Exploring Patterns: A Comparative Analysis of SUDO Data, Twitter Corpus, and Mastodon Data

Zhiyuan Chen[1], Natakorn Kam[2], Janya Kavit Pandya[3], Nandakishor Sarath[4], Atefeh Zamani[5]

[1] Melbourne, **Email**: zhichen1@student.unimelb.edu.au, **student ID**: 1080704.
[2] Melbourne, **Email**: natakorn.kam@student.unimelb.edu.au, **student ID**: 1244661.
[3] Melbourne, **Email**: janyakavit.pandya@student.unimelb.edu.au, **student ID**: 1291944.
[4] Melbourne, **Email**: nandakishor.sarath@student.unimelb.edu.au, **student ID**: 1470232.
[5] Sydney, **Email**: atefeh.zamani@student.unimelb.edu.au, **student ID**: 1129712.

1 System Design and Architecture

System Design and Architecture are essential parts of software development. They involve understanding how different components work together. In this section, we will discuss system architecture design and its components.

1.1 System Architecture Diagram

The diagram of the team's system is shown on Figure 1. The system is deployed on the University of Melbourne Cloud Network called MRC (Melbourne Research Cloud). The instances are deployed using ansible which is going to be discussed later. To access the cloud resources, we have to use VPN to connect to the university network in order to access MRC; therefore, we have to use a private key that was created in the computer section of the MRC dashboard. There are 4 instances on the cloud, which have different applications running on them, instance 1 runs CouchDB (Master Node) for CouchDB cluster and Mastodon crawler 1, instance 2 runs Mastodon crawler 2 and python mpi files (it contains huge Twitter file in /data directory), instance 3 runs flask (backend), and instance 4 is used to run reactJS (frontend). Furthermore, instances 2, 3, and 4 are part of the CouchDB cluster initiated in instance 1.

SUDO data is processed locally, and the resulting data is uploaded to CouchDB for storage, where it can be accessed by the backend when needed. CouchDB, mastodon crawlers, frontend and backend are all deployed using Docker. After the application is deployed using Ansible, the website hosted on instance 4 can be accessed to view the visualisations of the analysis.

1.2 Melbourne Research Cloud (MRC)

The team was allowed to use MRC to deploy the system. MRC offers researchers free access to computing resources that can be requested as needed. They provide features and capabilities that are comparable to those provided by commercial cloud providers like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platforms, [4].

1.2.1 Overview

The team has the flexibility to create up to 8 instances, each with 8 VCPUs and unlimited RAM. The cap volume for storage is set at 500 GB, which can be allocated as required, and can support a maximum

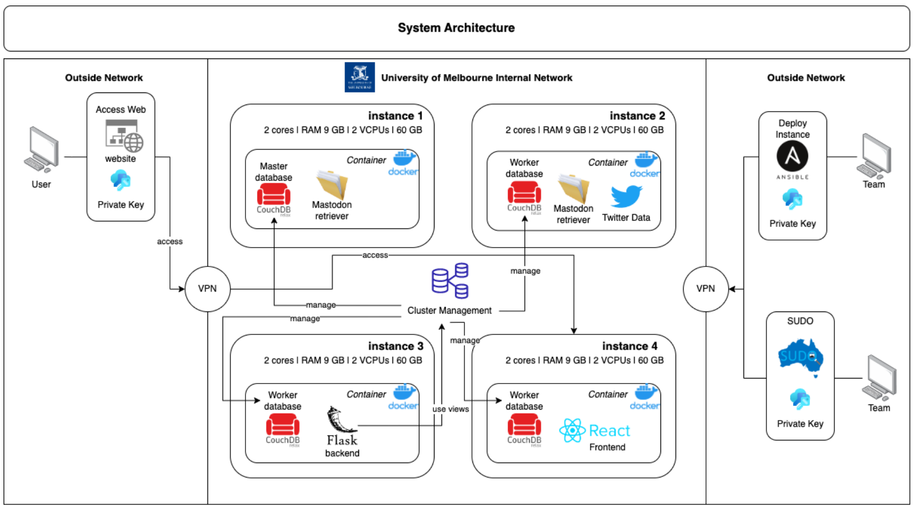
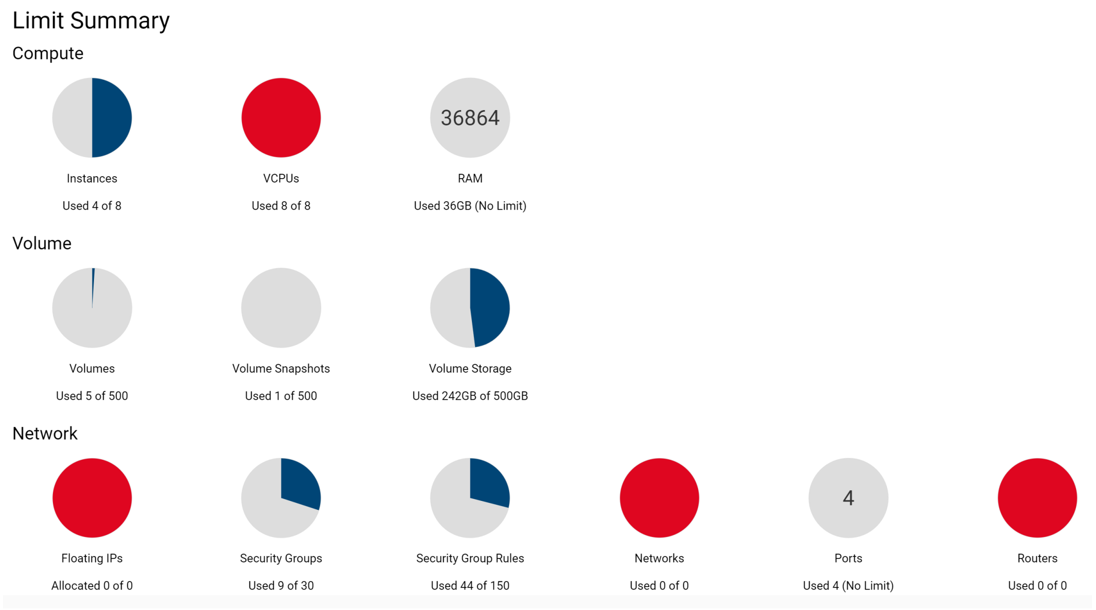


Fig. 1: System Architecture Diagram

of 500 snapshots. For the network, we can have 30 different settings for security groups, with a maximum of 150 security group rules and unlimited ports.

The team currently utilises 4 instances, each with 2 of VCPUs and 9 GB of RAM, and the total volume consumed is approximately 250 GB. In order to ensure secure access, there are 9 security groups established with a total of 46 rules, and 4 additional ports have been opened as required. Each instance has the same configuration using `uom.mse.2c9g` flavour, `melbourne-qh2-uom` network, and `NeCTAR Ubuntu 22.04 LTS (Jammy) amd64` image.


Limit Summary

Compute

Instances
Used 4 of 8

VCPUs
Used 8 of 8

36864
RAM
Used 36GB (No Limit)

Volume

Volumes
Used 5 of 500

Volume Snapshots
Used 1 of 500

Volume Storage
Used 242GB of 500GB

Network

Floating IPs
Allocated 0 of 0

Security Groups
Used 9 of 30

Security Group Rules
Used 44 of 150

Networks
Used 0 of 0

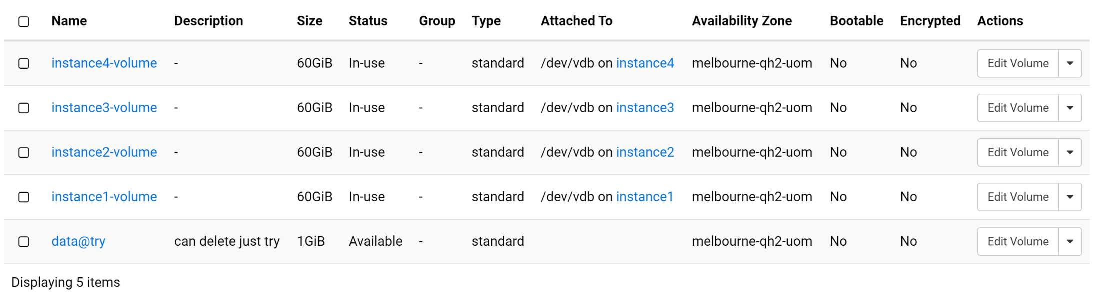
4
Ports
Used 4 (No Limit)

Routers
Used 0 of 0

Fig. 2: Overview of Instances Resources

1.2.2 Volumes

The team has created 4 instances with 60 GB each on the same network as the instance which is `melbourne-qh2-uom`, Figure 3. Each volume is attached to its corresponding cloud instances.



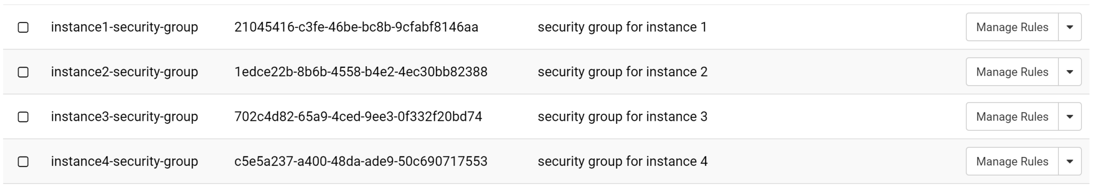
	Name	Description	Size	Status	Group	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
☐	instance4-volume	-	60GiB	In-use	-	standard	/dev/vdb on instance4	melbourne-qh2-uom	No	No	Edit Volume
☐	instance3-volume	-	60GiB	In-use	-	standard	/dev/vdb on instance3	melbourne-qh2-uom	No	No	Edit Volume
☐	instance2-volume	-	60GiB	In-use	-	standard	/dev/vdb on instance2	melbourne-qh2-uom	No	No	Edit Volume
☐	instance1-volume	-	60GiB	In-use	-	standard	/dev/vdb on instance1	melbourne-qh2-uom	No	No	Edit Volume
☐	data@try	can delete just try	1GiB	Available	-	standard		melbourne-qh2-uom	No	No	Edit Volume

Displaying 5 items

Fig. 3: Volumes of Each Instance

1.2.3 Network Security Group

To access cloud instances, the team needs to use VPN to access the cloud. This is because the cloud resource is hosted on a private network that is not directly accessible from the internet. By using VPN, a secure and encrypted connection can be established between the local machine and the university's internal network. This allows us to securely access the cloud resource and deploy our instance without exposing any sensitive information to potential security risks.



☐	instance1-security-group	21045416-c3fe-46be-bc8b-9cfabf8146aa	security group for instance 1	Manage Rules
☐	instance2-security-group	1edce22b-8b6b-4558-b4e2-4ec30bb82388	security group for instance 2	Manage Rules
☐	instance3-security-group	702c4d82-65a9-4ced-9ee3-0f332f20bd74	security group for instance 3	Manage Rules
☐	instance4-security-group	c5e5a237-a400-48da-ade9-50c690717553	security group for instance 4	Manage Rules

Fig. 4: Security Group Settings

The team has set an additional 4 security groups to control each instance, Figure 4. Instances 3 and 4 have special ports opening as instance 3 has to serve backend and instance 4 has to serve frontend, while all have to open the couchDB port. Opening all the ports is not safe because it can leave the system vulnerable to potential security threats; therefore, each port needs to be considered before opening. The team has configured the following ports for specific use cases in Table 1.

