Assessing the livability of Melbourne by the scale of Health, Environment and Entertainment

Group 47

Leqi Wang, 1126066, Shanghai China
leqi.wang1@student.unimelb.edu.au

Oliver Cheng, 912834, Melbourne Australia
o.cheng1@student.unimelb.edu.au

Xuxu Xue, 956895, Jieyang China
xuxux@student.unimelb.edu.au

Zhiqing Wu, 931919, Guangzhou China
zhiqingw@student.unimelb.edu.au

COMP90024 Cluster and Cloud Computing

Submission for Assignment 2

May 17, 2022

Link to Github

Link to Database

Link to Webpage

Link to Video

Link to Presentation

1. Introduction

This project demonstrates the assessment of if Melbourne is a livable city by scale of Health, Environment and Entertainment factor by analyzing the twitter data and related dataset from aurin. A Cloud base data visualization system has been developed to serve this purpose. This report would demonstrate the usage and development of the system, the findings based on the data analyzing, as well as a discussion of the tools involved.

2. User Guide

The following commands are executed under the ./Comp90024_g47/mrc directory.

There is an automated operation and maintenance tool *Ansible* which we used to achieve our goals below. It collects the advantages of many operation and maintenance tools (puppet, chef, func, fabric) and realizes the functions of the batch system such as configuration, batch program deployment, and batch running commands.

2.1 Deploy instances

To run our program in Melbourne Research Cloud, resource allocation to each virtual machine and installation of the development environment are necessary. In particular, the creation and maintenance of cloud infrastructures such as cloud instances, volumes, and security groups are required.

Firstly, we install the required dependencies on our host which are required by the Ansible playbook. Then for each virtual machine, We use NeCTAR Ubuntu 18.04 LTS (Bionic) amd64 (with Docker) as their operating system image. Volumes need to be assigned and attached to each instance, hence 60 GigaByte space was set to each instance in our project to store data. After that, we set up the security group for each instance. This operation is designed to allow multiple instances to communicate with each other and to block any potentially malicious access. For example, to allow HTTP requests between instances, we open the HTTP default port(80) for each instance. Finally, we used the *openstack.cloud.server* module for creating instances. In the meanwhile, once the instances are created, their corresponding IP address will be written in *./mrc/inventory/_inventory.ini* file so we can assign different tasks to each host

later. At this stage, every instance has been set up properly and is ready to take command.

Host	Role	Description
LocalHost	Openstack-common	Install Python-pip, openstacksdk and update pip on hosts
	Openstack-volume	Create volumes from vars
	Openstack-security-group	Create security groups
	Openstack-Instance	Create instances on NeCTAR Create a file for saving hosts and IP to "*./inventory/_inventory.ini*"
	Instances-allocation	Add hosts to Ansible in-memory inventory

Table 1. Deploy Instances

2.2 Deploy CouchDB database using Ansible

1. Install required dependencies to be able to run our couchdb on a remote instance.
   For example, python modules like "*couchdb*", and "*curl*" are needed.
2. Create docker containers for each couchdb node so they can run as if they were running on different hosts.
3. Set up the couchdb cluster by using the <*curl -X POST*> command
4. Enable the CORS policy so we can access our database in different domains.
5. Finish couchdb setup

2.3 Deploy crawler server using Ansible

1. Install required dependencies to be able to run our couchdb on a remote instance.
   For example, python modules like "tweepy", and "request" are needed.
2. Copy the crawler dictionary to the crawler server.
3. Compile the python program by using the nohup command so it will run continuously even though the connection of ssh is lost.

4. Now the program can read tweets by using Twitter API and save useful tweets to our database.

2.4 Deploy backend server using Ansible

1. Install required dependencies to be able to run our couchdb on a remote instance.
   For example, python modules like "*pandas*", and "*geopy*" are needed.
2. Copy the *data_processing* dictionary to the backend server.
3. Start the data processing program by using the nohup command so it will keep running in the background.
4. Now the program would read tweets and analyze the location and sentiments from the existing database.

2.5 Deploy front-end server using Ansible

1. Copy the frontend dictionary to the frontend server.
2. Terminate the remote front end server if it is already running.
3. Remove all existing node_modules using <*rm -rf node_modules*> command
4. Reinstall all the necessary node modules using <*npm install*> command
5. Start the front-end server by using the <*forever start*> command so it won't be terminated after the ssh connection is closed.
6. Now our front-end server has been deployed and can present our analysis results on (*http://172.26.130.158:3000*)

2.6 How to run

</bin/bash ./Comp90024_g47/mrc/run-nectar.sh>

Note:

1. To use an OpenStack cloud, authentication against the Identity service named keystone is required, which needs a token and service catalog. These two can be found on the MRC website.
2. To enable the creation of the file and save the file on your local device, the Ansible playbook would require a password for your device.

