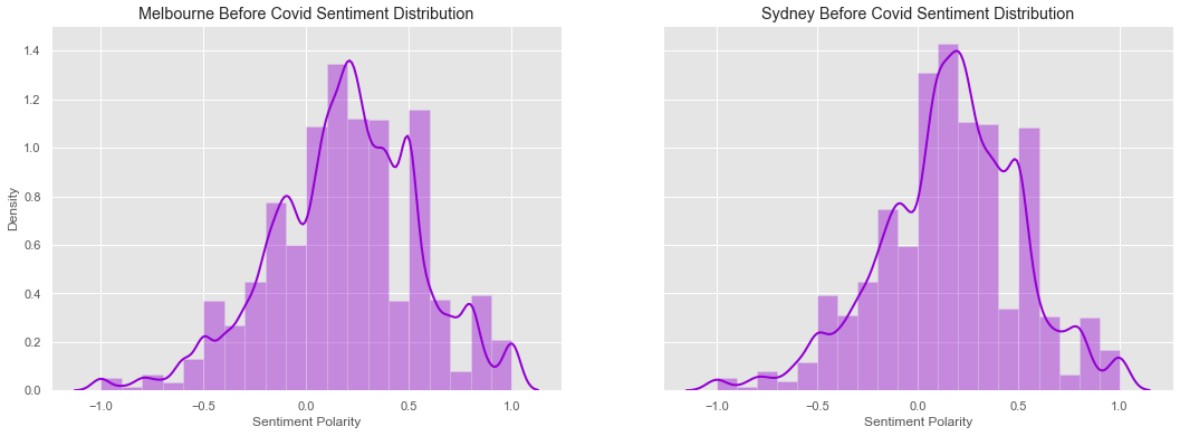

Figure 16: Sentiment polarity distribution before COVID-19 for Melbourne and Sydney

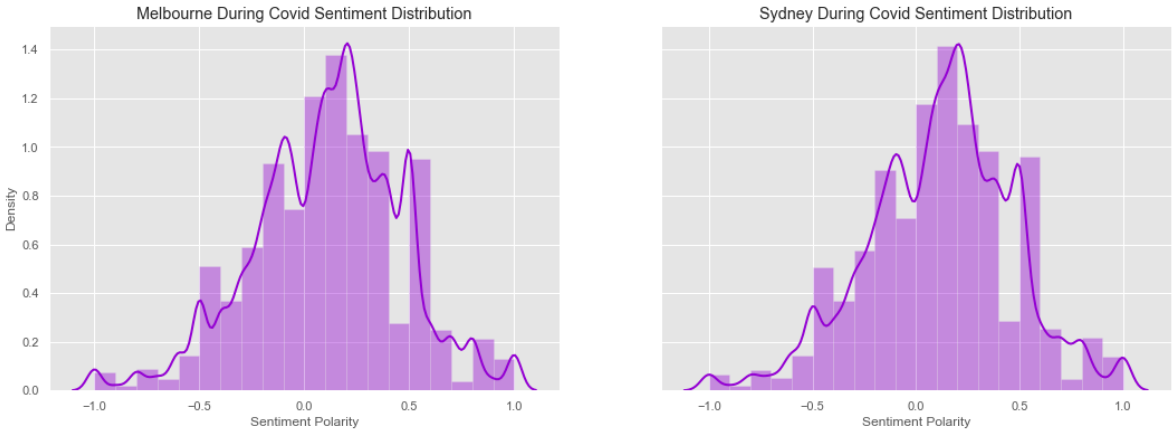

Figure 17: Sentiment polarity distribution during COVID-19 for Melbourne and Sydney

This is one of the indications that the citizens of Melbourne have very robust mentality, and even during catastrophic periods, people continue to remain very positive. The positivity may be due to many wonderful things Melbourne has to offer.

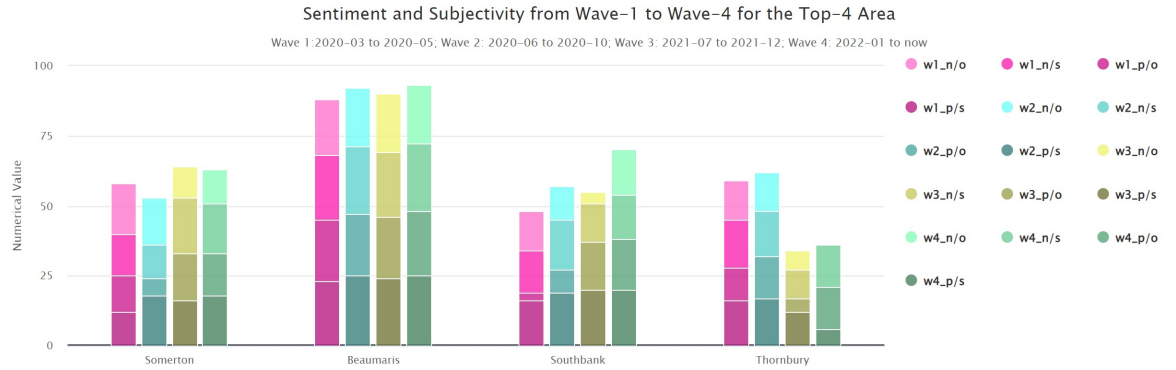

Figure 18: Sentiment polarity distribution before COVID-19 for Melbourne and Sydney

We have conditionally selected four representative areas of Melbourne as shown in Figure 18, namely Somerton, Beaumaris, Southbank, and Thornbury. Somerton is a seaside suburb of Adelaide in South Australia. Beaumaris is located on Port Phillip Bay, which is a suburb nearby Melbourne's central business district. Southbank is 1 kilometer away from the Melbourne business district and it was formerly a mostly industrial area. Thornbury is beside Melbourne's central business district and it is universally understood to be a demographic and commercial satellite of Northcote. Through the comparison of the four regions, we can find that the emotions of residents living in coastal areas are less affected by the fluctuation of the epidemic. The mood swings of residents living in commercial centers are more pronounced than in coastal areas. The subjective and objective tone of residents' tweeting also followed the above changes. As we could observe from the chart that Twitter message with positive and objective emotions is always a small portion compared with other emotions for all stages of the pandemic. Negative and subjective messages are dominant Twitter messages on the internet, which reflects the physiological condition of the population is low during the pandemic.

