COMP90024 2021S1 Project 2

Team 49

Agrim Binjola (1154479)

Jiashuai Yu (954064)

Vignesh Lakshminarayanan (1032043)

Yue Yang Ho (860212)

Zhuolun Wu (954465)

ABSTRACT

This is the Team 49 report for COMP90024 Project 2. In this project we analyze twitter data with various Australian data sets utilizing Melbourne Research Cloud.

Contents

1. Introduction

In this project, we endeavored to develop a cloud-based solution that utilizes several virtual machines (VMs) across the UniMelb Research Cloud (MRC) for analysis on different scenarios pertaining to COVID-19. These scenarios make use of tweet data harvested live through Twitter APIs, as well as data sourced from AURIN. The questions we wished answered through our three selected scenarios are:

1. Is there a correlation between unemployment rates and the number of COVID-related tweets per population across Australian states?
2. Is there a correlation between the number of COVID-related tweets and mortality rates across SA4 regions?
3. How does the number of vehicle crashes in VIC and NSW change along the pandemic timeline?

To achieve our goal of answering these questions, we used Python to develop a live tweet harvesting program, and programs to upload select AURIN data to a CouchDB cluster, which is the database of choice for this project. We also developed a frontend application using React.js and the Plotly plotting library for analysis and data visualization. Lastly, we utilized Docker for application containerization and Ansible to automate the deployment of all our applications.

2. System Architecture and Design

2.1 Allocation on instances

In our system, there are four instances that get launched, with three of them forming a CouchDB cluster to store tweets harvested through Twitter Developer APIs, and geolocation related data sourced from AURIN and The Bureau of Infrastructure and Transport Research Economics (BITRE) (as shown in Figure 1).

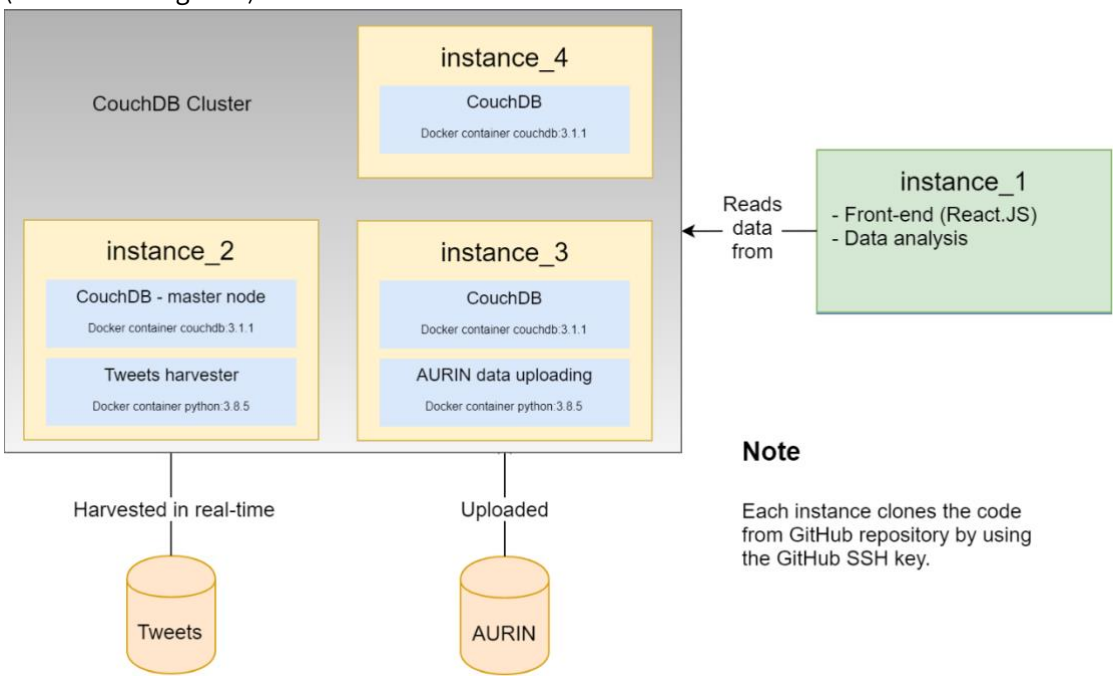


Figure 1 Architecture diagram

The CouchDB database receives tweet data from instance 2 (which is its master node), and AURIN/BITRE data from instance 3. Each database uses MapReduce functions for creating views, and these views are read by the React application in instance 1 for data analysis.

The architecture is designed as above because we estimated CouchDB would require the most computation power. Since CouchDB provides MapReduce functions, which handle the process of data aggregation, that takes most of the grunt work on analyzing data away from the frontend application.

As time goes by, the load on CouchDB would also increase as the number of tweets would grow dramatically. Therefore, we decided to allocate three instances with 2 cores and 9 gigabytes of RAM for CouchDB.

Since the instance where the frontend application is deployed on does not require much computing power and we could not fully customize instance flavors, the last instance with 1 core and 4.5 gigabytes of RAM is left for the frontend application.

2.2 Melbourne Research Cloud (MRC)

In this project, all systems are deployed on the Melbourne Research Cloud (MRC), which offers free on-demand computing resources to researchers at the University of Melbourne. In our case, we were provided with 4 instances, 8 cores, 32 gigabytes of RAM, and 500 gigabytes of storage on MRC. We discuss all the benefits and issues we have noticed regarding MRC during development and deployment.

Benefits

1. MRC provides a convenient solution for cloud computation and remote system deployment, which has an intuitive and clearly structured dashboard for interaction.
2. MRC provides various system images for different running environment (system)
3. Operating on OpenStack platform makes it easy to access via running Ansible playbooks.
4. Instances created can be accessed by multiple users with pre-added key pairs.

Issues

1. Every cloud computing platform has risk of data loss, so this is a general issue not specifically for MRC.
2. Requires campus network or UniMelb VPN access.
3. Insufficient hosts (IP addresses) cause failure on launching instances.
4. The development process is highly dependent on MRC's (NeCTaR) maintenance schedule.
5. Lack of customization for instance flavors. As described in the previous section about instance allocation, after we have chosen three instances for CouchDB, the remaining instance has barely 4 gigabytes of RAM for front-end.