Port	Detail
22	For ssh connection
80	For HTTP connection
3000	For frontend connection
5000	For backend connection
5984	For couchDB connection

Table 1: Port Open

1.2.4 Advantages

In this section, we will explore the advantages of using the Melbourne Research Cloud (MRC) in our project. Having utilised the MRC firsthand, our team has experienced the benefits it offers. We will explore each key advantage that the MRC brings.

1. *Cost*: Using MRC in our project has a significant advantage when it comes to cost. As a university resource, there is no financial burden involved, allowing us to access the necessary resources without worrying about expenses. Unlike other cloud services that can lead to unexpectedly high bills, the MRC provides us with ample resources without any cost concerns. This enables us to focus on our project without the added stress of financial constraints.
2. *Security and Privacy*: Since MRC is a private cloud network, we're able configure our security option to only allow users to access the project using a university network, which would require them to

be physically there or use VPN to access it remotely. This ensures that only authorised individuals with proper access can connect to the MRC. On top of that, MRC also provides SSH(secure shell) which is commonly protocol when logging into a remote server. It ensures data that's transmitted are encrypted and only authorized entities with verified private keys are able to access the server.

3. *Learning Curve*: When using MRC as the cloud platform the team was able to grasp most of the knowledge that was needed for this project within very minimal times. MRC also provides tutorials from how to set up your first instance in cloud to load balancing which would be very helpful to researchers who just started cloud. Other cloud platforms does provide additional perks the MRC do not have, but as a trade off it may be overwhelming if you're just first started. From knowing hundreds of different services, thousands of documentations you will have to read to complex pricing structures that can cost you thousands of dollars if you're not careful it would most likely not be suitable for beginners especially if you're trying to process a large quantity of data like twitter data.

1.2.5 Disadvantage

While we have highlighted the advantages of MRC, it is important to acknowledge that there are also certain disadvantages. In the following section, we will also explore these disadvantages to provide a comprehensive understanding of both the benefits and potential drawbacks of using MRC for research projects.

1. *Reliability and Uptime*: Since MRC is a private cloud platform, the likelihood of your cloud instances going down is significantly higher than other public cloud platforms like AWS or Azure. Both of them have more than 99.999 percent up-time. In the case of large-scale data processing tasks like twitter analysis, any downtime can significantly delay the process of the project and potentially even cause data loss. According to forums in Ed discussion, one of the hosts for MRC has gone down days before this assignment's due date causing many team's cloud instances to go down with it. While most teams were able to retrieve their data back after contacting MRC's admin, some were unable to recover and lost their data entirely.
2. *Limited Services*: One of the key disadvantages with using MRC is the lack of service we were provided that would be beneficial for our project. While MRC does have services such as trove for NOSQL database and LBaas for load balancing as a service. None of these were provided for this project and all of the infrastructure for twitter analysis either has to be downloaded from the internet or made from scratch. In other platforms such as AWS we're able to access a list of extra services. For example, AWS Lambda allows you to run your code without provisioning which would be very helpful for data collection for Mastodon. For data storage, Amazon S3 provides industry leading scalability and performance, it can also store any types of data which is ideal for the large twitter data file.To perform data analysis, we could either query from S3 directly or use Amazon's NoSQL database DynamoDB to store and query data
3. *Lower Scalability*: While MRC cloud instances can scale dynamically based on what students or researchers demand. if you ask for too much resources, it's unlikely that the MRC's admin would give you what you ask for. In our project the twitter corpse is only $\tilde{7}$0GB which would be easily satisfiable. However, twitter data could easily scale to the TB or even PB scales of data and you might need dozens of to set up your infrastructure. In those instances, MRC might not meet those demands so switching to a public cloud platform would be a much more appropriate choice where you are given unlimited storage and services that can assist with large scale data analysis.

1.3 Ansible Component

Ansible is a popular open-source automation tool used to automate the provisioning process, configure settings, and infrastructure management. Using it helps process these complicated tasks better. It uses a language called YAML to describe the tasks, which is easy to understand as it is similar to normal English commands, [12]. We have used Ansible to automate these 4 important steps in the application.

1. *Instance creation and configuration*: as mentioned in previous sections, we can currently create a maximum of 4 instances on the cloud, but in the future, this resource allocation might change hence we have created `generate_application_hosts.py` file that takes input as number of instances to be created and writes information including associated volume name and security group rules to instances.yaml file. The file created is then used to dynamically generate instances on the MRC.

2. *Installing dependencies and libraries*: it is highly time-consuming to ssh into each instance and install libraries and dependencies hence we have automated the process to install Python, pip, CouchDB, docker, and many other dependencies. This can be done by running the command `./configure_instances.sh` command in the ansible directory.
3. *Setting up couchDB cluster and adding nodes to the cluster*: we have used ansible to set up couchDB cluster in a docker container and add all instances to the cluster.
4. *Deploying application*: we have used ansible to deploy dockerized mastodon crawlers on instance 1 and instance 2, and we have deployed dockerized flask server and react_ui front end on instance 3 and instance 4, respectively. Furthermore, we have used github ssh public-private key feature to automate the cloning of GitHub repositories on all instances.

1.4 Containerisation

The fundamental concept behind containerization is to provide a portable, isolated and consistent runtime environment for applications. For example, in our group, 2 members use Windows and, other 3 use MacOS. Suppose someone with a Windows OS is building the application, and the team member with MacOS is trying to deploy the application, a situation might arise where the dependencies for both OS are different hence leading to inconsistency. This and many other issues discussed in the next paragraph are tackled by docker by dealing with inconsistency between production and deployment environments. Some of the benefits of using containerization are as follows:

1. *Portability*: containers can be easily moved between various run time environments or virtual machines. Besides, it can be easily moved to other instances or other platforms.
2. *Isolation*: this property ensures that the dependencies and installations are encapsulated in the container itself. This helps to tackle library version problems as it installs specific versions of libraries.
3. *Scalability*: containers can be easily replicated on multiple servers/instances, and they can be managed on on-demand bases.

1.4.1 Docker

Docker is an open-source containerization technology that can automate the deployment of applications in a lightweight container, Therefore, it can easily operate efficiently in various computing environments without compatibility issues. Traditionally to manage the compatibility issue, a virtual machine was expected to be the solution but virtual machines occupy a lot of memory space and running multiple instances leads to unstable performance. This was not an efficient solution therefore Docker was developed. A virtual machine creates a virtualized environment that runs an operating system, which in turn runs the application. However, docker allows the user to focus on just running the application without the operating system acting as a middleman.

1.4.2 Building Docker Image and Running the Docker Container

Docker File can be used to automate the process of creating a Docker image and ensures consistency while reproducing the image.

– Syntax for building a Docker Image from Docker File:

```
docker build [OPTIONS] PATH
```

where `OPTIONS` include optional arguments and `PATH` specifies the path to the Docker File.

– Syntax for running a Docker container:

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

where `OPTIONS` include optional arguments, `IMAGE` is the name and tag of the Docker Image, `COMMAND` contains the commands to run inside the container (if no commands are set the default command from Docker File will be used) and `ARG...` specifies the arguments to be used to run the command.

1.4.3 Justification for Selection

Docker is lightweight and can perform all the necessary functions of a virtual machine. It can share and reuse large data volumes between several containers. Besides, it starts up quickly, is highly effective and is easily transferable without any compatibility issues. The reason for selecting can be seen below.

1. *Portability*: Docker containers are very portable. As docker is supported by Windows, Linux and MacOS, it is easy to develop and deploy an application across different environments without compatibility issues arising.
2. *Efficiency*: Docker containers are lightweight and can be quickly deployed, which reduces the time and effort required to develop an application significantly. As the containers do not require a guest operating system, it reduces the bootup time, and as all the containers are hosted under a single engine, better performance is observed.
3. *Consistency*: Docker containers provide a consistent environment for the program to run, which ensures that the outcome will not differ across different operating environments, thus reducing errors and inconsistencies.
4. *Scalability*: Docker containers are lightweight and hence it is easy to add or remove resources thus allowing the application to scale up or down to meet changes in demand.
5. *Security*: Docker containers isolate different parts of the code to specific containers, thus making it secure from outside security breaches.

1.5 Frontend System

The team has decided to use ReactJS as frontend for many reasons which will be discussed below. This framework uses javascript, HTML, and CSS which is not hard to understand. The libraries used to help visualise results are kepler.gl and canvasJS.

1.5.1 Kepler.gl

The team was finding the visualisation to help visualise data similarly to what SUDO could do. We want to present data on the corresponding place in the map, which is why kepler.gl is suitable for this task. Kepler.gl is an open-source data visualisation tool developed by Uber, which provides a powerful and user-friendly interface to visualise data. It can work with many types of data eg. CSV, geoJSON, JSON, and API-endpoint. One of the key features of this library is that it can handle large amounts of data, and allow exploration of data in real-time with many customisation options including changing colors, and adjusting size and shape of visualisations. This is the reason why the team decided to use Kepler.gl.

1.5.2 CanvasJS

In addition to visualising data on a map, the team also requires additional libraries to display data in chart format. The team found CanvasJS which fits the requirements as it has a range of charts that can be easily implemented on ReactJS. CanvasJS is a good choice for creating charts in web applications due to its flexibility, ease of use, and powerful features.

1.5.3 Justification for Selection

As pointed out in [7], there are several reasons why we chose ReactJS for our front-end development, which are listed below:

1. *Widely-use*: ReactJS is a highly popular framework for front-end development, particularly in web applications. Its popularity ensures that solutions for most errors or problems can be found within the large and active community surrounding the framework.
2. *Supported Libraries*: Since the team has specific requirements to visualise data on a map, it can be a challenging task. However, kepler.gl provides the functionality that we require. Additionally, this library is available in ReactJS, which is a framework that the team is familiar with and has experience using. As a result, it was a natural choice for us to use ReactJS for our front-end development needs.
3. *Component-based Architecture*: Being component-based, it makes developing easier as components can be reused. These smaller components can be developed independently and assembled to create a complete user interface, resulting in more efficient and faster development.

1.6 Backend System

The backend system connects the database and front end, making it a good place to process or format data. This helps ensure that the data received by the front end is clean and structured, leading to a better user experience. The team has selected Flask as a framework for backend development as it uses Python, which is the language that the team is familiar with.

1.6.1 Flask

Flask is designed to be a small and lightweight Python web framework that provides useful tools and features. It offers flexibility and is more accessible for beginners as it requires a single file to run the system, [1].

1.6.2 Justification for Selection

To select an appropriate framework for the backend, the team has considered many aspects, which are listed below:

1. *SOAP vs RESTful API*: These two are different API styles. API is Application Programming Interface, which refers to how communication between services works using request and response [9]. RESTful API is based on HTTP protocol using JSON for data format. It is flexible and lightweight, which can be scaled to serve more complicated requirements if required. On the other hand, SOAP is a protocol which is more complex and focuses more on security than REST. Its messages are typically in XML format and sent using HTTP as well. The team chose RESTful over SOAP for many reasons. One of them is that RESTful is easier to use and requires less coding. Moreover, authentication and security are not the main focus of this project; therefore, RESTful is an ideal choice.
2. *Programming Languages*: Flask uses Python to develop, which is one of the languages that the team uses in common; therefore, making it an ideal choice for backend development.
3. *Ease of Use*: Flask is known for its simplicity and requires only a single file to deploy. The team has compared it with other popular frameworks like Node.js. It is a powerful backend framework that can manage many features at once. However, it may require more setup and has asynchronous features that can result in faster runtime, but may take some time to understand.