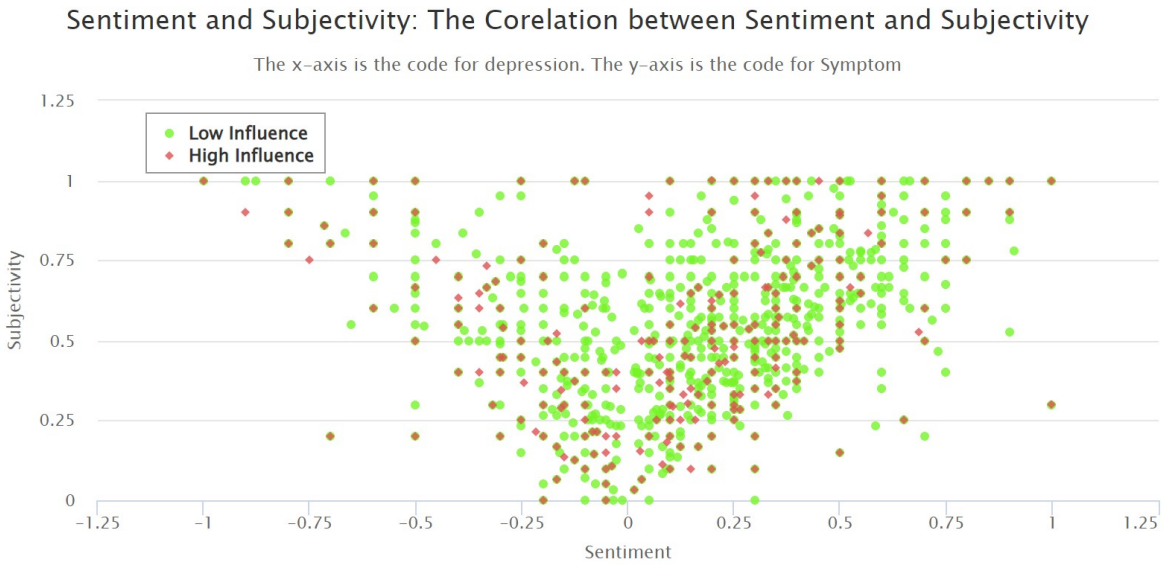

Figure 19: Sentiment polarity distribution before COVID-19 for Melbourne and Sydney

Next, sentiment analysis and subjective and objective trend analysis are performed in Figure 19 for tweets within the city of Melbourne. In this section we used the depression analysis from the psychological paper (Yang & Mu, 2015) in which the sentiment and subjectivity are analyzed in the numerical value and the co-relation between those two category for the textual information is compared. Tweets are roughly divided into two categories based on the number of retweets, high-influence tweets and low-impact tweets. According to the distribution of scatter points, most tweets are distributed in regions with positive sentiment but with neutral subjective characteristics. Other trends cannot be clearly observed through this graph, but at least it shows that even during the epidemic, the life attitude of residents living in Melbourne is optimistic.

To dig the analysis one step further, the team hopes to verify whether the number of mental related tweets is necessarily related to the number of local Covid PCR confirmed cases. Monthly data from January of 2020 to May of 2022 were given, the dot on Figure 20 shows the PCR confirmed cases and the dot on Figure 21 illustrates the mental tweets number in monthly seasons.

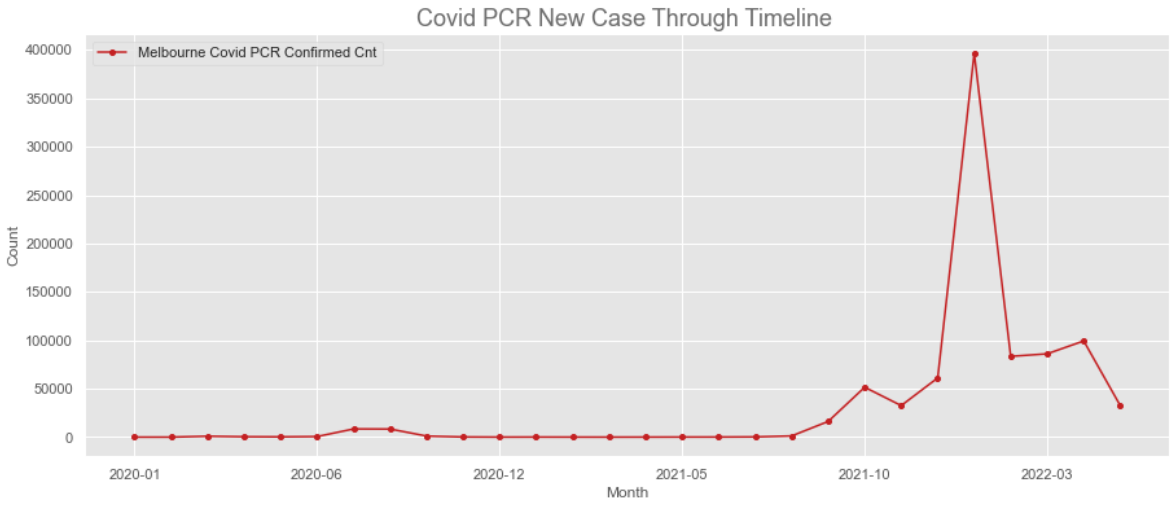

Figure 20: COVID PCR testing cases through timeline

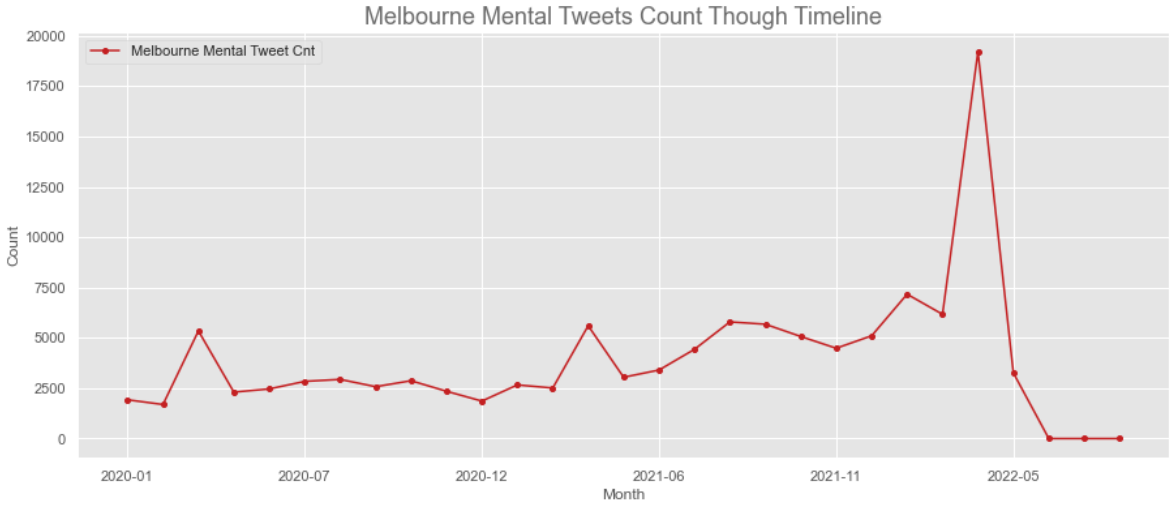

Figure 21: Melbourne mental health tweets count through timeline

Figure 20 shows that at the beginning of 2022, an absolute numerical increase pushes up the whole picture, making the relatively mild outbreak of the previous two years even less obvious. However, combining the data behind the images and the official statement, we can roughly divide the epidemic in the past two years into four stages, which we call four waves of epidemics. The periods of four waves are March 2020 to May 2020, June 2020 to October 2020, July 2021 to December 2021 and January 2022 to May 2022.