2.3 Ansible

Ansible provides us with the ability to write scripts in the form of Ansible playbooks to automate the deployment of instances and applications.

Benefits

1. Low learning curve makes it easy to use for starter.
2. YAML syntax and playbook structure makes it quite easy to adjust scripts.
3. Ansible uses standard SSH module for connection.

Issues

1. Lack of support on Windows operating system.
2. Lack of graphical interface making it difficult to navigate.
3. Ansible does not have concrete community support.

2.4 Docker

Docker provides us with small and lightweight containers which helps us to deploy our systems on our instances without much concern of configuration on environment and resources needed, as compared to configuring a virtual machine directly and manually.

Benefits

1. Reduce workloads on setup running environment, also eases the pain for debugging environment.
2. Easy to deploy by using Dockerfiles / docker-compose.
3. Docker has a large user base and active community, with massive storage of images in the Docker Hub.

Issues

1. Codes or systems must be debugged before deploying in container, otherwise there is no feedback if the container started successfully and then exited when an error occurs.
2. Running a docker container on MacOS is relatively less efficient than running on our instances.

2.5 CouchDB

CouchDB is a clustered database where data is stored in documents. This is perfect for our system as we need to store large document-based data (i.e. tweets and AURIN data). CouchDB provides MapReduce functions which can be used to easily aggregate document data from databases, creating Views that are easily queried.

2.6 Tweet harvesting program

Detail on this program is on Section 4 where we discuss the finer details of data delivery.

2.7 AURIN/BITRE data uploading programs

We use several python programs to push AURIN/BITRE data to CouchDB. The data is loaded and preprocessed using basic python libraries like json. The relevant data is extracted, the values processed into correct data types and stored into dictionaries before being saved in CouchDB as a document. We also make use of data from different government organizations which are in a CSV format, the python library, pandas, is used to load and clean the data and then saved in CouchDB.

2.8 Frontend application (React.js + Plotly)

The team developed the frontend application for data analysis and visualization using React.js and a plotting library called Plotly. We utilized the SPA (Single-Page Application) nature of React.js to design three views, with each handling one of our three scenarios. A simple navigation bar was developed for ease of access to all three scenarios.

Plotly (https://plotly.com/javascript/) and its React-compatible library was utilized for charting bar and scatter plots. They were easy to integrate with the data that we had. Their readily available functionalities were also more than enough for our analysis.

We did not have a dedicated backend application as data manipulation and analysis was mostly done with CouchDB's inbuilt MapReduce functions. Any other required data manipulation (i.e. reformatting to integrate with Plotly's charting capabilities) were easily performed with JavaScript on the client-side.

3. User guide

3.1. Instances assignment

We allocate a total of four instances with a capacity of 100GB. Except for the first one that will be used to process the frontend, the remaining three will have CouchDB deployed to establish a cluster. Instance 2 will be the master node of the cluster, as well as where the tweet harvester is deployed.

Instance Number	Function/applications	Volume (GB)
1	Frontend	100
2	Cluster Master, Tweet harvester	100
3	Cluster node, AURIN/BITRE data uploading	100
4	Cluster node	100

Table 1: Instances assignment

3.2. Deploy instances

In this step, we use Ansible to set dependencies, security groups, volumes, and images, also dynamically generate instances in MRC.

In the beginning, we install the dependencies required to run the script task (such as Python-pip,

OpenStack sdk and update pip) and get the image we need from OpenStack. Then define the volume required for each instance (these volumes will be used to store data such as tweets) and define the security rules for each instance (We will open all ports of the instance for emergency and convenience.). Finally, all the above settings are used to create four instances.

The table below outlines the details of the instances deployment Ansible playbook.

Hosts	Role	Description
localhost	Instance-common	Install pip, update pip, and openstack sdk dependencies.
	Instance-images	Retrieve all images from openstack and select the right one we use.
	Instance-volumes	Set the volume required for each image.
	Instance-security-groups	Create security groups and their rules
	Instance-creation	Create instances on MRC and save the IP address of each instance to the inventory folder.

Table 2: Instances Deployment

3.2.1 How to run

Locate and go to the directory where launch-instances.sh is situated and run ./launch-instances.sh in the ubuntu system.

3.3. Environment configuration

In this next step, we configure the environment of instances deployed in step 3.2, this contains adding proxies, assignment of volumes, docker configuration, and git initialization.

First, we add proxies to instances to allow them access to network outside LAN of UniMelb. (or it cannot install dependencies from the Internet). We then reboot instances while waiting for a *reboot_timeout: 6000* to make sure proxies work. Then we can start using the instances by installing some dependencies. Second, we mount the volumes defined in step 3.2 to their corresponding instances. Third, to reduce the risk of compatibility, we run our worker program in a docker container, so we need to install docker on our instances and add a proxy for it for the same reason as above. Fourth, to make it easier to get the latest version of our worker program and other codes, we decided to clone our GitHub repository in every instance.

The table below outlines the details of our environment configuration Ansible playbook.

Hosts	Role	Description
instances	environment-installation	Add proxy to instances and reboot, then install

		dependencies.
	environment-mount	Assign volume to each instance.
	environment-setup-docker	Add proxy for docker and then restart docker
	environment-git-clone	Setup GitHub repository in instances.

Table 3: Environment Configuration

3.3.1 How to run

Locate and go to the directory where setup-environment.sh is situated, and run ./setup-environment.sh in ubuntu system.

3.4. Start applications

In this step, we deploy the frontend application on the first instance, then we deploy CouchDB on every instance used for building a cluster. In addition, we run HTTP requests to form the CouchDB cluster. After the cluster is established, our harvesting program is run to begin harvesting tweets from twitter API to our CouchDB cluster. We also pull the AURIN datasets from our GitHub repository and run the relevant Python scripts to upload them to CouchDB.

The table below outlines the details of the Ansible playbook that starts up our applications and scripts.