1.7 CouchDB

In order to store and organize all the data the team has prepared for analysis, we have to find a proper place to do that. This is where a database is needed as it is a place to store and retrieve data. CouchDB is a type of database. It is a database management system that uses a NoSQL scheme to handle data. It can store JSON format which allows flexible and scalable storing of data.

Being a NoSQL database makes it unable to use a powerful SQL language to query data. SQL (Structured Query Language) is a programming language that can process information in a database eg. store, update, remove, search, and retrieve information. It is powerful yet easy to learn as it is similar to common English keywords, [11]. SQL databases have a fixed data format, which is suitable for structured data. On the other hand, NoSQL is optimized for applications that require large data volumes, and flexible data models, which is ideal for this project as we have many data formats and large amounts of data to process.

1.7.1 Justification for Selection

It is clear that the team has to use a NoSQL database as the data we have is unstructured and in large amounts. However, we chose specifically CouchDB for many reasons.

1. *Format*: Data is stored in CouchDB in a JSON format, which is easy to handle and manipulate. The original data from twitter and Mastodon also has this form; therefore, making it a good choice.
2. *Clustering*: The database system is deployed as a cluster, which means that we have multiple instances of CouchDB running in a coordinated way to provide high availability, fault tolerance, and scalability for database needs.
3. *MapReduce*: CouchDB's querying mechanism, implemented through MapReduce functions, provides users with a powerful tool for performing complex data retrieval. Views, which are generated through

the MapReduce process, can be customized to fit a wide range of use cases. Users can define their own MapReduce functions to perform operations ranging from simple counts and sum to more complex custom functions. This flexibility allows users to analyse and process large amounts of data in a variety of ways, making CouchDB a versatile choice for handling different types of data and applications.

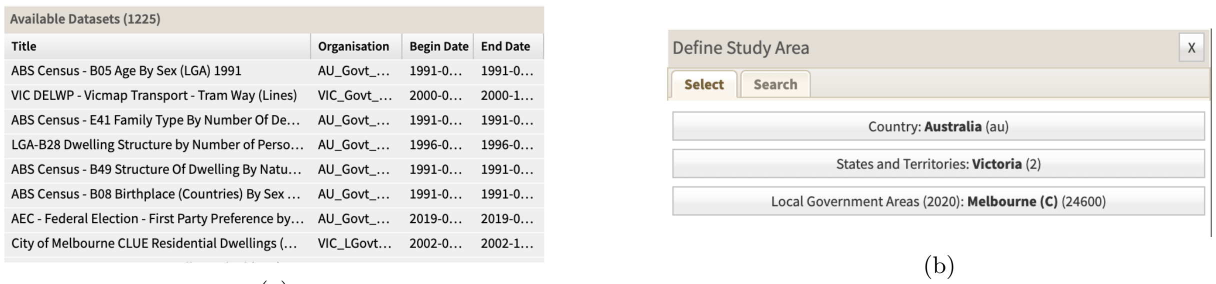
4. *HTTP request*: Querying data from CouchDB is easy as it can be done via basic `GET` with an appropriate parameter. We can query directly from the database table or to views.

2 Data Management

In this section, we will discuss how data is managed in this project, which involves multiple sources of information. We will explore how the data is stored and utilized within the system architecture.

2.1 SUDO Data

SUDO (Spatial Urban Data Observatory) is a platform that maintains many sources of datasets concerning many aspects in Australia such as crime, transportation, buildings, or employment rate, Figure 5(a). Data can be drilled down from the whole country to state and LGA (Local Government Area), which is really useful when trying to focus on finding information in a more refined area, Figure 5(b).



Available Datasets (1225)

Title	Organisation	Begin Date	End Date
ABS Census - B05 Age By Sex (LGA) 1991	AU_Govt_...	1991-0...	1991-0...
VIC DELWP - Vicmap Transport - Tram Way (Lines)	VIC_Govt_...	2000-0...	2000-1...
ABS Census - E41 Family Type By Number Of De...	AU_Govt_...	1991-0...	1991-0...
LGA-B28 Dwelling Structure by Number of Perso...	AU_Govt_...	1996-0...	1996-0...
ABS Census - B49 Structure Of Dwelling By Natu...	AU_Govt_...	1991-0...	1991-0...
ABS Census - B08 Birthplace (Countries) By Sex ...	AU_Govt_...	1991-0...	1991-0...
AEC - Federal Election - First Party Preference by...	AU_Govt_...	2019-0...	2019-0...
City of Melbourne CLUE Residential Dwellings (...	VIC_LGovt...	2002-0...	2002-1...

(a)

Define Study Area

Select | Search

Country: **Australia** (au)

States and Territories: **Victoria** (2)

Local Government Areas (2020): **Melbourne (C)** (24600)

(b)

Fig. 5: (a) Dataset, (b) Drill Down Area

The team uses information from SUDO to compare with the results gained from Twitter and Mastodon data and find the possible connections with SUDO data. The data obtained from SUDO may not be immediately suitable for use as it contains many columns that may not be relevant for certain use cases. This means that pre-processing is required before the data can be utilized. There are many ways to achieve this. SUDO provides a range of data manipulation tools to process the data, Figure 6. However, using these tools can be time-consuming and dependent on internet connectivity. Therefore, the team decided to download the dataset and process it locally using Python to avoid any delays or interruptions.

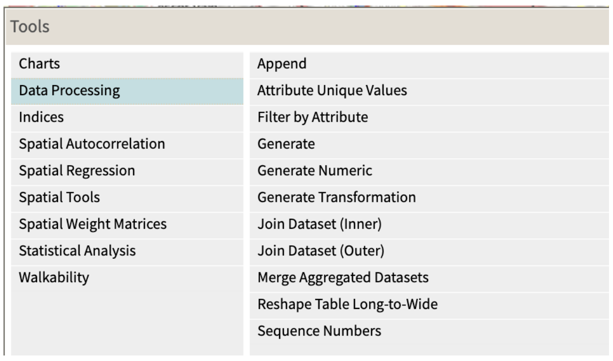


Fig. 6: SUDO tools

2.2 Dataset

The team has decided to narrow down the scope of the project to the Victoria state only, due to the limited and outdated data available on SUDO. We picked some datasets that we found interesting for each topic, which is listed below:

1. Rich and poor area
    a. SA2 Economy - National Regional Profile 2009 - 2013
2. Transportation
    a. PTV - Metro Tram Stops (Points)
    b. PTV - Metro Bus Stops (Points)
    c. PTV - Regional Train Stations (Points)
    d. PTV - Regional Bus Stops (Points)
    e. PTV - Regional Coach Stops (Points)
3. Crime / Domestic Violence
    a. VIC CSA - Family Violence - Domestic/Family/Sexual Violence Rate (LGA) June 2016-June 2018
    b. VIC CSA - Crime Statistics - Offences Recorded by Offense Type (LGA) 2008-2017
4. Park
    a. VIC DELWP - Public Land Management - Community Use Area (Polygons)
5. Population
    a. VIC DSDBI - Industry Atlas - Population by Gender (LGA) 2011

2.3 Implementing LGA (Local Government Area) GeoJSON

The team aims to display the results on the map, which requires the dataset to have latitude and longitude information. However, this information is not uniformly available across the dataset, with some data being present at the suburb level and some at the LGA level. Therefore, the team needs to process the data to ensure a consistent format.

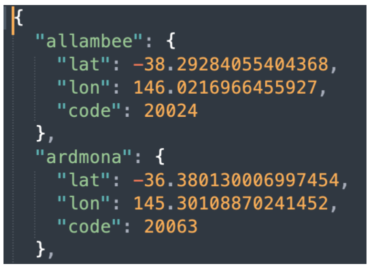
To make the data more visually understandable, the team opted to use LGA level and GeoJSON format. With 80 LGAs in Victoria, [3], it's much easier to comprehend data at the LGA level rather than trying to make sense of almost 3000 suburbs, [6]. This allows us to see different areas clearly on the map. We use GeoJSON of LGA and suburb data from OpenDataSoft website, which allows us to map according to code and name present in the SUDO dataset.

2.4 Twitter Data

We have applied an MPI-based tweet processing approach similar to assignment 1 to deal with the provided Twitter data file. The team had arranged the files that can be executed separately to populate the CouchDB database with Twitter data. These Python files can be run using the following command. `mpiexec -n <number_of_nodes> python file.py` .

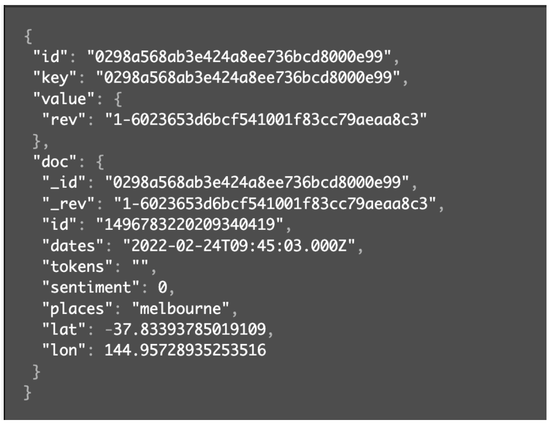
2.4.1 Implementing Suburb Coordinates for location data

The team has applied a similar approach to what was used in Assignment 1 to clean up location information in Twitter JSON data. However, more data is needed to meet the requirements of showing the data on a map. As a result, the team utilized GeoJSON information from the OpenDataSoft website to add latitude and longitude coordinates comparing by code or name of the suburb, Figure 7(a), and added it to the processed Twitter data. This is how we were able to obtain coordinates in our data in Figure 7(b). The command for populating couchDB with location data from Twitter is `mpiexec -n 2 python location.py`.



```
{
  "allambee": {
    "lat": -38.29284055404368,
    "lon": 146.0216966455927,
    "code": 20024
  },
  "ardmona": {
    "lat": -36.380130006997454,
    "lon": 145.30108870241452,
    "code": 20063
  },
```

(a)



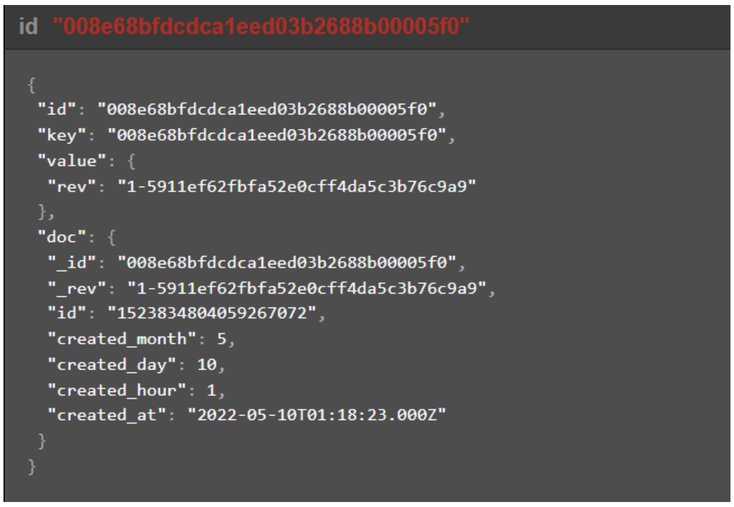
```
{
 "id": "0298a568ab3e424a8ee736bcd8000e99",
 "key": "0298a568ab3e424a8ee736bcd8000e99",
 "value": {
  "rev": "1-6023653d6bcf541001f83cc79aeaa8c3"
 },
 "doc": {
  "_id": "0298a568ab3e424a8ee736bcd8000e99",
  "_rev": "1-6023653d6bcf541001f83cc79aeaa8c3",
  "id": "1496783220209340419",
  "dates": "2022-02-24T09:45:03.000Z",
  "tokens": "",
  "sentiment": 0,
  "places": "melbourne",
  "lat": -37.83393785019109,
  "lon": 144.95728935253516
 }
}
```