As depicted in Figure 21, it is found that the number of tweets posted does show a trend of seasonality. And this number at the beginning of each year will have a sudden increase. This change will last for two to three months and then the total monthly counts will slowly decline back to normal levels. Of course, in the second half of 2021, there is a case where the monthly total is higher than before but maintains a steady growth rate. Given this situation, we believe this is related to the third wave of the epidemic. Therefore, from a timeline perspective, each wave of the epidemic will indeed lead to the publication of emotional tweets at that time. And during the period of epidemic restricting people's travel, online entertainment has become the mainstream way of pastime and venting emotions.

It is important to know the specific topics that Melbourne's citizens are focused on, especially during

the COVID-waves. Using hashtag analysis is an effective method to determine the mentality of the citizens. Hashtags are associated with the key topics that citizens have in their mind. If the hashtags are associated with political turmoil, diseases, or economic issues, then it is representative of negative mentality. If the hashtags are more related to entertainment and sports, this would be associated with positive mentality.

Thus, the information everyone is paying attention to in each wave of the epidemic has become the new focus. And the top ten hashtags during the first COVID-19 wave is shown in Figure 22 below.

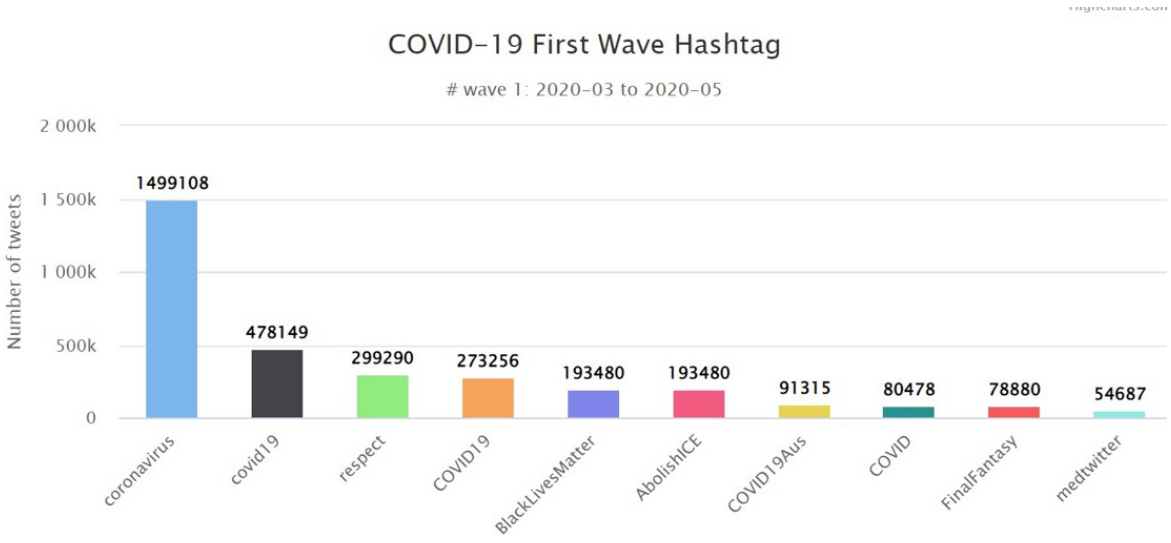

Figure 22: Top ten hashtags during the first wave

The first wave of the epidemic in Melbourne came relatively late compared to cities in other countries, and the situation was not serious due to strict government policies. Even so, with the implementation of the policy of lockdown and the mask order, residents still have a high degree of attention to the hashtags related to the "coronavirus". In preventing the spread of a new virus, residents also cooperate well with local government policies. This indicates that the resident's sense of trust in the Melbourne government is sufficient. At the same time, it is noticed that the hashtag "BlackLivesMatter" is also in the top ten. The importance of human rights can be reflected from this chart. Five out of the top 10 hashtags are pandemic related indicating that the mentality of the people are somewhat affected by COVID-19.