Hosts	Role	Description
frontend	deploy-frontend	Deploy frontend on instance 1.
backend	deploy-aurin	Pull aurin data to instance from Github.
dbServers	deploy-couchdb	Deploy couchdb on every instance used for building cluster.
dbMaster	deploy-cluster	Create cluster and make this host the master, also active CORS.
	deploy-harvester	Start Tweets harvester program, it harvest corresponding tweets to couchdb cluster we set up.

Table 4: Starting applications

3.4.1 How to run

Locate and go to the directory where deploy.sh situates and run ./deploy.sh in ubuntu system.

4. Data Collection and Delivery

This section discusses the data that we collected, processed, and utilized for the project and its included scenarios. It also includes discussions on how we obtained and stored data in CouchDB, as well as how that data is retrieved from CouchDB to be visualized on the frontend application.

4.1. Twitter Data

4.1.1 Collection

The tweet harvester was developed to get COVID-19 related tweets and store them in CouchDB. The "tweepy" package in python was used for performing this task and CouchDB was used for storing the tweets. The harvester was able to successfully harvest live tweets using the stream API of Twitter and accessing the user timelines. In order to achieve this a Twitter developer account was created, and the required tokens were generated and stored in a python file named "creds.py" and these credentials were then further used for facilitating the tweet harvesting.

4.1.2 Pre-processing

Filtering of tweets based on location and COVID-19 related keywords:
Since the project focuses only scenarios for Australia, the harvester was developed to filter tweets only for Australia. The bounding box coordinates of Australia were used for the filtering. The coordinates used were [113.338953078, -43.6345972634, 153.569469029, -10.6681857235]. The problem statement of the project focused on COVID-19 and as a result, a set of COVID-19 related keywords were chosen (See appendix). These included keywords like "vaccine", "Pfizer" etc. First, the live tweets were harvested and only the tweets which had one or more of the keywords were chosen.

Tackling the duplicate tweets issue and storing relevant tweets:
To tackle the duplicate tweet problem, the "ID" field of each tweet was used to check if the tweet already exists in CouchDB or not and if yes then the tweet will not be stored again. Only information that was relevant for the analysis was stored from the tweet. The information like tweet ID, tweet text, tweet creation date, user details, hashtags included in the tweet, and the place information (including the name of the place and its bounding box coordinates). While storing the tweets it was also taken into consideration that only those tweets which had a 'not null' place attributes value and whose language was English. Apart from just storing live tweets, the user ID was further used to access the user's timeline. A python list was made to store the user IDs and was checked each time a user timeline was about to be accessed and if the user ID already existed in the list, then that user's timeline was not accessed and if the user ID was not found in the list, then tweets from that particular user's timeline were filtered for Australia. The location of tweets from the user in their timeline were checked again as it could have been the

case that the user timeline tweets were made outside Australia. Country code and country code of each of these tweets were checked and the tweet was only stored if that tweet came from Australia. The presence of one or more keywords was also checked and if the tweet fulfilled all these requirements, then its stored else not.

Working around the rate limit:

The "wait_on_rate_limit" attribute of the Twitter API was set to be true so that the rate limit set by Twitter was not violated. The "wait_on_rate_limit_notify" attribute was also set to true. The harvester was successfully able to harvest ~20k relevant tweets at the time of writing. All these tweets were relevant to the scenarios chosen, and all had locations attribute necessary for the analysis.

4.1.3 CouchDB storage

The harvested and processed tweets are stored in a database called 'tweets-db' in our deployed CouchDB as individual documents.

4.2. AURIN/BITRE Data

4.2.1 Collection

The data for scenario 1 and 2 is downloaded from the AURIN portal in a JSON format. Accessing the data has been intuitive and easy, however often there was an error encountered when connecting to Aurin. For Scenario 3 data is obtained from The Bureau of Infrastructure and Transport Research Economics (BITRE), since AURIN did not have the relevant data for the time period.

4.2.2 Pre-processing

The data from AURIN needed to be pre-processed. Any additional information in the data that was not required was removed. The SA4 division of regions was dissimilar to the way location is reported in tweets, we thus combine the SA4 districts in a city into one by using a simple average across various rates. A simple average was used since the relevant information needed to get the true aggregate rate was not available. The data for scenario three was grouped according to the frontend's needs before being stored into the database.

4.2.3 CouchDB storage

The three sets of data for our three scenarios are stored in the databases 'labour-db', 'mortality-db', and 'crash-db' respectively in CouchDB.

4.3. Visualization of data

4.3.1 MapReduce

As mentioned in previous sections, CouchDB provides MapReduce functions to aggregate data and create views for easy querying.

Our group has written separate MapReduce functions for each of our databases ('tweets-db', 'mortality-db', 'labour-db', 'crash-db') to filter, aggregate, and perform calculations on data.

4.3.2 Visualizing data on the frontend

Our frontend application calls RESTful GET APIs to access Views created by our MapReduce functions and the data that they return. However, we still need to perform some more data manipulation in the frontend before we could use them for visualization. The JSON-formatted data returned from the APIs are parsed and further processed into structures understood by the Plotly plotting library.

5. System Functionality

5.1. Scenarios

5.1.1 Relationship between COVID-19 and unemployment rates

5.1.1.2 Description

In this scenario, we analyze whether there is a correlation between unemployment rates and the number of COVID-related tweets (averaged over population) across Australian states. We hypothesize that there is a higher number of COVID-related tweets per population in states that have lower employment rates as we theorize that people are more concerned about COVID-19 if it affects their livelihood (i.e. employment).

5.1.1.3 Why this scenario?

Following our hypothesis, we expect unemployed people to be more concerned about COVID-19 and its effect on their lives. If COVID-19 has little effect one's livelihood or ability to make a living (i.e., employment), then people may potentially not worry about COVID-19 as much as they would if COVID-19 did have an impact on their employment. We perform analysis on actual data to see whether there is a pattern that aligns with our hypothesis.