(b)

Fig. 7: (a) Extract latitude and longitude, (b) Processed Twitter Data

2.4.2 Time Analysis Data

Since the dataset is huge and there are around 60 million tweets, we have capped the data as follows. It was identified that we have data from February 2022 till August 2022, hence for each month, we have taken 600,000 tweets and for those tweets, we have stored the created_at field with day, month, and hour variables. Figure 8 shows processed time data. The command for populating couchDB is `mpiexec -n 2 python time.py`.



```
id "008e68bfdcdca1eed03b2688b00005f0"

{
 "id": "008e68bfdcdca1eed03b2688b00005f0",
 "key": "008e68bfdcdca1eed03b2688b00005f0",
 "value": {
  "rev": "1-5911ef62fbfa52e0cff4da5c3b76c9a9"
 },
 "doc": {
  "_id": "008e68bfdcdca1eed03b2688b00005f0",
  "_rev": "1-5911ef62fbfa52e0cff4da5c3b76c9a9",
  "id": "1523834804059267072",
  "created_month": 5,
  "created_day": 10,
  "created_hour": 1,
  "created_at": "2022-05-10T01:18:23.000Z"
 }
}
```

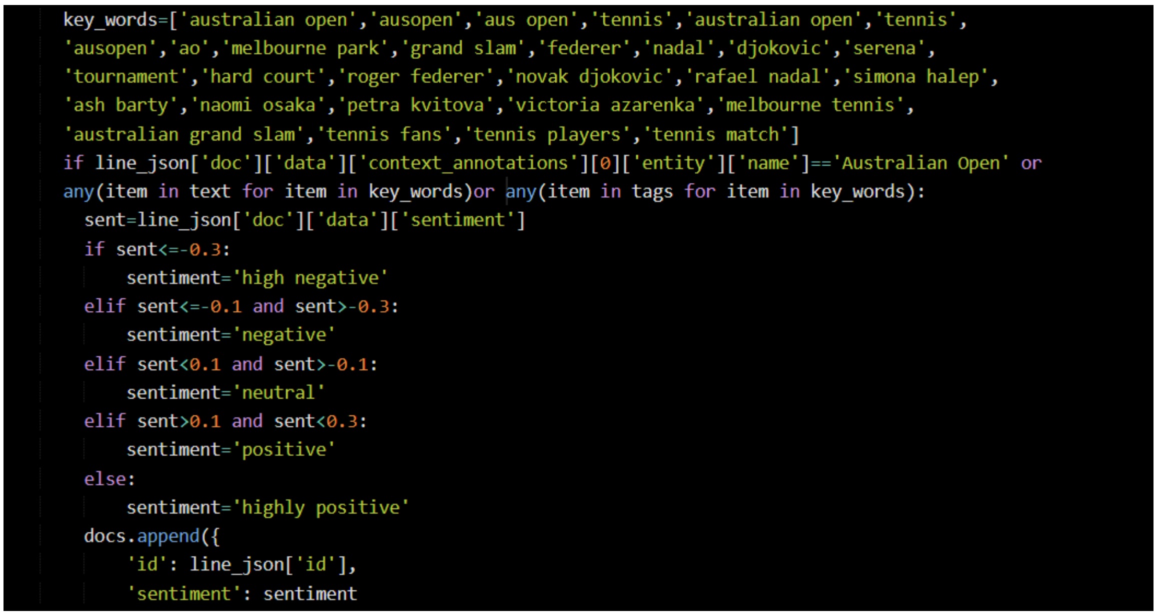
Fig. 8: Processed Time Data

2.4.3 Event Analysis Data

To populate event data, we have mapped popular keywords related to events with tags and texts associated with tweets. furthermore, we have converted the sentiment obtained from mapped tweets into 5 categories: high negative, negative, neutral, positive, and high positive. For example, Figure 9 illustrates different keywords and sentiment conditioning that are used for Open Australia. The complete list of words considered for each event can be found in Table 2. The command for populating couchDB is `mpiexec -n 2 python ao.py / lgbtq.py / melbourne_cup.py / vivid_sydney.py / gp.py`.

Event	Relevant keywords
LGBTQ	sydney gay and lesbian mardi gras, lgbtq, pridemonth, pride month, lgbtqiaplus, lgbt-pride, lgbtcommunity, sydneyworldpride, gay,lesbian, bisexual, transgender, loveislove, rainbow, sydneygay, gayaustralia, nonbinary, gbtqcommunity, queer, transrightsarehumanrights
Melbourne Cup	melbournecup, flemington, melbournecupcarnival, horseracing, springcarnival, emiratesmelbournecup, cupday, cupweek, theracethatstopsanation, melbournecup, cupday, flemington, springcarnival, horseracing, horse, racing, melbournecupcarnival, racing-fashion, punters, sweepstakes, racehorse, thoroughbred, jockey, trainer, bookmaker
Vivid Sydney	vivd sydney, vivid sydney, vivid, vividsydney, vividsydney2022, vividsydney2023, light festival, lightart, lightinstallation, artinstallation, music, performance, harbour, harbourbridge, opera, operahouse, circularquay, the rocks, sydfilmfest, sydney light festival
Australian Open	australian open, ausopen, aus open, tennis, australian open, tennis, ausopen, ao, melbourne park, grand slam, federer, nadal, djokovic, serena, tournament, hard court, roger federer, novak djokovic, rafael nadal, simona halep, ash barty, naomi osaka, petra kvitova, victoria azarenka, melbourne tennis, australian grand slam, tennis fans, tennis players, tennis match
Grand Prix	australiangp, ausgp, f1australia, melbournegp, albertpark, f1melbourne, f1downunder, f1oz, f1, car racing, ausgp2019, ausgp2020, ausgp2021, albertparkcircuit, melbournegrandprix, f1melbourne, f1downunder, f1australia, australiangp2021, australiangrandprix2021, australianformula1, australianformulaone, formulaoneaustralia, f1melbourne2021, f1australia2021, melbournef1, melbournegp2021, albertparkgp, albertparkgrandprix, agp2021

Table 2: Relevant Keywords on Events

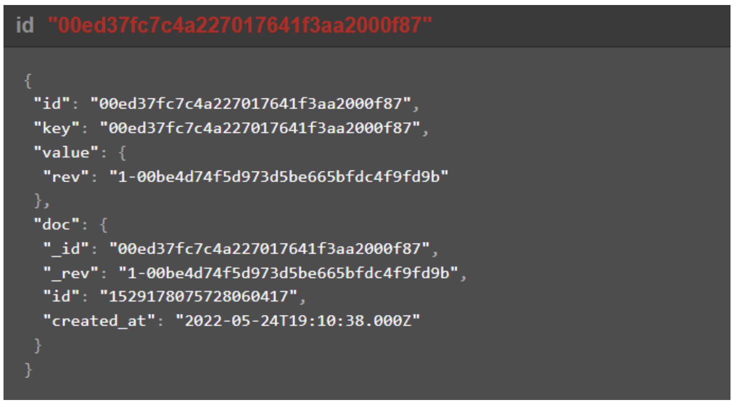

```
key_words=['australian open','ausopen','aus open','tennis','australian open','tennis',
'ausopen','ao','melbourne park','grand slam','federer','nadal','djokovic','serena',
'tournament','hard court','roger federer','novak djokovic','rafael nadal','simona halep',
'ash barty','naomi osaka','petra kvitova','victoria azarenka','melbourne tennis',
'australian grand slam','tennis fans','tennis players','tennis match']
if line_json['doc']['data']['context_annotations'][0]['entity']['name']=='Australian Open' or
any(item in text for item in key_words)or any(item in tags for item in key_words):
  sent=line_json['doc']['data']['sentiment']
  if sent<=-0.3:
      sentiment='high negative'
  elif sent<=-0.1 and sent>-0.3:
      sentiment='negative'
  elif sent<0.1 and sent>-0.1:
      sentiment='neutral'
  elif sent>0.1 and sent<0.3:
      sentiment='positive'
  else:
      sentiment='highly positive'
  docs.append({
      'id': line_json['id'],
      'sentiment': sentiment
```
Fig. 9: Keywords and conditioning for Australian open event

2.4.4 Crime Analysis Data

To populate crime data, we have used a similar approach to event and time analysis. We have used crime-related keywords to identify tweets and, for those tweets, we have extracted the created_at field along with day, month, and hour variables. Figure 10 shows processed tweet data. The command for populating couchDB is `mpiexec -n 2 python crime.py`.

2.5 Mastodon Data

As Mastodon toots lack geographical information and sentiment scores that twitter data possess, our analysis with Mastodon data would revolve around comparison of toot numbers and temporal distribution with twitter. More specifically, we've decided to focus crime related toot for toot number comparison and



```
id "00ed37fc7c4a227017641f3aa2000f87"

{
 "id": "00ed37fc7c4a227017641f3aa2000f87",
 "key": "00ed37fc7c4a227017641f3aa2000f87",
 "value": {
  "rev": "1-00be4d74f5d973d5be665bfdc4f9fd9b"
 },
 "doc": {
  "_id": "00ed37fc7c4a227017641f3aa2000f87",
  "_rev": "1-00be4d74f5d973d5be665bfdc4f9fd9b",
  "id": "1529178075728060417",
  "created_at": "2022-05-24T19:10:38.000Z"
 }
}
```

Fig. 10: Crime analysis processed tweet

all toots from mastodon.au(an Australian Mastodon Server) to compare temporal distribution with twitter data. To gather data from mastodon.au, we use Mastodon.py's API functions Mastodon.timeline_local which can iteratively gather all public timeline's toot. For crime related toots, Mastodon.timeline_hashtag was used to collect data and identifying crime-related toots using a list of relevant hashtags in table 3, assuming that any toots with these tags relate to crimes.

Scenarios	Relevant hashtags
Crime	abduction, blackmail, bribery, bombing, corruption, crime, cybercrime, domestic violence, drug, espionage, embezzlement, family violence, felony, forgery, fraud, genocide, gunviolence, kidnapping, hit and run, identify theft, kidnapping, manslaughter, missingperson, murder, rape, robbery, riot, terrorism, theft, trafficking, sexcrimes, scam, smuggling

Table 3: Relevant hashtags

2.6 CouchDB Data storage

In this section, we have discussed how map-reduce functions of CouchDB are used for generating valuable insights in the form of views after performing steps of data extraction and processing described in previous sections. Figure 11 illustrates the databases that we have created to fit particular scenarios. Besides, in the following sections, we are going to discuss how the CouchDB cluster is being used for load balancing.