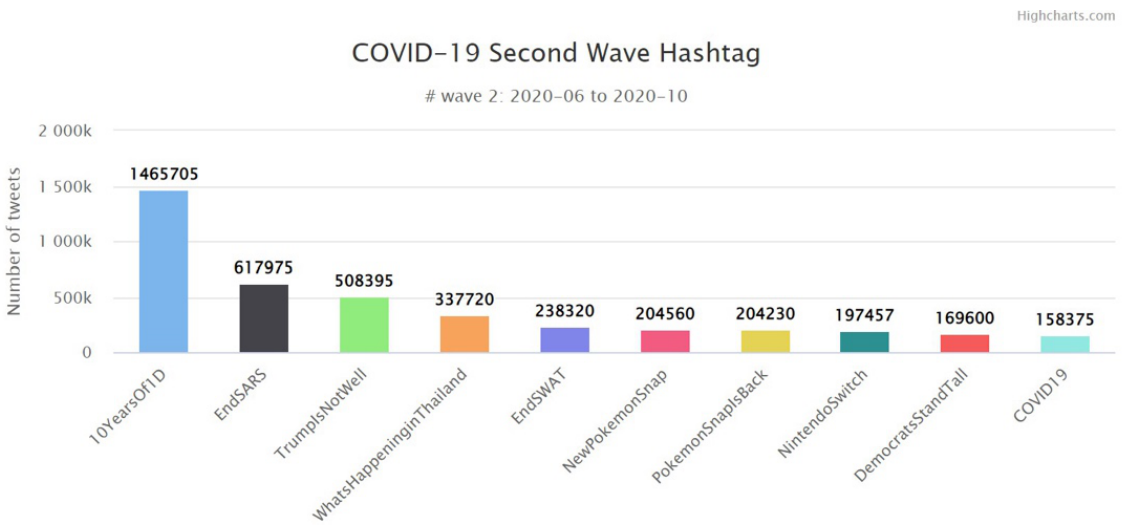

Figure 23: Top ten hashtags during the second wave

As for the second wave, shown in Figure 23 above, the gradual relaxation of the lockdown policy and the epidemic situation that did not become serious have brought residents the idea that the epidemic is about to end. So, the hashtag "EndSARS" is cited by more people. People's attention has gradually shifted to other media channels such as news, audio-visual entertainment, and games. For example, "10yearof1D" returned to everyone's field of vision and captured the number one position of popular hashtag during this period. The popularity of the "PokemonSnapIsBack", a hashtag of mobile game, has also rushed into the top ten. While entertaining, everyone can't wait to go out and enjoy outdoor life. During the second wave, only 2 out of 10 top hashtags are related to the pandemic. People are less worried about the pandemic comparing to wave 1, and more people are going back to their normal mentality and start enjoying various entertainments.

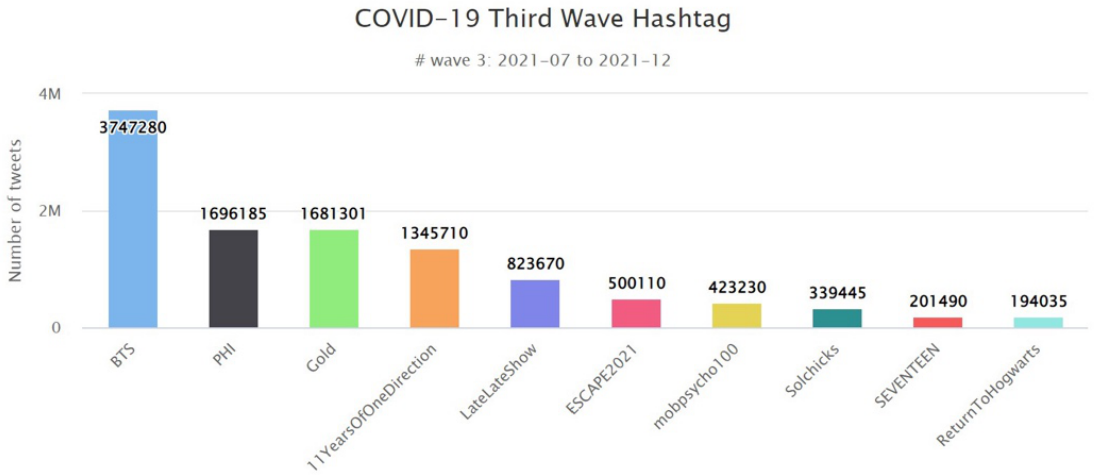

Figure 24: Top ten hashtags during the third wave

As shown in figure 24, after half a year, the third wave began, and the local epidemic situation began to gradually become serious, and the number of PCR diagnoses in the month exceeded the previous period. But the interesting phenomenon is that everyone's attention to the pandemic has disappeared. None of the top 10 hashtag is even related to the pandemic, but rather dominated by K-pop bands and

Hallyu culture. This indicate that people are mainly immersed in fun and entertainment, resulting a positive mentality. It is a very strong indicator that even during middle of the pandemic, the people of Melbourne are very happy. Most people have entertainment related topics in their mind, suggesting that Melbourne is a very livable city.

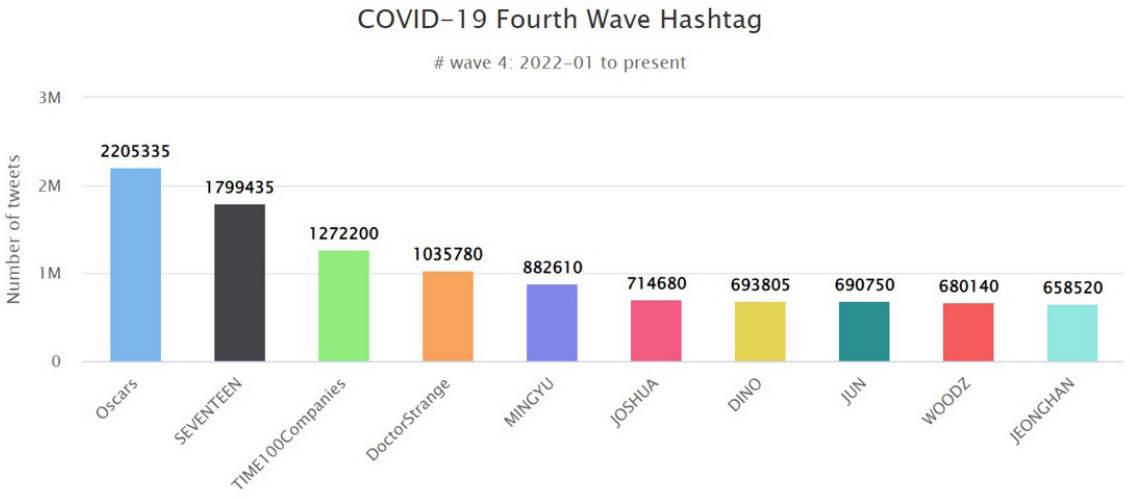

Figure 25: Top ten hashtags during the fourth wave

As shown in figure 25, With the lifting of all government restrictions, the fourth wave of the pandemic is unprecedentedly severe. However, the weakening of the virus' toxicity and the fact that it lasted for more than two years has made people tired. More and more people choose to go back to their normal life. Similarly, as to the third wave, the number of citations in the hashtags of "COVID" has disappeared from the list. People are more interested in entertainments such as the Oscars, and K-pop bands rather than afflictive topics. Again, this stipulates that people are very happy living in Melbourne.

Another indicator to examine people's mentality in the city during the pandemic is by evaluating the number of people on the streets during different COVID waves. Figure 26 below depicts the map of various regions in Melbourne and the red dots represent various sensor locations that count the number of people on the street recorded by the sensor for one hour period.