5.1.1.4 Results and analysis

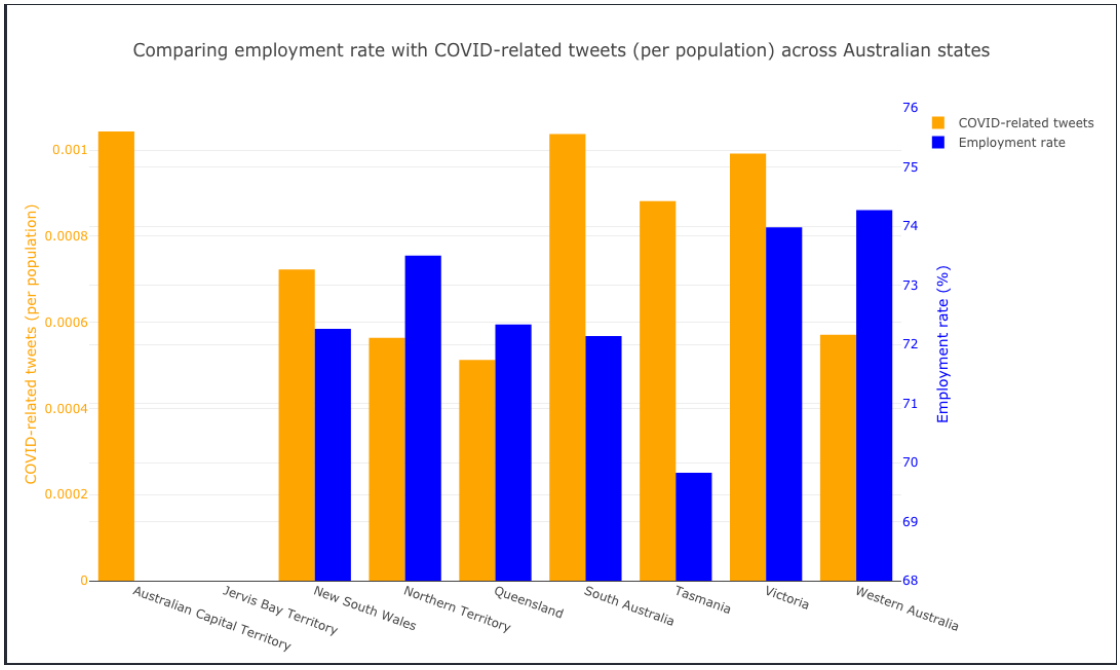


Figure 2: Employment rates and number of COVID-related tweets across Australian states

We graphed a dual bar chart, with orange representing the number of COVID-related tweets and blue representing employment rates. The employment rates were data obtained from December 2020 – still well within the pandemic. Due to the lack of employment data on Australian Capital Territory and Jervis Bay Territory, we ignore those two states.

South Australia, Victoria and Tasmania has the highest number of COVID-related tweets per population. However, their employment rates vary drastically. Tasmania had the lowest employment rates across the states. Perhaps people living there are truly more concerned about COVID-19 and its effect on job security (i.e. having a higher number of COVID-related tweets) since their employment rate was the lowest.

Victoria has the second highest employment rate even though it is one of the states with a high number of COVID-related tweets per population. Victorians may instead be more concerned about the effects of COVID-19 on aspects of life other than employment. Another reason could be that Victoria has a higher population of young adults compared to other states, who tend to use Twitter more than other age groups (i.e. elderly, children)

Most other states seem to exhibit a more neutral behaviour, not explicitly showing signs of aligning with our hypothesis. Apart from Tasmania, Western Australia seems to be the only other state that agree with our hypothesis. It has the highest employment rate but has one of the lowest scores for COVID-related tweets per population. Western Australia is known to have handled the pandemic rather well compared to states like Victoria. That may be a core reason why residents there may be less concerned about COVID-19's effects on employment!

5.1.2 Correlation between COVID-19 and mortality rates across SA4 areas

5.1.2.1 Description

This scenario aims to analyze whether the mortality rates in the SA4 level (see 'SA4' in the appendix) has any effect on number of tweets related to COVID-19. We hypothesize that the higher the number of tweets made regarding COVID-19, the higher the mortality rate in that area.

5.1.2.2 Why this scenario?

COVID-19 is a word wide pandemic that is highly contagious and known to cost lives if undetected early on. Thus, we wanted to see if our harvested data portrays the assumption that people tend to tweet more about COVID-19 if there is a higher mortality rate in their region. This follows the logic of people living in more susceptible regions (say with poorer healthcare in their area) will be more concerned about covid and is effects.

5.1.2.3 Results and analysis

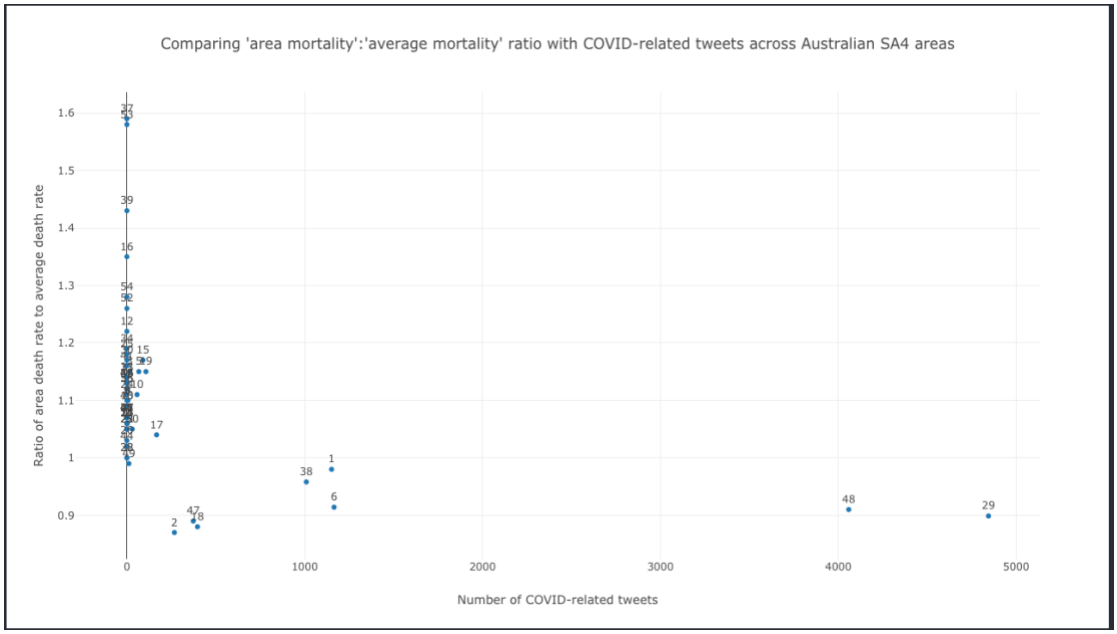


Figure 3: Area mortality : Average mortality ratio vs. Number of COVID-related tweets across SA4 areas