2.6.1 MapReduce Function

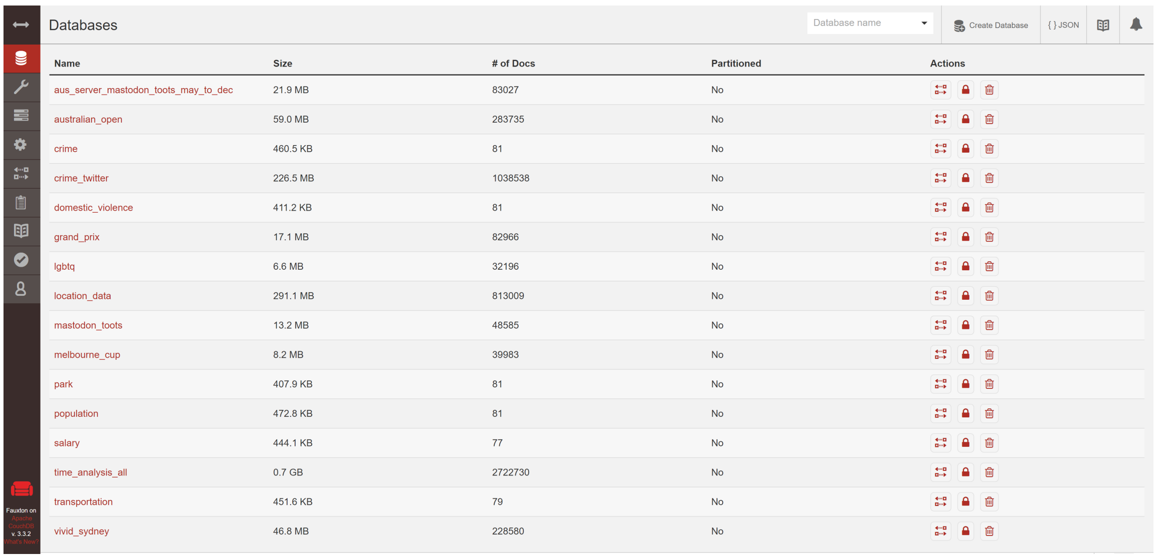
In order to query the database for needed information efficiently, we use the MapReduce function to call for results. For example, we are counting each level of sentiment for people talking about the Australian Open event. From Figure 12(a), we can see that we use only sentiment fields, and use function _sum to calculate the sum of each level. The result can be seen in Figure 12(b).

Another example can be seen as Twitter location data. In this scenario, we need to extract all the coordinates available in the database, which can be done using the function presented in Figure 13(a). The result can be seen in Figure 13(b). Using this function, we can also know how many tweets have been made in each LGA area.

These views created using mapreduce functions are accessed by the flask server backend to retrieve reduced data and generate visualizations.

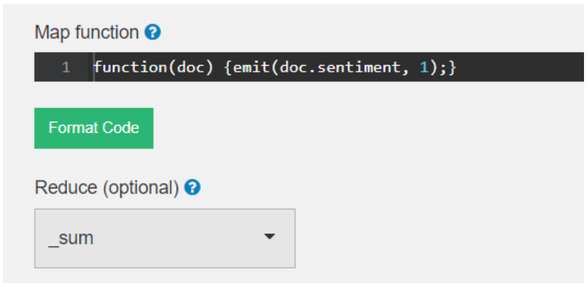
2.6.2 Load Balancing using CouchDB Cluster

We have deployed a CouchDB cluster that has instance 1 as the master node and all other nodes as workers. This essentially means that all the nodes can access the data and views. Furthermore, CouchDB ensures that the data is evenly distributed amongst all the nodes and the data is always consistent and the database is fault tolerant. We have utilized this functionality of CouchDB by evenly distributing the



Name	Size	# of Docs	Partitioned	Actions
aus_server_mastodon_toots_may_to_dec	21.9 MB	83027	No	
australian_open	59.0 MB	283735	No	
crime	460.5 KB	81	No	
crime_twitter	226.5 MB	1038538	No	
domestic_violence	411.2 KB	81	No	
grand_prix	17.1 MB	82966	No	
lgbtq	6.6 MB	32196	No	
location_data	291.1 MB	813009	No	
mastodon_toots	13.2 MB	48585	No	
melbourne_cup	8.2 MB	39983	No	
park	407.9 KB	81	No	
population	472.8 KB	81	No	
salary	444.1 KB	77	No	
time_analysis_all	0.7 GB	2722730	No	
transportation	451.6 KB	79	No	
vivid_sydney	46.8 MB	228580	No	

Fig. 11: CouchDB Database



```
function(doc) {emit(doc.sentiment, 1);}
```

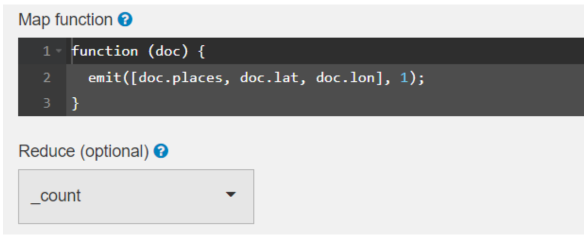
_sum

(a)

```
{"rows":[
{"key":"high negative","value":4197},
{"key":"highly positive","value":26812},
{"key":"negative","value":24635},
{"key":"neutral","value":170356},
{"key":"positive","value":57734}
]}
```

(b)

Fig. 12: (a) MapReduce Function for Australian Open, (b) Results of MapReduce Function



```
function (doc) {
  emit([doc.places, doc.lat, doc.lon], 1);
}
```

_count

(a)

```
{"rows":[
{"key":["aberfeldy",-37.699794103329765,146.3785204898876],"value":4},
{"key":["ada",-37.84223083873772,145.85183638224137],"value":3},
{"key":["airly",-38.03077362266902,147.09742184134436],"value":1},
{"key":["allambee",-38.29284055404368,146.0216966455927],"value":1},
{"key":["allansford",-38.393522206667065,142.61172317470766],"value":4},
{"key":["amherst",-37.13593670245508,143.65321231431145],"value":1},
{"key":["amphitheatre",-37.18793913813407,143.41480840705552],"value":1},
{"key":["anglesea",-38.38081202771149,144.1542871656297],"value":261},
{"key":["ararat",-37.26243116186075,142.97564593258485],"value":1582},
{"key":["arawata",-38.392792479568335,145.87806181301934],"value":1},
{"key":["arbuckle",-37.369038333947174,146.82916850709285],"value":2},
{"key":["archerton",-36.86887315293998,146.20905393432608],"value":1},
{"key":["ardmona",-36.380130006997454,145.30108870241452],"value":1},
{"key":["avenel",-36.92496793783064,145.28274656927258],"value":24},
```

(b)

Fig. 13: (a) MapReduce Function for Twitter Location Data, (b) Results of MapReduce Function

data fetches in the backend, so that the load of fetching the data is evenly distributed (balanced) amongst all nodes.

2.7 Communication Between Systems

After all the data is stored in the database and the MapReduce function is used for query data, the next step is to use the backend system to fetch data in the database, then use that data to display on frontend. These steps are demonstrated in Figure 14. It can be seen that backend acts as a medium between the database and frontend. We need a backend to handle data and implement additional logic. Additionally, we can implement security such as authentication, authorization, and validation for data.

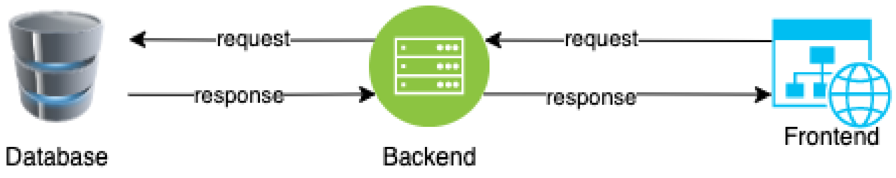


Fig. 14: Communication Between Systems

3 System Functionalities

In this section, we will explore the different features and functionalities offered by our system, and how they cater to the scenarios we have analysed. We will also showcase the frontend visualisations for each scenario, to give a better understanding of how the system works.

3.1 Dynamic Scaling

Our system provides dynamic scaling in the following ways:

1. Automatic Resource Allocation: the system uses a Python file that takes input as a number of instances to be created and in the same Python file volume size, security groups, and security group rules can be modified to scale the number of instances and change configuration based on resource requirement and changes. Furthermore, we have used Ansible to deploy and configure these instances.

2. CouchDB Cluster: the system uses the couchDb cluster for balancing the load on the flask backend. If the number of tweets increases or the number of scenarios increases the backend will automatically distribute the load amongst Couchdb cluster nodes and reduce output times on the front end. Moreover, on creation, new nodes can be added to the cluster by running the configinstances.sh file.

3. Mastodon Server: the system uses docker to deploy 2 mastodon crawlers on different instances, the number of crawlers can be changed on demand and easily deployed using docker, the system uses ansible to deploy these mastodon crawlers.

3.2 Functions

The team has achieved many requirements in the project, which is listed below:

1. Use ansible to automate deploy cloud instances process and install necessary dependencies
2. Use ansible to automate deploy of couchDB cluster, frontend, and backend
3. Process huge amounts of Twitter data for each scenarios using MPI across all instances
4. Automation of mastodon crawlers on instances
5. Use Docker containers for CouchDB, mastodon crawler, frontend, backend
6. Use MapReduce function in CouchDB for creating views
7. Use data from CouchDB through view to be processed at backend before sending it to frontend
8. Visualise processed data comparing it with SUDO data and display it on web application

3.3 Scenarios

As mentioned before, the team has decided to focus on Victoria state only, as SUDO offers limited information on the team's interest scenarios. We have chosen the following scenarios for analysis:

1. Sentiment analysis
2. Time-based comparison of crime-related tweets vs crime related toots
3. Analysis of tweeting habits of people,

In the following, we present a short description of each scenario along with an example on the visualisation provided on the frontend. For the complete set of visualisations, we refer reader to the webpage of this project.

3.3.1 Sentiment analysis

Sentiment analysis is a technique which applies natural language processing and machine learning methods to present the emotions and opinions that can be found in found within comments, feedback or critiques, [5]. Here, the sentiment analysis of the events is based on the tweets from Victoria which mentioned one of the five big events in Australia: LGBTQ, Melbourne Cup, Vivid Sydney, Australian Open, Grand Prix.

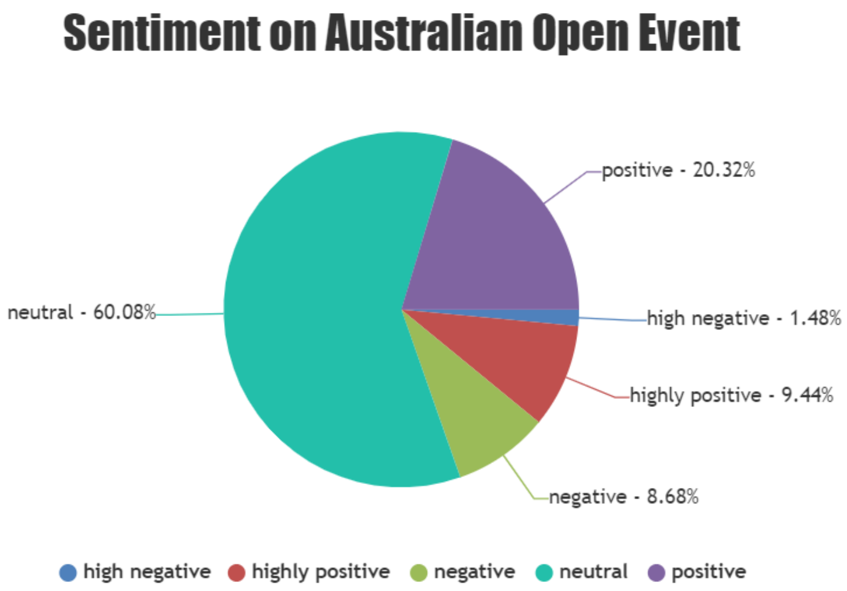


Fig. 15: Sentiment analysis of tweets mentioning one of the keywords related to Australian Open, mentioned in Table 2.