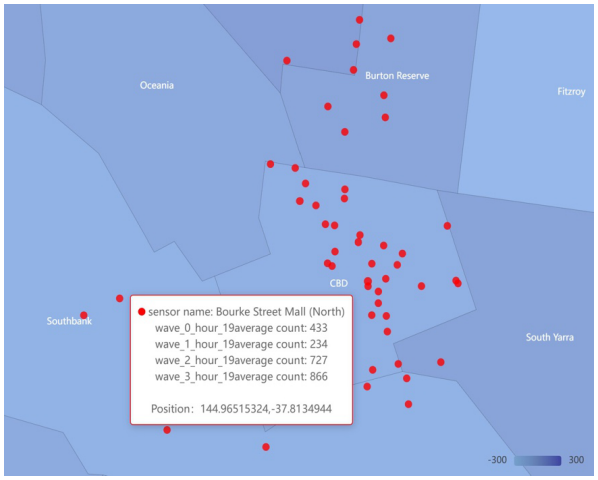

Figure 26: Average number of pedestrians for one hour period during various COVID waves

As shown in figure 26, the red dots show the camera locations and the number of people during the first wave, the second wave, the third wave and the fourth wave of the pandemic. Those camera information are collected from the "city of Melbourne" dataset. (*City of melbourne - open data portal: Tyler Data Insights*, n.d.) This analysis methodology was adopted from the modern crime data analysis paper, including (Gerber, 2014) and the (Ristea et al., 2017), in which the number of pedestrians was analyzed inside the paper before the crime happens and after the crime happens. We use the similar method to compare the number of people on the street during different stages of the pandemic. It has shown that during the second wave, the number of people on the street is at minimum, due to lockdowns and other restrictions. But by the third and fourth wave, there has been a huge increase in the number of people outside. This show that by the third and fourth wave, the city has lifted most of the lockdowns, people are free to go out and socialize in restaurants, bars and enjoy all the culture and entertainment the city offers. The fact that more people are going out is another indication that they are happier and have a positive mentality.

The figures 27 shows the hashtag timeline for some of the most popular hashtags on Twitter and those figures reveal the pattern through the timeline. As shown in figure 27, we could observe that the COVID-19 topic start at a very high position at around 3500 k and the amount of related topics drops extremely quick in just one month and it gradually disappeared and the popularity of this hashtag dies out in the next couple months. In figure 28, after we filter out the COVID-19 topic, we could observe that the *auspol* has a small raise after 5 month of the outbreak. The main reason is because the Australia government just released the announcement the decision to close the border and the hashtag *auspol* is the representation for fake news, which means there were a lot of rumors spread at the time about the false information about the policies. Additionally, we could also observe that *auspol* raises a lot after 6 month of the outbreak. The reason is because the people has already get rid of the shadow of the pandemic and the fake new get popular again for entertainment purposes. As shown in figure 29, the popular topics raise and fall in the following months and the changes does not that obvious as the other topics discussed above.

Finally, a word cloud analysis was generated to illustrate the top topics found in Tweets from the four different cities the team has selected. Tokenization and lemmatization were performed to treat the Tweets, where the key words are extrapolated and counted for their frequency. After counting all the words, the algorithm in the package determines the importance of the words appeared and then assigns a weight to each word. The higher the coefficient is reflective of higher importance. And those words with higher importance and frequency appear the largest in the word cloud.

Figure 30 depicts the main topics from Melbourne. Figure 31 shows the top topics from Sydney. Figure 32 shows the important issues from Brisbane and Figure 33 shows the most concerning topics

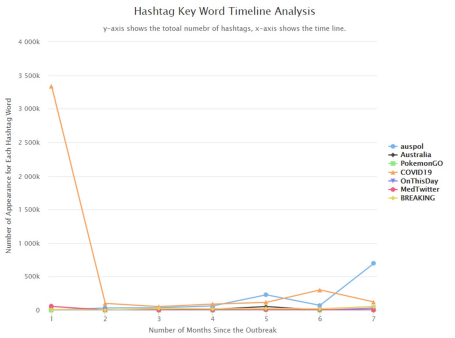

Figure 27: Hashtag Timeline without filter

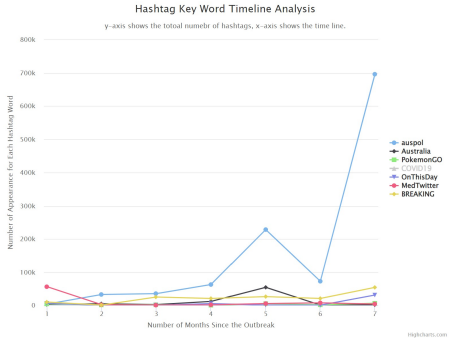

Figure 28: Hashtag Timeline without COVID

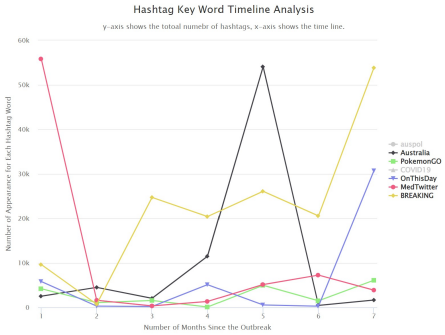

Figure 29: Hashtag Timeline without COVID and auspol

in Adelaide.

Figure 30: Word cloud of Melbourne

Figure 31: Word cloud of Sydney

According to the word clouds above, it is clear that ‘people’, ‘event’ and ‘media’ are the top topics across all the four cities. However, Sydney, Brisbane, and Adelaide have few more negative topics than Melbourne. For instance, Sydney has many negative words such as ‘covid’, ‘avoid’, ‘failure’, and ‘confirmed’. For Brisbane, ‘covid’, and ‘confirmed’ are modest size topics. Melbourne, on the other hand, have the least number of negative topics and they are in the smallest size which indicative of lowest importance. The word clouds is another features that suggest people in Melbourne are very happy and have a positive mentality, and the city itself compares itself slightly better than Sydney, Brisbane, and Adelaide in terms of mental health and livability.

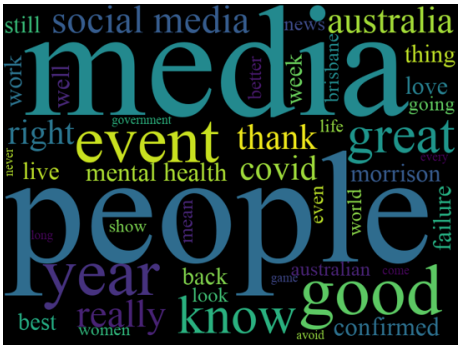

Figure 32: Word cloud of Brisbane

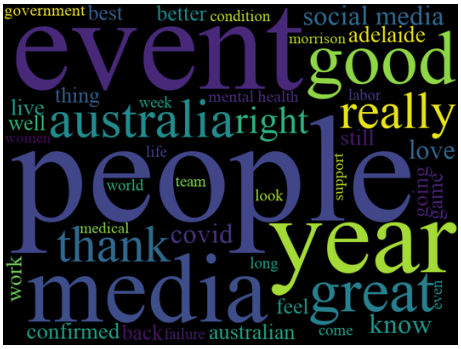

Figure 33: Word cloud of Adelaide

7.3 Issues & Challenges

7.3.1 Bias from random sampling.

Since we randomly sample and crawl Twitter data, in order to prevent duplicate data, a specific group of users has become our target users. Combined with the above tag analysis results, it is not difficult to see that the age group of our target users is relatively young. The fact that Hallyu dominates the trend is well explained. Expanding the sample size and broadening the range of target users will further improve our analysis results if there are sufficient time and space costs.