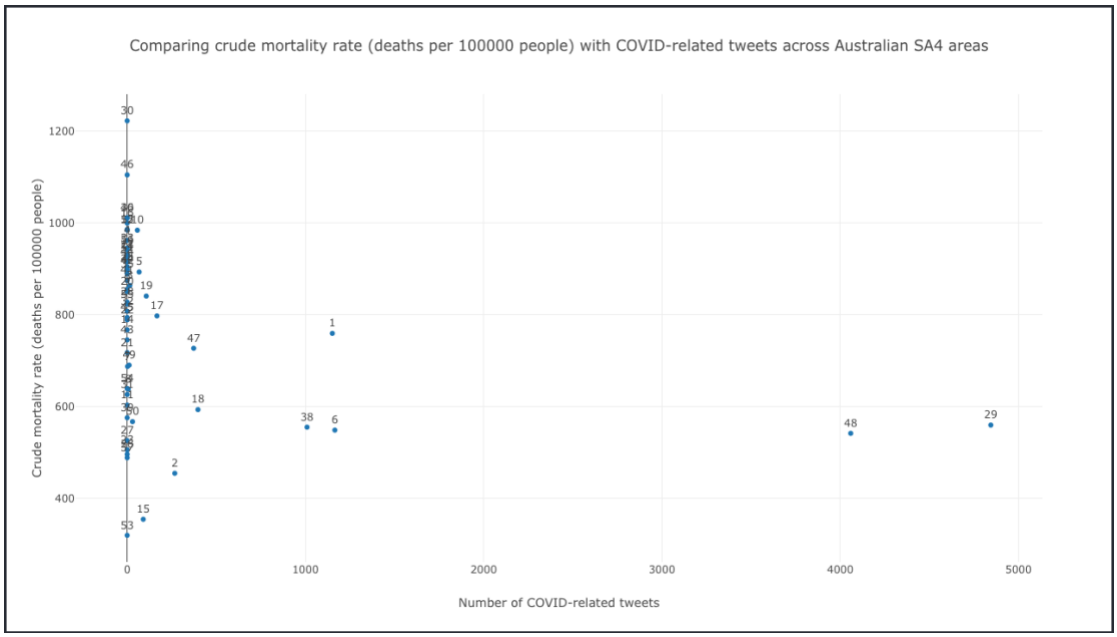


Figure 4: Crude mortality rate vs. Number of COVID-related tweets across SA4 areas

For this scenario, we compared the number of COVID-related tweets against two different mortality measures – ratio of area to average mortality (Figure 3), and crude mortality rate (Figure 4).

The two obvious outliers in this scenario are the areas Sydney (point 48) and Melbourne (point 29) (kindly refer to the 'Figures 3 & 4 legend' in the appendix for a mapping of SA4 areas to number ids). These two areas have the highest number of COVID-related tweets, but their mortality rates are among the lowest. This goes against our hypothesis where we assumed the opposite to be true. Apart from Sydney and Melbourne, Adelaide (point 1), Brisbane (point 6) and Perth (point 38) were also separated from the large cluster sitting close to the x=0 line.

This may signify that even though certain SA4 areas have a relatively large number of COVID-related tweets, it may not mean that those areas have a higher mortality rate. This could mean that the healthcare professionals in these areas may have been incredibly efficient in handling the spread of COVID-19, and taking drastic measures to minimize the number of deaths caused by the pandemic. Conversely, this may mean that the other areas with a higher mortality rate may not have performed as well.

We have to acknowledge that our harvested tweets were the raw number rather than a percentage of the all the tweets made in each area. In addition, the large cluster situated close to the x=0 lines in both figures above may be due to these areas having less dense population, not to mention that not everyone geotags their tweets. Lastly, there could be unknown data on other causes of mortality in these areas that display high mortality rates. For example, perhaps car accidents due to poorly maintained roads in Mid North Coast (point 30) could be the main reason why it has the highest mortality rates, instead of COVID-19.

5.1.3 Comparing number of COVID-related tweets and number of vehicle crashes in VIC and NSW from 2020 to 2021

5.1.3.1 Description

In this scenario we aim to analyze whether there is any correlation between the number of tweets in a state and the number of automobile crashes observed in the state. Our hypothesis which we aim to test is higher number of covid related tweets will correspond with lower number of crashes.

5.1.3.2 Why this scenario?

Different states in Australia had different responses to COVID-19, with states like Victoria going under a lockdown for months while other states remained relatively unscathed by the pandemic. We expect those states where a lockdown was implemented will have higher number of tweets on covid, as people confined to their homes will spend more time on social media, concurrently the number of automobile accidents should also be low since people would not be allowed to drive outside.