As an example, Figure 15 presents the result of sentiment analysis for the Australian open. As can be seen, more than 29% of the tweets about this event are positive/high positive while 60% are neutral. Therefore, in general, it can be concluded that people do not provide negative emotions and opinions about this event. In general, the percentage of neutral sentiments for the mentioned events ranges between 58.93% to 66.55%, and the percentage of positive/high positive sentiments is between 25.48% to 33.86%.

To provide a better understanding of sentiment analysis, we use the information extracted from SUDO. Just note that for the scenarios based on the SUDO data, we focussed on the tweets with location information, which is almost 6% of the tweets. The possible scenarios based on the SUDO data might be the average of tweets with positive sentiment analysis in different clusters in Victoria along with the data about crime, domestic violence, parks, population and salary in these areas. Figure 16 presents the sentiment analysis along with the crime data in different LGAs. In this figure, the circles provide information about sentiment analysis. Each circle provides the number of clusters in the area (the diameter of the circle) along with the average of positive tweets in that area (the greener the more positive tweets) while the crime data is displayed as the color of LGAs (the darker presents more crime data) in the background. Therefore, it seems that people being positive doesn't reflect from crime rate within the area.

Besides, the comparison between SUDO data and the sentiment analysis for each LGAs can be found in website as demonstrated in Figure 17 for Greater Bendigo.

3.3.2 Time-based comparison of crime using tweets toots

In this scenario and the next one, we will focus on time, since although most of the tweets and toots do not have location information, all of them have time. Therefore, focusing on the time might help us to gain some insight into the tweeting habits of people, who use social media as a tool to share their ideas.

Figure 18 demonstrates the monthly number of tweets and toots mentioning the keywords about crime. These keywords are presented in Table 3. As can be seen in this figure, the twitter data that we have access were from February 2022 to August 2022, while in this time period the we did not have

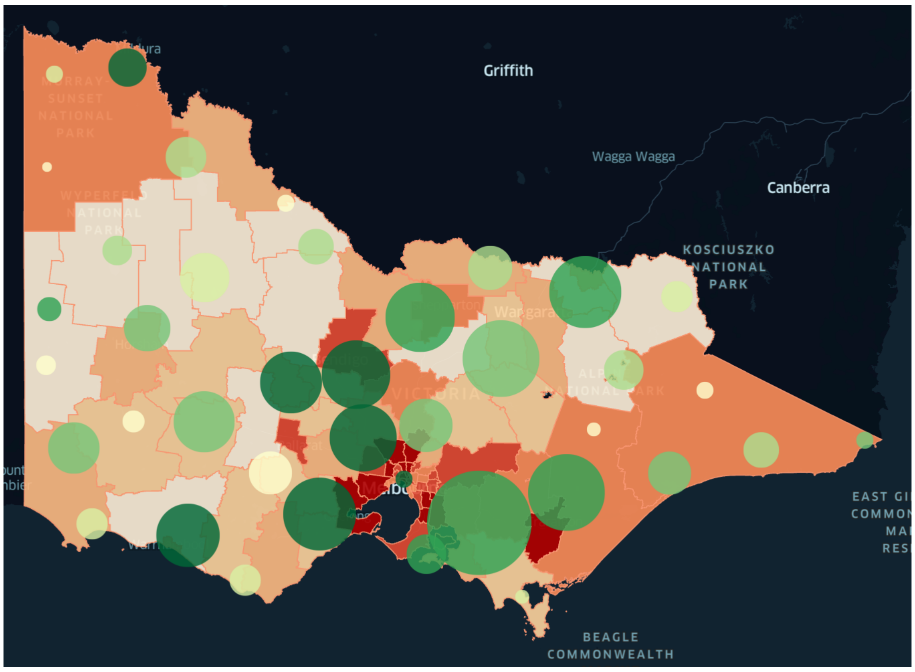


Fig. 16: Sentiment analysis of tweets using the SUDO data about crime in different LAG in Victoria.

Crime	DV %	Park	Population	Salary	Transportation
20534	49.2	402	101995	346786	722

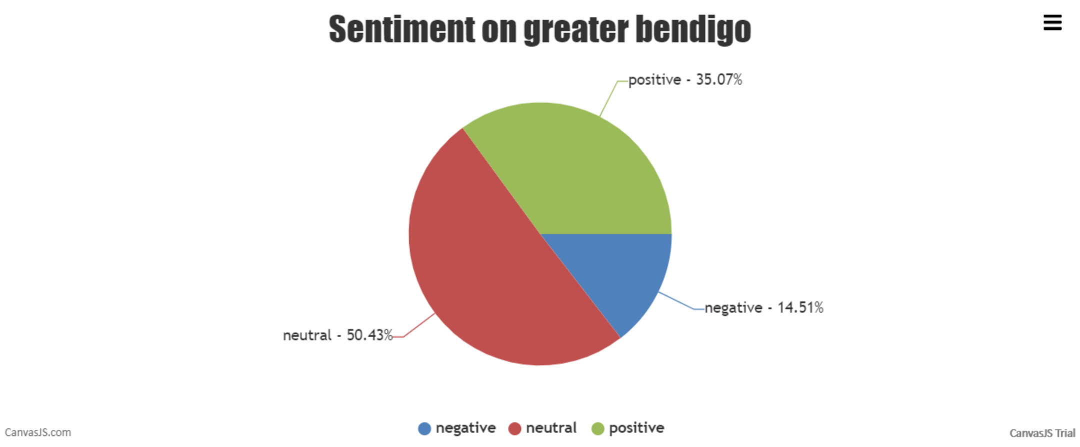


Fig. 17: Sentiment analysis of tweets in Greater Bendigo along with SUDO data for this suburb.

enough toots about crime. However, the number of toots increased drastically since October 2022, which was almost the same time as Elon Musk took control of Twitter [2].

To study the trend of posting tweets and toots about crime, we studied the posts mentioned related keywords based on day (Figure 19) and hour (Figure 20). As can be observed in Figure 19 the minimum number of posts concerning crime publishes during weekends. However, during working days, we see a different trend in the number of toots and tweets. The maximum number of tweets posted on Wednesdays with a decreasing trend towards Saturday and we can observe a large gap in the number of tweets on Mondays and Tuesdays with the rest of the days. However, we can see a more stable trend in the number of toots, especially since there is not a significant difference between the number of toots from Tuesday to Thursday.

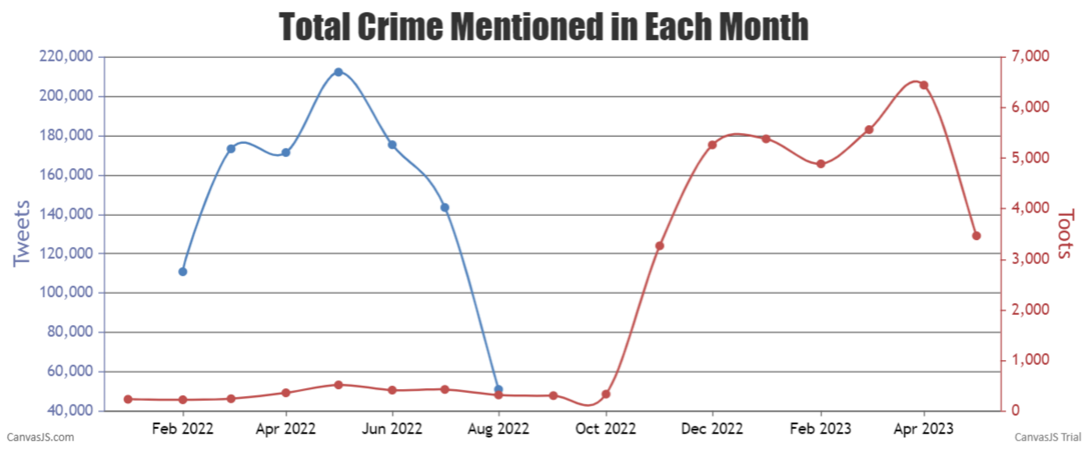


Fig. 18: Monthly number of tweets and toots mentioning keywords about crime.

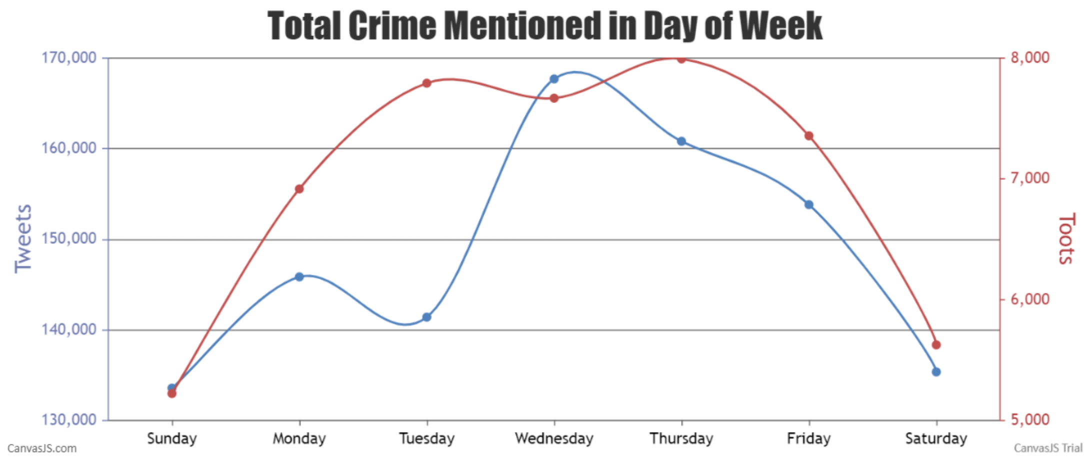


Fig. 19: The number of tweets and toots mentioning keywords about crime based on Day.

To take a closer look at crime-related posts, we examine the hour of their publication, Figure 20. It can clearly be observed that the number of toots and tweets follow different trends in this case, Although the number of tweets decreases gradually through working hours till 6 pm and then starts an increasing trend, the number of toots increases gradually till 2 pm to 3 pm and then starts the decreasing trend. It seems that people using Twitter as a social media are more active on this platform through the night while this is reversed for those who use Mastodon.

3.3.3 Analysis of tweeting habits of people

Analysing the number of tweets and toots based on day and hour might provide us some insight into when people might spend more time tweeting. Figure 21 provides the pie charts for the percentage of tweets and toots per day. As can be seen, For the tweets, Thursdays and Fridays have the highest numbers and Saturdays have the minimum number of tweets. For toots, the maximum value can be observed on Friday, while Tuesdays, Wednesdays, and Thursdays have the minimum number of tweets.

From another point of view, we can compare the number of tweets and toots based on hour of the day. As can be observed in Figure 22, from 7 am, the decreasing trend for the number of tweets and toots starts. By the end of the working hours, this trend gets reversed and we can see an increasing trend. The maximum number of tweets are posted around midnight and 5 am to 6 am, while for the toots, this happens from 8 pm to 10 pm and between 2 am to 3 am.