7.3.2 Machine processing sentiment analysis has inherent drawbacks.

In many cases, it is difficult for machines to accurately distinguish the emotions expressed by humans, such as "Zzz" expressing exhaustion or other tweets expressing emotions using emojis. For the above two situations, most solutions are to ignore this data, which also brings errors to the analysis results to a certain extent. Of course, we can try our best to avoid the above situation in terms of data preprocessing, such as replacing some abbreviations with original words to facilitate the scoring of machine sentiment analysis.

7.3.3 Sentiment analysis is too objective in scoring words.

The scoring of emotional words by natural language processing is fixed and objective. Thus, implicitly lyrical users and even most users get relatively low scores, while extroverted users get high scores. The occurrence of this situation will make the emotional distribution more neutral, making it difficult to observe the differences among reference objects, which brings difficulties for subsequent analysis.

8 Data Visualisation

8.1 Back-end data interface

8.1.1 Flask

Flask is a backend server based on the python framework, which is compatible with the Vue framework. The two main libraries used from the Falsk are the *"jonsonfy"* library and the flask Cross-Origin Resource Sharing (CORS) library. The CORS library is the skeleton used for starting the backend application. By using Flask to develop the back-end server, we can benefit from the following advantages:

1. Developing a system with Flask is simple. Flask provides us with a convenient interface with python, which means we could use all the python library utilities and the other libraries. Using the Flask library inside the python framework enables us to get information from the CouchDB easily.

2. We can flexibly make modifications to the existing system. Since the data are from some living documentation such as Aurin, City of Melbourne, and Twitter data, we need the flexibility to manage those living data within the back-end and provide a living port to the front-end. However, the Flask library is different from other back-end services such as fire-base and spring-boot, whose main focus is on the user management system. The logging system is not required in this project and our main goal is to exhibit data visualization, so the Flask eliminates the unnecessary logging system from our system while maintaining some of the most convenient features as passing the data.

3. There are very few parts of Flask that cannot be easily and safely altered because of its simplicity and minimality. Flask can be thought of as a micro framework being slightly more "low-level" than Django. There are fewer levels of abstraction between you and the database, the requests, the cache, etc. So performance is inherently better from the start. Thus a better performance can be achieved.

4. The Flask has a manageable budget that provides the user with forever free access which could essentially lower the budget while giving us the host for the back-end.

However, there can be several disadvantages to using Flask.

1. Firstly, Flask is a mini web serving framework, it doesn't provide enough scalability for larger projects and is comparatively slower than others. You don't get enough flexibility like in Django. However, our project does not need scalability on the back-end and all the processing and big data management happening within the CouchDB with the MapReduce functions. The back-end only needs the useable routers for the front-end to access.

2. Secondly, As Flask is a mini web serving framework, it is the best suited for short routes. We just use routers to get the required resources across the web. In the future, if enough time is provided, we could use the multi-threaded back-end server to provide a shorter access time for the front-end. Nevertheless, the front-end webpage right can work for a reasonable amount of time and the web page's fresh time is within one second, which is more than enough for demonstration and data visualization purposes.

3. Thirdly, the main function used for the front-end and back-end connection is the *"app. route()"* function with an identifiable and unique route for each function. The main scheme and design idea is to allocate every back-end function with one router for the front-end to access. The data format passed in between the back-end and front-end is in the python dictionary. The specific data format needs the CouchDB developer and the front-end developer to check concurrently to identify the connection protocol. The final dictionary which fills with data is converted to JSON and then passed to the front-end to unpack.

```
@app.route("/MentalTimeLine")
def Chart_MentalTimeline():
    data = {}
    dictList = data_mental_timeline
    data['mental'] = {}
    data['mental']['_0'] = dictList[0]['data']
    data['mental']['_1'] = dictList[1]['data']
    data['mental']['_2'] = dictList[2]['data']
    data['mental']['_3'] = dictList[3]['data']
    data['mental']['_4'] = dictList[4]['data']
    data['mental']['_5'] = dictList[5]['data']
    data['mental']['_6'] = dictList[6]['data']
    data = jsonify(data)
    return data
```

Figure 34: Back-end router demonstration

8.1.2 Axios

Back-end Connection:

When developing projects using the Vue.js framework, we will often send Ajax requests to the server-side interface, in the development process, we need to further encapsulate Axios to facilitate the use in the project. Axios is not a new technology, it is essentially an encapsulation of the native XML Http Request, which can be used in the browser and node.js HTTP client, except that it is based on Promise. The Axios application is written in JavaScript and stored inside the API category as the global file for all other view files to use. The whole application is based on the initial Axios HTTP request to the back-end server and gets the promise from this server. For each back-end router function, a new catch function is defined for the Axios. This gets and catches function takes the JSON data from the back-end and enables the front-end to directly call this function within the High-chart or E-chart files, and the catch function catches any errors during this transmission process. As an example the local back-end serves address for the Flask is *'http://127.0.0.1:5000'*, which is the base URL for the front-end. The *"GetHealthRelatedTopicTrend()"* function gets the data from the *"/HealthRelatedTopicTrend"* route.

```
// api.js
import axios from 'axios';
// create an axios instance with default options
const http = axios.create({ baseURL: 'http://127.0.0.1:5000' });
```

Figure 35: Axios request from the back-end

```
export function GetHealthRelatedTopicTrend() {
  // then return the promise of the axios instance
  return http.get('/HealthRelatedTopicTrend').catch((e) => {
    // catch errors here
      console.log(e);
  });
}
```

Figure 36: Axios Get and Catch function

Front-end Connection

Within each *Vue* file, the function from the Axios is needed to be imported from the API category, which means this function can only be used locally within the *Vue* file. As an example, the *"GetDepressChart"* is a function written in the Axios function system and this function is imported from the Axios JavaScript file and then we could use its information and put it into the Highchart.

```
import Highcharts from 'highcharts';
import { GetDepressChart } from '../api/api';
```