5.1.3.3 Results and analysis

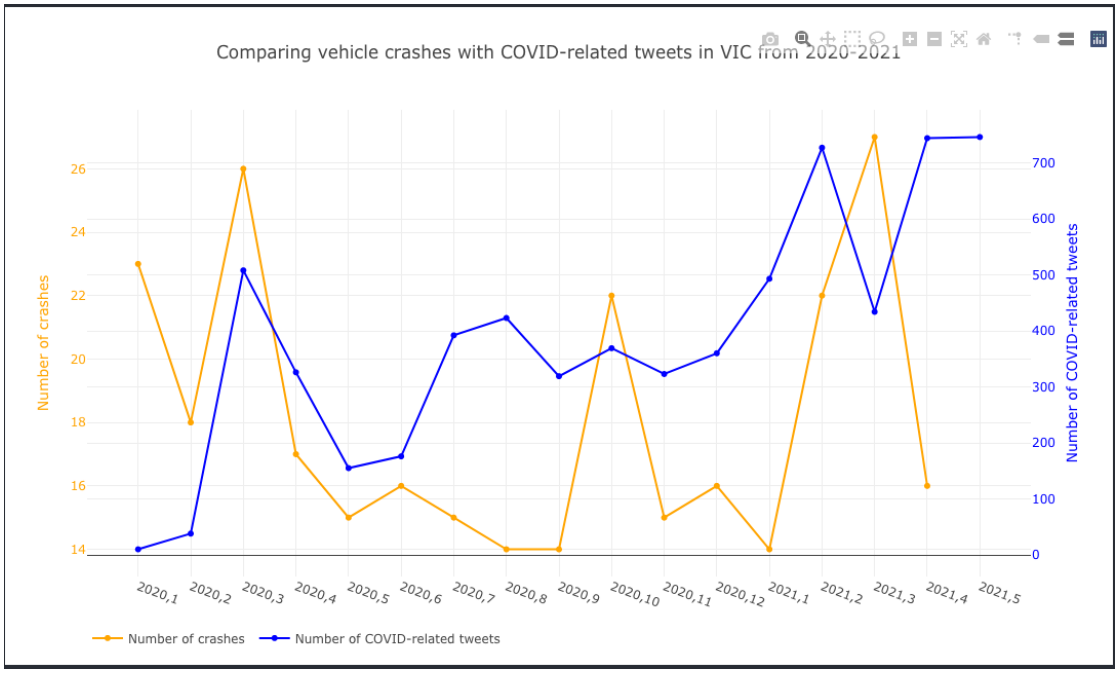


Figure 5: Comparing vehicle crashes with COVID-related tweets in VIC from 2020 to 2021

An assumption we make for this scenario is that the number of COVID-related tweets represent how much society is affected by COVID-19 (i.e. a higher number of tweets => potentially more cases => more lockdown restrictions). Following this, we assume the usage of vehicles to travel is lower during periods when there are more restrictions corresponding to less automobile

accidents. Thus, we hypothesize that the higher the number of COVID-related tweets, the lower the number of crashes.

Looking at the number of crashes (orange in above plot), we see three obvious spikes. They are in March 2020, October 2020 and March 2021. This roughly aligns, respectively, with the start of the pandemic-induced lockdowns, the first big loosening of lockdown restrictions, and when lockdown restrictions are further loosened to almost going back to 'normal life' (masks not required outdoors).

The number of COVID-related tweets seem to show a trend that does not align with our first assumption. Q2 to Q3 of 2020 were when there were most active cases of COVID-19, and the pandemic has 'slowed down ' since December 2020. However, we see that February, April and May 2021 were the months that had the most COVID-related tweets. Perhaps this is due to the introduction of COVID vaccines instead or people talking about COVID crisis in other countries.

We can conclude that for Victoria, there is no inverse relationship between the number of COVID-related tweets and number of crashes. In fact, if we look at the spikes of both number of COVID-related tweets and number of crashes around March 2020 and March 2021, we may even assume that these two show a more direct relationship instead.

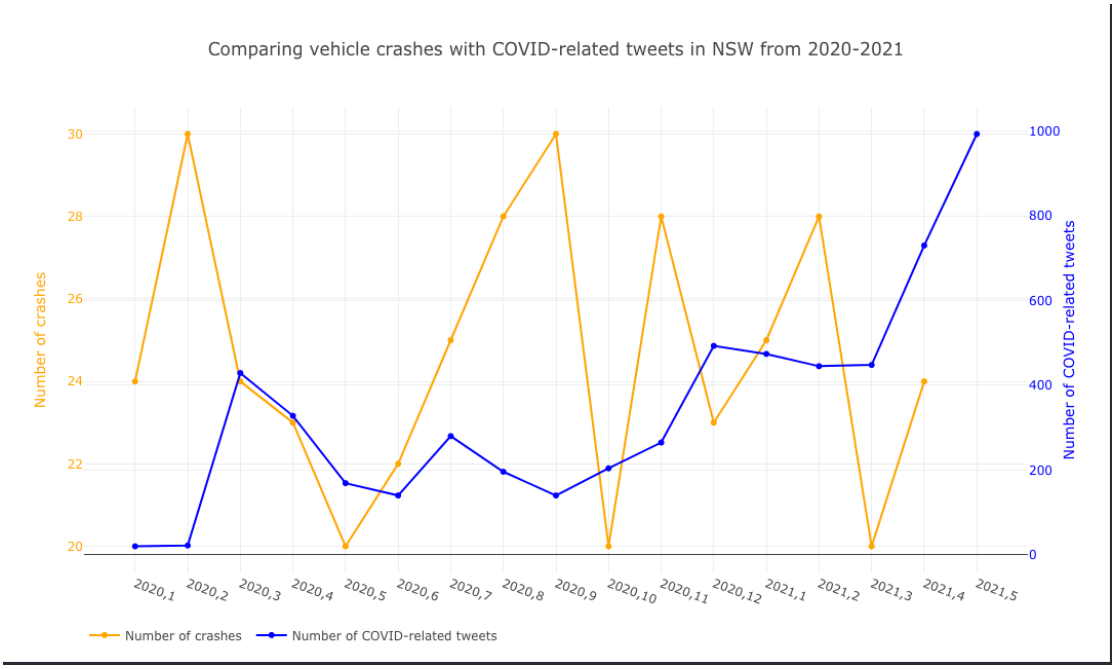


Figure 6: Comparing vehicle crashes with COVID-related tweets in NSW from 2020 to 2021

New South Wales seems to exhibit more drastic shifts from month to month in terms of car crashes. We see spikes in February, September and November of 2020, as well as February 2021. This may be due to the state implementing changes in lockdown restrictions more frequently than Victoria did.