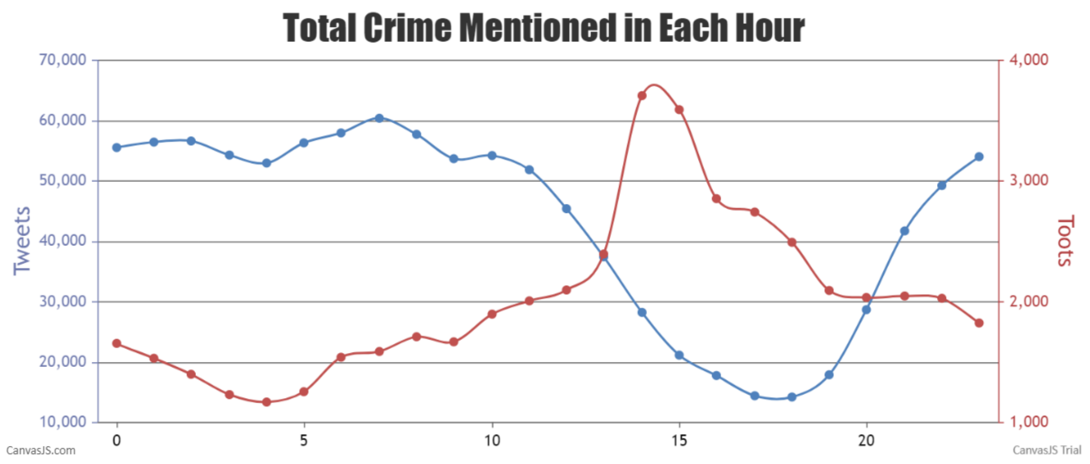


Fig. 20: The number of tweets and toots mentioning keywords about crime based on hour.

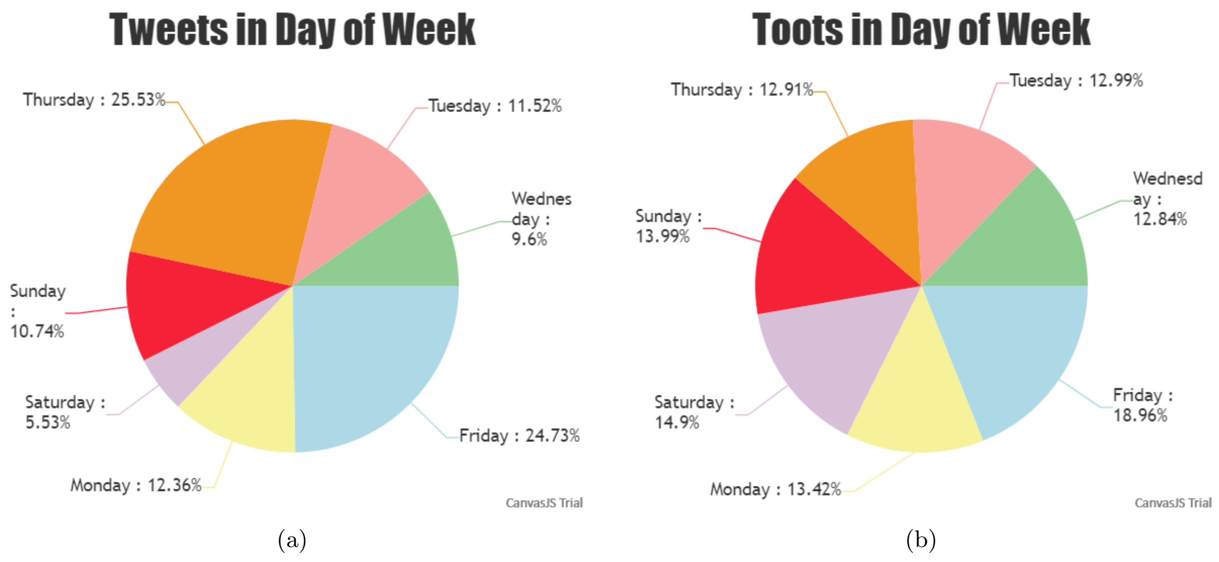


Fig. 21: (a) Tweets per day (percentage) (b) Toots per day (percentage)

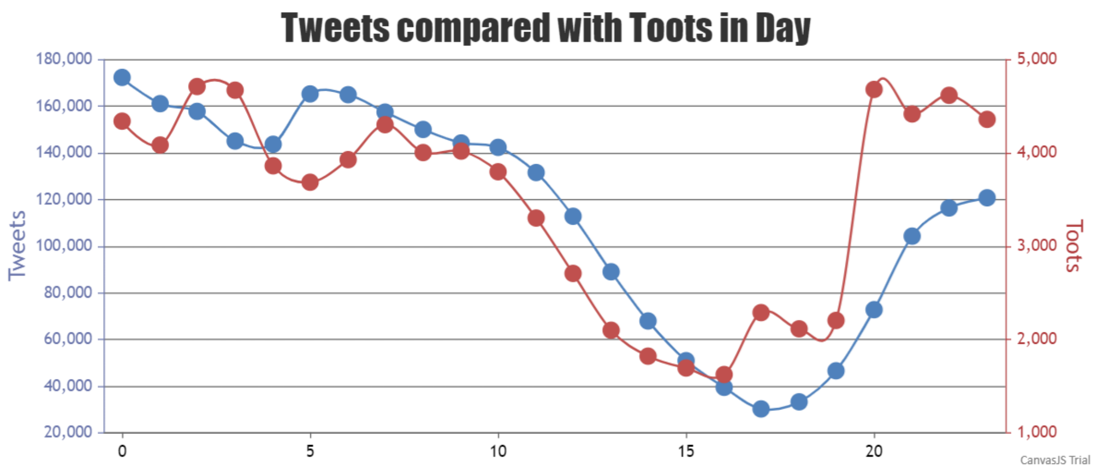


Fig. 22: Number of tweets and toots posted based on hour.

4 Issues and Challenges

In this section, we will discuss the issues and challenges that the team faced during the development of this project. These challenges can range from technical difficulties to data-related issues. By identifying these challenges, the team was able to address them.

4.1 Frontend Systems

Frontend is the first point of interaction between the end users and the system, and it plays a crucial role in providing a good user experience. However, developing a front comes with its own set of challenges and issues, ranging from understanding the framework to processing responses in requests. In this section, we will discuss some of the challenges and issues faced by the team during the development of the frontend for this project.

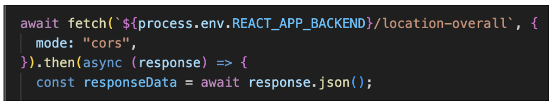
4.1.1 Unfamiliar with Framework

One of the teammates is quite familiar with developing frontend applications; however, it is not with ReactJS. Flutter is what one of my teammates is familiar with but it is rather difficult to apply it to our requirements. As mentioned before, Kepler.gl is such a powerful tool that the team couldn't ignore, which is why ReactJS is chosen as the frontend framework. Even though it is quite difficult to understand how this system works, it doesn't take long to learn and understand the basics in order to create a proper UI visualising data.

4.1.2 CORS Issues

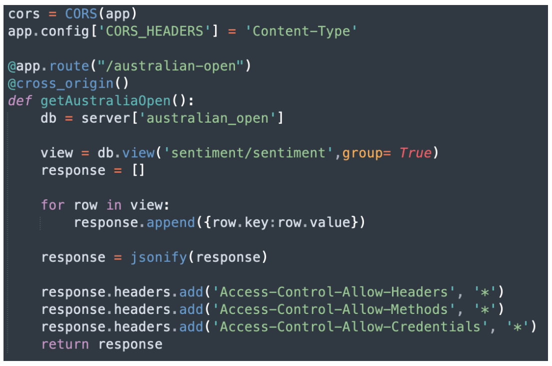
After starting to develop a static UI in react, the next step is to utilize data from calling from the backend. However, the team has encountered issues involving CORS. CORS (Cross-Origin Resource Sharing) is a security feature implemented by web browsers that restrict web pages from making requests to a different domain than the one that served the original web page, [10]. In this case, port 3000 is calling port 5000 which is in a different domain which is why this problem occurs.

In order to solve this problem, there are 2 places to be fixed. The first one is in the frontend where the request is called, we have to identify the mode we are using which is cors mode from Figure 23(a). The next one is in the flask or our backend. We need to install the CORS library in order to allow cross-origin in Figure 23(b).



```
await fetch(`${process.env.REACT_APP_BACKEND}/location-overall`, {
  mode: "cors",
}).then(async (response) => {
  const responseData = await response.json();
```

(a)



```
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'

@app.route("/australian-open")
@cross_origin()
def getAustraliaOpen():
    db = server['australian_open']

    view = db.view('sentiment/sentiment',group= True)
    response = []

    for row in view:
        response.append({row.key:row.value})

    response = jsonify(response)

    response.headers.add('Access-Control-Allow-Headers', '*')
    response.headers.add('Access-Control-Allow-Methods', '*')
    response.headers.add('Access-Control-Allow-Credentials', '*')
    return response
```

(b)

Fig. 23: (a) Frontend Fix, (b) Backend Fix

4.2 SUDO data

This section shall discuss the problem we found when processing SUDO data. Processing data from different sources is always a challenge, and the SUDO data used in this project was no exception.

4.2.1 Inconsistent Data

Information in each dataset may vary due to different sources and formats. Moreover, we found that some columns in some datasets had missing values or were entirely unnecessary for our project's scope, which required us to perform data cleaning and preprocessing to remove irrelevant data. This inconsistency was expected, given that the data was collected from different sources and had to be formatted to form a new dataset to use in the project. It presented a significant challenge in terms of data processing.

4.2.2 Missing Proper Location Data

This problem is mentioned before as data has inconsistent information between LGA and suburb; therefore, we try to compare it with the data we found online to form the proper GeoJSON information. We try to clean places columns as needed and compare them by code where available.

4.3 Ansible

Ansible is a virtualization tool used to automate the deployment of applications, this seems pretty straightforward as we just have to automate what we are doing manually but while implementing we came across many issues and challenges including, including ssh problems on Windows, problems with private keys and access to the created instance, setting up permissions for directories in an instance and many other small issues that are time-consuming as almost everyone is unfamiliar with the syntax and functionality.

4.3.1 SSH Problems

This was one of the most recurring problems we encountered in the initial stages of the project as we couldn't ssh into the server later on we figured out that windows wsl cannot connect to unimelb vpn directly hence we were unable to ssh into the server. Furthermore, when trying to configure the instances we encountered ssh issues again as the playbook couldn't ssh into the server the reason being the key permissions were not set correctly. We had to convert the permissions of the key from read, write execute to Only Admin read by applying chmod 0600 command on the file, this command makes the private key secure from leaking.

4.3.2 Git Clone

We were able to clone the repository manually on the instances but we were not able to do the same using Ansible the reason being we were using HTTP method and we were not able to protect the identity, hence failing to clone the repo. To deal with this issue we used an SSH-based cloning method which did not require us to provide our credentials.

4.3.3 Load Balancing

Loading data on frontend and backend was taking too much time as we had multiple views and they were all getting accessed by a single couchDB server that we had to hard code. We had to fix this so that the data was fetched from servers evenly, we did this by creating a .env file for backend and adding all couchDB servers to the file, and evenly distributing the load. Moreover, we used an inventory file to update the .env file dynamically so as to automate the process.

4.4 Twitter Data

Twitter data comes with lots of data analysis possibilities: trend analysis, time series analysis, sentiment analysis, and many more but this becomes really difficult when the data is 60GB in size. We encountered some issues in processing the data and grinding data from the given data.

4.4.1 Dealing with Shear Size of Data

In the initial stages we were not sure how exactly to use the 60GB file because many of us did not have that much free space on our local PC nor did we have the resources to compute the results. We tried using MRC by uploading the data to the root folder but we could not unzip the whole file. Later on we were able to use the 60Gb created volume attached to the MRC instance, which solved our computing and space concerns.