Figure 37: Front-end import function

Another function that plays an important role is the response function, this function waits for the response from the Axios function system and assigns values to the local *Vue* variables. Within this function, we first implemented a log function for debugging purposes. As mentioned before, the Axios function system uses the Promise as the transmission media and the response function on the front-end receives this *"Promise"* package the main job we need to do is to unpack this promise function and give the data it contains to all of the local variables.

```
mounted() {
  GetDepressChart().then((response) => {
    console.log(response)
    this.high = response.data.high;
    this.low = response.data.low;
    this.first_wave = response.data.first_wave
    this.second_wave = response.data.second_wave
    this.third_wave = response.data.third_wave
    this.fourth_wave = response.data.fourth_wave
    this.displayHighCharts();
  })
  // this.displayHighCharts();
},
```

Figure 38: Front-end response function

After the local variables inside the Vue file receive this unpacked function, this data is transferred to the desired position within the Vue export functions. This export function contains the *Highchart* embedding system and as shown in the graph, they part is the input port for the *Highchart* function and it takes this data into the Web Page.

```
{
   name: 'coronavirus',
   y: _this.first_wave._1,
   drilldown: 'coronavirus',
},
```

Figure 39: Front-end response function

8.1.3 Issues & Challenges

There are some issues and challenges happening during the implementation for the back-end and by learning about those issues and problems we could gain further experience in the future designing process.

1. The first problem happening in the back-end is deciding which library to use and we changed back and forth among several different application hosting libraries. The final decision is to use Cross-Origin Resource Sharing (CORS) library as the host because CORS is the original application ruining service for the Flask and it inherits properly from the Flask system framework. The main advantage of CORS is it supports all kinds of routers and functions which is suitable for our multiple data formats. By using CORS we could easily start running the application and add different routers to the back-end functions. One of the most important experiences we learned is the router defining procedure. In the development phase, we tried to use one router for multiple back-end functions; however, this mechanism generally leads to conflict among different functions within the same router address. Later on, we switched to a different strategy. We use one identical router for only one back-end function and each back-end route corresponds to only one Axios function. Then this Axios function will be imported into the *Vue* file and get the local variables from them.

2. The returning format from the back-end was another obstacle we encountered during the development. The Axios library only takes in the data form that's in the dictionary JSON format and any other form of data can not compromise in the system. Depending on the specific data format we would like to pass to the front-end, all the data need to be stored in a dictionary first and transmitted through Axios in the form of promise. The Flask library solves this problem by providing the *"jsonfy"* command which could turn the final dictionary into the JSON format.

The place to start receiving the information in the front-end framework is also another decision to make. The information from the back-end needs to be received before the rendering happens in the front-end and we encountered some problems when the figure of *Highchart* can not be loaded when connecting to the database. The main problem is the position of the receiving response function in the front end. The problem was solved perfectly by removing the responsive function into the *"mount"* part of the *Vue* file and embedding the *Highchart* function into the end of the "mount" function. The resultant graphs can be rendered together with the data received.

3. The final challenge we need to solve is the slow loading of the frontend graph when the backend is connected to the CouchDB. In the beginning, when the back-end was trying to connect to the CouchDB, this connection function was put inside of the back-end router function, which means every time the front-end needs to get information from the route, the request to receive information will be processed by CouchDB one more time, which leads to the slow loading of the back-end and the front-end graphs. To solve this problem, the CouchDB connection functions were defined outside of the route function as a global variable, which means the information is only needed to get from the CouchDB all at once. This refining of the system greatly improves the system performance and leads to a much faster speed.

8.2 Front-end user Interface

8.2.1 Dependencies

Vue.js

Vue.js is a set of incremental JavaScript frameworks for building user interfaces. Unlike other heavy-weight frameworks, *Vue* is designed for incremental development from the bottom up, and its goal is to implement responsive data binding and combined view components through the simplest possible API. *Vue* is a combination of frameworks that consisted of three main languages, including CSS, JavaScript, and HTML templates. Cascading Style Sheets (CSS) is the main popular style language used today for designing the modern website and other related markdown documents this language can decide the presentation format for the end-users and we use this language extensively for the website style inside of our website designing. JavaScript (JS) is the main language used today for the inner logic design for the website and the advantage of JS is the extensibility of this language and the related error handling communities because almost all the website developers nowadays use JS for the website designing and data visualization. HyperText Markup Language (HTML) language is the main language used for designing textual information and constructing the framework of the website and we used this language for table formatting purposes in our project. The *Vue.js* platform provides us with an easy-to-manage interface to combine those three components all together to achieve the data visualization purpose. In our project, the *Vue* project structure consists of the following components. The assets component, where we put all the related pictures information into this category. The components category is where we define the homepage headers and the homepage pictures. The router folder contains the *index.js* file, which is a JS in which we define all the router addresses for our header menu.

```
import DepressionAnalysis from '../views/DepressionAnalysis.vue'

{
    path: '/depressionanalysis',
    name: 'Depression Analysis',
    component: DepressionAnalysis,
}
```

Figure 40: index.js front-end router

The view folder contains all the webpage components for different charts, and the main structure for each of the view folders contains three sections. The template section is where we define all the IDs for different containers and we could define other textual material inside of this template category.

In our file structure, some of the table text information is also defined inside of this category and the main language used in this section is the HTML language. The next section is the script section and the main language used in this section is the JS language. In this section, we could import all kinds of libraries and the Highchart embedding. The main body of the script section is the "export default" function and all other information is contained in this main body function. There are two subsections contained inside of the main body, which is the method function and the mount function. The method function is where we define all the related Highchart functions and the mount function is where we start running all the methods defined in the main body. The third part of the Vue file is the style section and in this section, the main language used is the CSS language. We define all the style functions inside of the style section. A brief description of the view file structure is shown in Figure 41.

```
<template>
  <figure class='highcharts-figure'>
    <div id='container'></div>
  </figure>
</template>

<script>
import Highcharts from 'highcharts';
import { Somechart } from '../api/api';

export default {
  mounted() {
    Axiosfunction().then((response) => {
    console.log(response)
    this.data = response.data
    this.displayHighCharts();
    })
  },
  methods: {
    displayHighCharts() {
        const _this = this
        Highcharts.chart('container', {
    });
  }
 }
<style scoped>
.highcharts-figure,
#container {
  height: 400px;
}
.spacer {
    height: 20px;
}
</style>
```

Figure 41: Example of the .vue file structure

Node.js

Node.js is originally a web front-end language, Node.js makes JavaScript a server-side scripting language. Node.js has a built-in library of functions for handling network requests and responses, which means it comes with its own HTTP server, so there is no need to deploy an additional HTTP server, just import it directly when you use it. By starting the Node.js server, clients can fetch data from the Flask endpoint, making it easy to communicate between the front-end and back-end. node.js further enhances JavaScript's ability to access files, read databases, and access processes to be competent for back-end tasks. Node.js applications are single-process Node.js applications are single-process, single-

threaded, but support concurrency through events and callbacks, and improve the performance of our project. Node.js is the embedding that makes our frontend capable of extending all kinds of functionalities. Node.js is a package collection JavaScript running environment where we could different open-source packages into this system. The Node.js works with the npm package unzipping command or the yarn unzipping command to extract all the packages installed into our Vue.js system. The two main node packages we extended inside of our project are the Highchart node and the Echart node and those two chart packages are used extensively inside our project.