The number of COVID-related tweets show a more steady upward trend instead. Once again, we can interpret this as our initial assumption of tweets representing the negative effect of COVID-19 on society to be wrong, or at least incomplete. With the trend shown by New South Wales, we may make an educated guess that the small spikes in March 2020 and December 2020 align with the start of the pandemic and Australia's overall successful progress with minimizing active cases respectively. On the other hand, the large spike in May 2021 may be due to vaccines being rolled out.

Similar to Victoria, there seem to be no inverse relationship between the number of COVID-related tweets and number of crashes in New South Wales.

6. Issues and Challenges

6.1 Where team worked well and where issues arose

Everyone worked rather well with each other and took on individual tasks responsibly to complete the project. The team members were able to complete assigned tasks within specified times. The communication between team members has been very smooth and was not affected by our difference in nationality, beliefs or language. Also, everyone was proactive in providing help whenever someone faced a specific technical issue.

That being said, there were some team issues that we had to address throughout the project. A major one was that the implementation part of our project later than planned, since everyone was preoccupied with other assessments at the start. Although initial planning meetings were held, there was no actual progress in producing workable code. Since most member had not worked with each other prior to this project, it took some time in understanding each other’s personalities and work behaviors. In addition, none of us have ever the key technologies needed for the project before (e.g. Docker, Ansible, CouchDB). However, we were quick to adapt. Even though writing Ansible scripts was tricky to get used to, we helped each other out in understanding the underlying concepts and the YAML syntax. Whenever faced with bugs or issues with deployment, we never hesitated to jump on video calls to solve those issues together.

We also worked quickly to the best of our abilities to catch up to speed even though implementation started late. We also made the best use of our time by choosing slightly simpler scenarios to analyze and present.

6.2 Assumptions on the effect of COVID-19 in Australia

All the team members of group 49 are international students; therefore, our assumptions were influenced by the situation regarding COVID-19 in our individual home countries. This might have led to us overestimate the negative impact of COVID-19 in Australia, leading to some of the

hypotheses for our scenarios not aligning with the actual results. Since the Australian government was relatively more successful in handling the pandemic, people seemed to be less concerned about health issues (i.e. scenario 2 on mortality rate), but rather social issues like lockdowns and vaccination rollouts (i.e. scenario 1 and 3).

6.3 Error handling in tweet harvester

The errors that could occur (e.g. connection error 501 when connecting to Twitter APIs) were anticipated and relevant try and except were used to address these errors. The initial database initialization was also put in a try and except block to counter the issue of already having a database with the same name. The protocol error was addressed explicitly. The protocol error was arising when the stream API of Twitter was being used and this was handled by implementing a try and except block. Another issue of exceeding the rate limit when using the Twitter API was handled by setting the "wait_on_rate_limit" to true. A try and except was also used for addressing the other errors that may not be anticipated and hence making the harvester more fault-resistant.

6.3.1 Limitation on tweets mining

We created a Twitter developer account to collect context-related tweets. However, due to Twitter's control over the tweet harvester program, our program was limited and prevented from calling the APIs from time to time. Later, we discovered that this restriction is related to the consumer key pair and application key pair used by the program. To remove this restriction, we went to Twitter developer's management page and created a new application to obtain and use a new key pair on the harvester.

6.4 Data and error handling in the frontend

6.4.1 CORS

The first challenge faced when developing the frontend was a CORS (Cross-Origin Resource Sharing) issue. Since our CouchDB master node was on one of the MRC instances, but our frontend was developed locally and will be deployed on a different instance, making API calls to retrieve data impossible without allowing CORS on CouchDB.

We could not change the CORS setting on the Fauxton client since our CouchDB was deployed in a cluster, so we had to manually configure that setting in the CouchDB config file. However, it was difficult locating the file, so solving this problem was not straightforward. After some research, we found an existing solution that only required installing an npm package and

running that package as a command (https://github.com/pouchdb/add-cors-to-couchdb). Thus, we added that dependency and the relevant commands to our Ansible playbook, effectively solving the CORS issue. As a result, the frontend could freely make API requests to our deployed CouchDB cluster to retrieve data.

6.4.2 Dynamically constructed variables

To make RESTful API requests to our CouchDB cluster, the frontend requires knowledge of the host IP address and the different database names. The problem was that these values, especially the IP address, are dynamically created and provided to us upon instance creation using Ansible playbooks, so its value can only be known after the fact.

We utilized the Jinja 2 syntax to create .j2 files, used the template module for Ansible to dynamically create, store and export these variables in a JavaScript file in the frontend directory. With this, we can obtain the correct values for host IP address and database names whenever our instances are recreated.

6.4.3 Matching AURIN data location values to Twitter location values

The location data from the harvested tweets are not in a consistent format. Some may be in the format '<city>, <state>', some may only have '<city>', while some only have 'Australia'. Location data from the AURIN data sets, however, were of a consistent standard (e.g., SA4 values).