4.4.2 Data for Time Analysis

All the tweets had a created_at field which meant we had to deal with over 60 million tweets on couchDB which is not possible hence we came up with the idea that we can set up counters for the number of tweets collected per month. We set a condition that we will just capture a snapshot of 600,000 monthly tweets and analyse those tweets for daily, weekly and monthly trends.

4.4.3 Sentiment Analysis

On initial sentiment analysis it was found that the sentiment cannot be just grouped into positive, negative, and neutral as there were many tweets that had sentiment values between -0.20 and 0.20 hence we have set custom grouping for sentiment analysis of event data. We came to the conclusion that sentiment can fluctuate this much just because of mentions about events hence we are considering high positive and high negative sentiment.

4.4.4 Location Data

Format the places field in the raw data, it was quite tricky to do as the raw data has different formatting. Therefore, we compare it with the suburb data that we have from assignment 1. The logic used is quite similar to the first assignment which helps a lot in designing logic.

4.5 CouchDB

CouchDb is a very powerful NoSQL database that can create views using the MapReduce function but setting up the CouchDB cluster has its own issues, we encountered an issue that the nodes were not communicating with each other because necessary ports were not open. Furthermore, map reduce functions for time series analysis are a bit tricky as the months and days start from 0 and not 1.

4.6 Mastodon

The biggest issue for Mastodon would be the API limit which is at 300 per every 5 minutes and each API call only allows you to fetch 40 toots at a time. This means regardless how many crawlers you have on a particular server the theatrical limit would be 12000 toots extracted every 5 minutes. In our test, we have found that it takes the crawler around 3 hours to extract 1 day worth of data from Mastodon's federated timeline. Therefore, we're not able to extract all the toot from timeline and then apply filtering/map reduce function from CouchDB like we did with the twitter data. Instead, the approach we have taken is to limit the number of toots we need to fetch yet still have a comprehensive result. For toots related to crime, we decide to only fetch toots with hashtags that are highly associated with criminal offense. The advantage of this approach is that the number of toots we need to process is significantly less and there is no need for NLP because the toots with hashtags can be manually checked and we only keep hashtags where almost all of the toots are related to crime. The disadvantage would be that we're ignoring a significant amount of toot that would also be related to crime but might not have the hashtag we have chosen or does not have hashtags at all. For temporal distribution analysis we decide to only fetch toots from one medium sized Mastodon server rather than trying to fetch toots across all servers. The advantage is that the number of toots is a lot more reasonable to be harvested in time. However, the disadvantage would be the distribution we generate might be less accurate because server's popularity will have a much higher variance.

5 User Guide

In this section, we will discuss how to deploy the whole system of our project, including the ansible deployment to frontend. There are a few things to do before following these steps, first is to connect to the University of Melbourne network, which can be done through Cisco anyconnect [8]. The second thing is to get an openrc.sh file and get an OpenStack password from MRC.

Figures 24 and 25 shows how to get `openrc.sh` file and OpenStack password from MRC password.

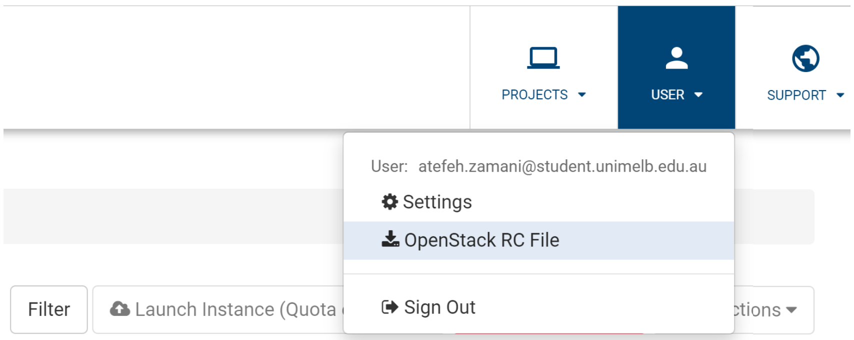
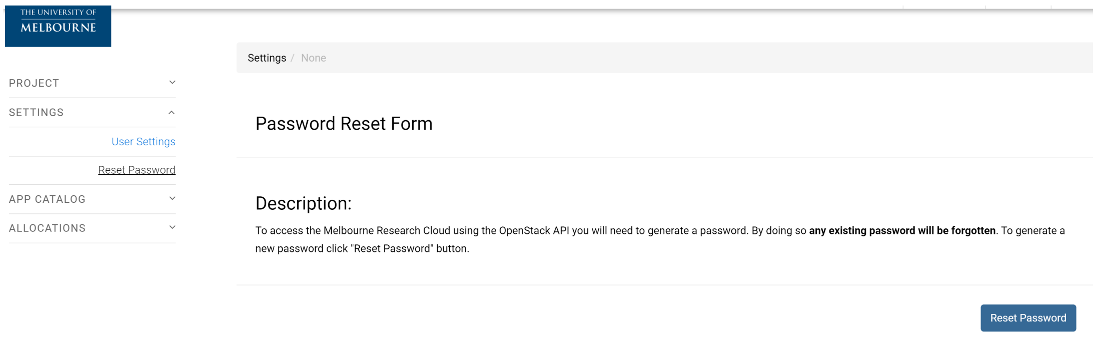


Fig. 24: How to get the `openrc.sh` file from MRC dashboard



Password Reset Form

Description:

To access the Melbourne Research Cloud using the OpenStack API you will need to generate a password. By doing so **any existing password will be forgotten**. To generate a new password click "Reset Password" button.

Fig. 25: How to reset Password and use it when prompted for openstack password

5.1 Deploy Instances

In this step we have used Ansible to deploy instances dynamically on MRC, furthermore, in this step, we have created volumes, security groups, and security group rules. The whole process can be divided into 4 sub-processes that are as follows:

1. Install python, python-pip, update pip and install openstack sdk on local host
2. Creating volumes for all instances
3. Create security groups and add rules (ports) to the group
4. Create instance with above configuration

5.1.1 Command Line Arguments

1. On shell cd into `host_vars` and then run `command python generate_application_hosts.py -n <number of nodes>` to create `instances.yaml` file containing, details about volumes and security groups and rules (this file can be edited according to security group requirements)
2. Cd ansible and run `/.deploy_instances.sh` to run the application

Since MRC is having problems we have to manually add the created instances to `application_hosts.ini` file.

5.2 Configure Instances

Firstly, before performing the step we have to generate keypairs on MRC, download the permanent key file from MRC and copy it in the ansible folder

The instances created in the previous step are essentially empty servers without functionality or dependencies, moreover, they do not have volumes mounted to them. So in this step, we firstly add `NectarGroupKey.pem` to localhost and add unimelb proxies so that we can access the instances using ssh.

(to run ssh on windows wsl run `Get-NetAdapter | Where-Object {$_.InterfaceDescription -Match "Cisco AnyConnect"} | Set-NetIPInterface -InterfaceMetric 6000` on elevated powershell)

Secondly, we install dependencies like python, pip, amongst others so that they can be used in future installations and mount volumes for all instances. Finally we install docker, install couchDB and setup couch cluster followed by adding all nodes to the dockerized couchDB cluster.

Command line: `/.config_instances.sh`

5.3 Deploying Application

In this step, we add github ssh private key to all the instances and then clone the github repository to all instances. Secondly, we deploy dockerized Mastodon crawlers on instance 1 and instance 2. Furthermore, we have deployed a dockerized backend server on instance 3 and dockerized REACTful API frontend on instance 4.

Command line: `/.deploy_application.sh`

After deploying, the application can be assessed according to its port. For example, backend can be accessed on `http://<instance 3>:5000`, while frontend can be accessed on `http://<instance4 ip>:3000`.

6 Links

A list of links to associated video, GitHub and front-end are listed in this section.

6.1 Video Demo Link

https://www.youtube.com/playlist?list=PLRqd5IXjVRXLMJK8euWrlTLvo2xrFH9vn

6.2 GitHub Link

https://github.com/atefehz/CCC2023-Team29

6.3 Front-end Link

The front-end webapp is accessible through The University of Melbourne Internet or Anyconnect VPN at:
http://172.26.129.49:3000

7 Individual Contributions

Name	**Contribution**
Zhiyuan Chen	Pre-process and upload Mastodon Data Design and Implement CouchDB MapReduce Function Design Frontend Web Application Making demo for Mastodon Report writing
Natakorn Kam	Collect and Pre-Process SUDO Data Pre-Process Twitter Location Data Design and Implement CouchDB MapReduce Function Design and Implement Frontend Web Application Design and Implement Backend System Design and Write the Report Make Demo about Frontend and Backend Upload Videos
Janya Kavit Pandya	Pre-Process and Upload Twitter Data Design and Implement CouchDB MapReduce Function Ansible Deploy Implement CouchDB and CouchDB Cluster Dockerisation of Frontend, Backend, Crawlers and CouchDB Design and Write Report Make Demo about Ansible and final Application Deploy
Nandakishor Sarath	Map Reduce Twitter Data Analysis Docker Setup Sudo Data Analysis Make Demo about Processing Twitter Data Made Presentation slides Format Report
Atefeh Zamani	Create the GitHub Repository Design and Implement CouchDB MapReduce Function Pre-Process Twitter Time Data Design the Latex File and Write Report Make Demo about Couchdb and Scenarios

Table 4: Contribution of group members into the project

References

[1] Dyouri. A. (2022, December, 21). How To Make a Web Application Using Flask in Python 3. DigitalOcean. https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3.

[2] Elon Musk takes control of Twitter in $44bn deal. (2022, October, 28). BBC News. https://www.bbc.com/news/technology-63402338.

[3] Local Government Areas - Australia. (n.d.). Opendatasoft. https://public.opendatasoft.com/explore/dataset/georef-australia-local-government-area/map/?disjunctive.ste_code&disjunctive.ste_name&disjunctive.lga_code&disjunctive.lga_name&location=3,-27.78413,132.40695&basemap=jawg.light.

[4] Melbourne Research Cloud Documentation. (n.d.). The University of Melbourne. https://docs.cloud.unimelb.edu.au/.

[5] Prabowo, R., & Thelwall, M. (2009). Sentiment analysis: A combined approach. Journal of Informetrics, 3(2), 143-157.

[6] State Suburbs - Australia. (n.d.). Opendatasoft. https://public.opendatasoft.com/explore/dataset/georef-australia-state-suburb/map/?disjunctive.ste_code&disjunctive.ste_name&disjunctive.lga_code&disjunctive.lga_name&disjunctive.scc_code&disjunctive.scc_name&location=3,-27.83417,132.40691&basemap=jawg.light.

[7] The benefits of ReactJS and reasons to choose it for your project. (2023, January, 09). Peerbits. https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html#:~:text=One%20of%20the%20main%20benefits,other%20parts%20of%20the%20application.

[8] VPN. (n.d.). The University of Melbourne. https://studentit.unimelb.edu.au/wireless-vpn/vpn.

[9] What Is An API (Application Programming Interface)? (n.d.). AWS. https://aws.amazon.com/what-is/api/#:~:text=API%20stands%20for%20Application%20Programming,other%20using%20requests%20and%20responses.

[10] Cross-origin resource sharing (CORS). (n.d.). Portswigger. https://portswigger.net/web-security/cors.

[11] What Is SQL (Structured Query Language)? (n.d.). AWS. https://aws.amazon.com/what-is/sql/.

[12] Why choose Red Hat for automation? (2022, August, 11). Red Hat. https://www.redhat.com/en/topics/automation/learning-ansible-tutoria.