High-chart

High-chart is a chart library written in pure JavaScript. High-chart is very easy to configure and supports zoom for multi-dimensional charts. The main charts used in our project use the High-chart template.(*Simply visualize*, n.d.) High-chart has different chart format templates to use including the pie chart, bar chart, line chart, and scatter plot. The main advantage of the high chart is that all the chart designs are easily extensible and the chart designs are interactive. The chart generally has tool-tip features which mean if the end-users put the cursor on a specific char component, the related information about this section will appear and the information includes the filed name and the data values. Another feature of the High-chart is the interactivity the chart. We could easily turn on and off the filters of the chart to choose which information we would like to display on the dashboard.

E-chart

Charts is a pure JavaScript icon library, the underlying reliance on the lightweight Canvas class library render, based on the BSD open-source protocol, is an excellent visual front-end framework. The Echart is the main framework we use for the interactive map design and we use the interactive map framework inside of our project. Echart enables multiple polygon inputs and it enables the polygon color scheme to change. The map also enables the scatter plot points to overlap on those colored polygons which give us more information about facilities inside each suburb. The chart information was collected from the AURIN database and the polygon information is loaded from CouchDB from the AURIN file.

Jquery

Jquery is a programing language inside of the JavaScript library and the main advantage of Jquery is the way it handles the events. It provides us with a much easier way to handel all kinds of events including listening events and click events. In our project, the main usage for Jquery is to enable the animation of the Echart to implement the interactive map polygons. In other web pages design, the Jquery is not used that extensively and the normal Highchart library is used for those chart designing.

9 Conclusion

In conclusion, this paper reports the system's structure, research object and results, research technologies, and issues&challenges. Firstly, we set up the *Ansible* system, designed the system structure, and set up four different instances on the Melbourne Research Cloud (MRC). Moreover, we designed several scenarios for research. After that, we crawled data through $TwitterAPI$ and $AURIN$ and transmitted data to CouchDB. The last step is performing these data on the website interface. Secondly, we analyze tweets and $AURIN$ data in CouchDB. Compared to these cities' life quality in crime data, camera data, and mental health data. Then we got a result that there has been a slight rise in the negative sentiments, but the overall polarity shape is still very similar to before COVID times. Finally, We also solved issues and challenges, including obtaining a large number of data and deleting duplicated data when harvesting them from Twitter and existing datasets. In addition, we designed views and MapReduce to solve problems in the CouchDB database, for example, counting the same keys value. In the front-back-end designing, we solved problems of deciding which library to use, returning format, and the slow loading of the front-end.

References

(n.d.).

Aurin home. (2022, Apr). Retrieved from https://aurin.org.au/

Banda, J. M., Tekumalla, R., Wang, G., Yu, J., Liu, T., Ding, Y., ... Chowell, G. (2021). A large-scale covid-19 twitter chatter dataset for open scientific research—an international collaboration. *Epidemiologia*, *2*(3), 315–324.

City of melbourne - open data portal: Tyler data insights. (n.d.). Retrieved from https://data.melbourne.vic.gov.au/

Crime Statistics Agency Victoria, S. G. o. V. (2022, Mar). *Homepage.* Author. Retrieved from https://www.crimestatistics.vic.gov.au/

Culotta, A. (2014). Estimating county health statistics with twitter. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. doi: 10.1145/2556288.2557139

Gerber, M. S. (2014). Predicting crime using twitter and kernel density estimation. *Decision Support Systems*, *61*, 115–125. doi: 10.1016/j.dss.2014.02.003

Giles-Corti, B., Vernez-Moudon, A., Reis, R., Turrell, G., Dannenberg, A. L., Badland, H., ... et al. (2016). City planning and population health: A global challenge. *The Lancet*, *388*(10062), 2912–2924. doi: 10.1016/s0140-6736(16)30066-6

Higgs, C., Badland, H., Simons, K., Knibbs, L. D., & Giles-Corti, B. (2019). The urban liveability index: Developing a policy-relevant urban liveability composite measure and evaluating associations with transport mode choice. *International Journal of Health Geographics*, *18*(1). doi: 10.1186/s12942-019-0178-8

Ristea, A., Langford, C., & Leitner, M. (2017). Relationships between crime and twitter activity around stadiums. *2017 25th International Conference on Geoinformatics*. doi: 10.1109/geoinformatics.2017.8090933

Sahealth - hospital locations (point) - aurin data. (n.d.). Retrieved from https://data.aurin.org.au/dataset/locationsa-locsa-southaustraliahospitallocations-na

Simply visualize. (n.d.). Retrieved from https://www.highcharts.com/

Twitter api documentation — docs — twitter developer platform. (n.d.). Twitter. Retrieved from https://developer.twitter.com/en/docs/twitter-api

Victorian covid-19 data. (n.d.). Retrieved from https://www.coronavirus.vic.gov.au/victorian-coronavirus-covid-19-data

Yang, W., & Mu, L. (2015). Gis analysis of depression among twitter users. *Applied Geography*, *60*, 217–223. doi: 10.1016/j.apgeog.2014.10.016

Zhang, S., Sun, L., Zhang, D., Li, P., Liu, Y., Anand, A., ... Li, D. (2021). The covid-19 pandemic and mental health concerns on twitter in the united states. doi: 10.1101/2021.08.23.21262489

A Collaboration

Name	Responsibilities
Qi Li	Deployment of cloud infrastructure Setup of CouchDB Implementation of connection between frontend and backend Data collection
Yuheng Guo	Implementation of web-based visualization fronted Implementation of connection between frontend and backend Backend setup CouchDB Data collection
Zhaoyang Zhang	Data analysis Setup of CouchDB Mapreduce
Zhangyu Wei	Implementation of connection between frontend and backend data collection Setup of CouchDB
Xiaohan Ma	data collection data analysis Orchestration of application