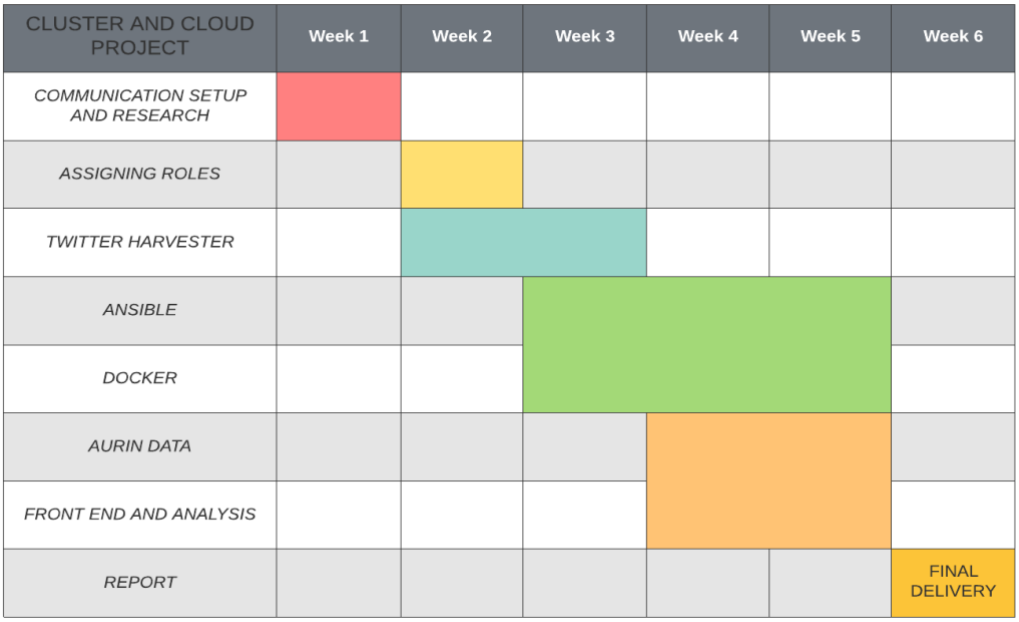
To overcome this, we performed some data manipulation on these location data in the frontend prior to visualization. We took the AURIN location data as our source of truth and used them to match the data from harvested Tweets. As a result, some tweets were discarded from analysis and visualization if they did not have matching location data (e.g., a tweet tagged in 'Australia' is not considered in our second scenario where SA4 areas were used).

7. Individual Contribution

Name	Contribution
Zhuolun Wu	Automated system deployment
YueYang Ho	Developing the frontend, designing MapReduce functions, visualizing and analyzing scenarios
Vignesh	Implementation of Twitter Harvester

Lakshminarayanan	
Jiashuai Yu	CouchDB, add MapReduce functions to playbook.
Agrim Binjola	Looking for scenarios, finding and uploading relevant data and basic twitter API implementation

TIMELINE:



The entire project lasted over a span of a six-week period, it began by establishing communication between group members and setting up the GitHub repository.
Each member was continuously assigned tasks to accomplish within specific timeframes. As a result, the team members were able to put together a great system and a thorough report.

8. Links

GitHub repository link: https://github.com/zachy-ho/cloud-assignment-2

Ansible deployment: https://youtu.be/t1M5dPROdqM

Frontend video: https://www.youtube.com/watch?v=qcu82MMD_WQ

9. Appendix

Twitter Developer App Tokens:

Access_token="1389053699012448256-i6DOTUDOKaGmWzJ1TpONQd1gQYyJdz"
Access_token_secret="lKGF6LO0niG3oUiPEhQVHA7qaPRav7b30mkFRnEfYJUjH"
Consumer_key="ZZlUEJQpRTv1i2QUyGGCMny1L"
Consumer_secret="AwgyemcsnEYGV2OI50ptsYNKQm0zgo8j96n7Sfyh2xtIFnWleh"

Key words used for harvesting COVID-19 related tweets:

“Antibodies, astrazeneca, clinical trial, community spread, confirmed positive case, contact tracing, contactless, coronavirus, covid, covid19, covid-19, double mutant, fights covid, financialaid, flatten the curve, forehead thermometer, herd immunity, jobloss, lock down, lockdown, mutant, mypandemicsurvivalplan, novel coronavirus, outbreak, oxygen, oxygen concentrator, oxygen cylinder, pandemic, pfizer, physical distancing, ppe quarantine, quarintineandchill, remdesivir, screening, self-isolation, self-isolation, selfquarantine, self-quarantine, social distancing, socialdistancing, spike protein, sputnik, steroids, use santizer, vaccine, ventilator, virus, wage, wageloss, wear mask, wfh.”

Note: The following keywords were chosen based on trending words in google trends for harvesting coronavirus related tweets, these are the commonly used words for tweeting about COVID-19.

SA4:
Statistical Areas Level 4 (SA4) are geographical areas built from whole Statistical Areas Level 3 (SA3s). The SA4 regions are the largest sub-State regions in the Main Structure of the Australian Statistical Geography Standard (ASGS), and have been designed for the output of a variety of regional data, including data from the 2016 Census of Population and Housing. They are specifically designed for the output of ABS Labour Force Survey data and therefore have population limits imposed by the Labour Force Survey sample. These areas represent labour markets or groups of labour markets within each State and Territory.

Whole SA4s aggregate to Greater Capital City Statistical Areas (GCCSA) and State and Territory. There are 107 SA4 regions covering the whole of Australia without gaps or overlaps. These include 18 non-spatial SA4 special purpose codes comprising Migratory–Offshore–Shipping and No Usual Address codes for each State and Territory.

The Other Territories of Jervis Bay, Cocos (Keeling) Islands, Christmas Island and Norfolk Island are together represented by a single SA4 in the 2016 ASGS.

Figures 3 & 4 legend:

1. Adelaide
2. Australian Capital Territory
3. Ballarat
4. Barossa - Yorke - Mid North
5. Bendigo
6. Brisbane
7. Bunbury
8. Cairns
9. Capital Region
10. Central Coast
11. Central Queensland
12. Central West
13. Coffs Harbour – Grafton
14. Darling Downs – Maranoa
15. Darwin
16. Far West and Orana
17. Geelong
18. Gold Coast
19. Hobart
20. Hume
21. Hunter Valley exc Newcastle
22. Llawarra
23. Ipswich
24. Latrobe – Gippsland
25. Launceston and North East
26. Logan – Beaudesert
27. Mackay - Isaac – Whitsunday
28. Mandurah
29. Melbourne
30. Mid North Coast
31. Moreton Bay
32. Mornington Peninsula
33. Murray
34. New England and North West
35. Newcastle and Lake Macquarie
36. North West
37. Northern Territory – Outback
38. Perth
39. Queensland – Outback
40. Richmond – Tweed
41. Riverina
42. Shepparton
43. South Australia – Outback
44. South Australia - South East

45. South East
46. Southern Highlands and Shoalhaven
47. Sunshine Coast
48. Sydney
49. Toowoomba
50. Townsville
51. Warrnambool and South West
52. West and North West
53. Western Australia - Outback (North)
54. Western Australia - Outback (South)
55. Western Australia - Wheat Belt
56. Wide Bay