3. System Architecture

3.1 Overall Architecture

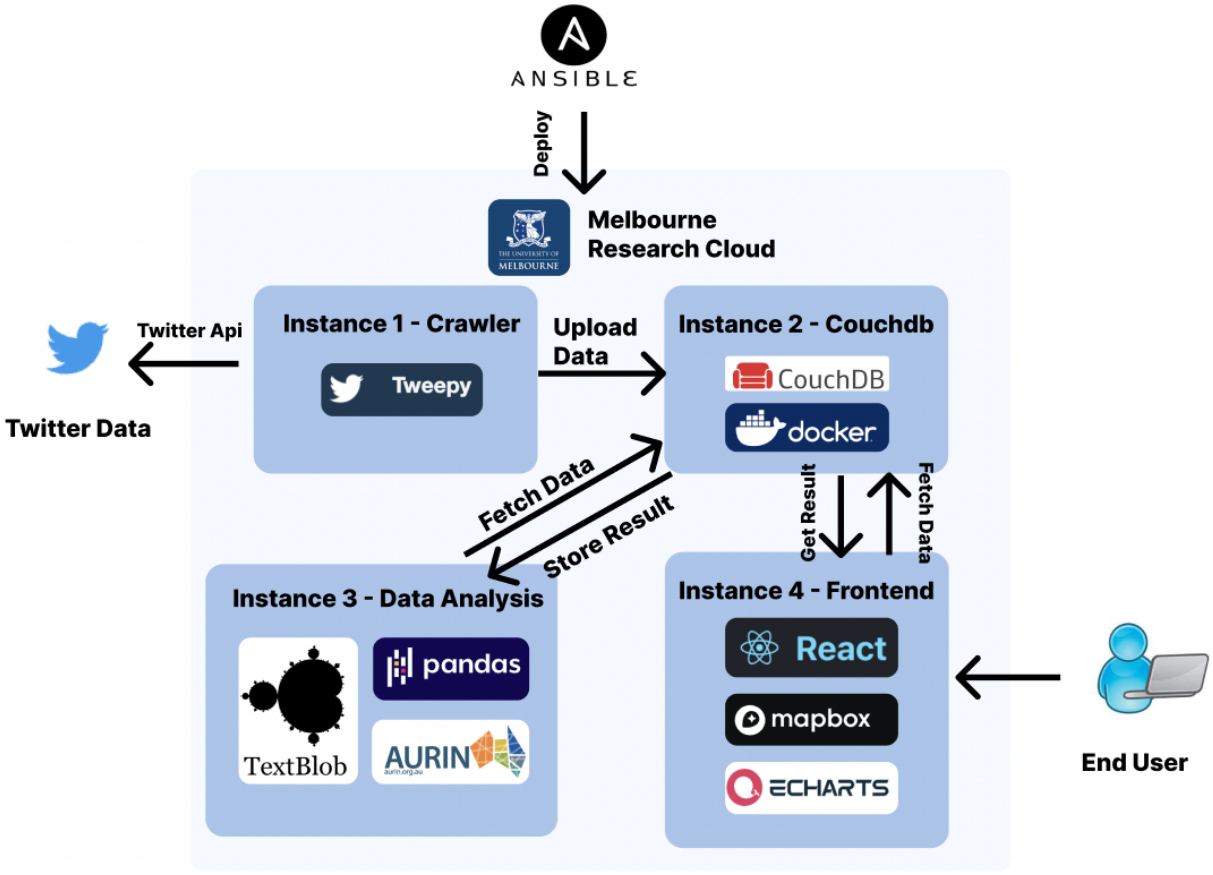

Figure 1. System Architecture

As the above diagram shows, we created the instance and assigned the instances from Melbourne Research Cloud to serve the Crawler, CouchDB, Data analysis and the live website as demonstrated on Section 1. Crawler on Instance 1 is used to fetch twitter data continuously through Twitter Api by streaming and searching, and the obtained data would be uploaded and stored to the database. CouchDB ran on instance 2 used to store the harvesting data, the data being processed and the final result. Data Analysis ran on instance 3 fetching data from couchdb continuously, processing and analyzing twitter data with the data retrieved from aurin,

as well as storing the processed data back to couchdb. Further details for the flow of data would be demonstrated in Section 3. Frontend ran on Instance 4 used to render human-readable results, which fetched results from CouchDB by http request, and end users were able to visualize the result by accessing instance 4. Details for tools used in each part would be discussed further in this section.

3.2 Crawler

3.2.1 Tweepy Library

Tweepy is a python library which is used to access the twitter API. It provided a set of powerful interfaces for accessing the twitter data through twitter API. It perfectly fits the requirement of this project by fetching twitter data for both searching and streaming methods.

3.3 Data analysis

The majority of data analysis is done using python and its libraries. The input data, comprising twitter data and aurin data, are fetched from CouchDB and the analyzed output results are then stored back to the database. The following python libraries are used for data analysis in this project.

3.3.1 Pandas

Pandas is the most popular open-source data analysis package in python that is easy to use and very efficient. It is used to read *.csv files for the aurin data and *.json files for tweet data.

3.3.4 Numpy

Numpy is used to compliment pandas when dealing with numbers and arrays.

3.3.5 TextBlob

TextBlob is used to perform sentiment analysis on twitter tweets. It has a built in trained library to assess the sentiment, either negative or positive based on a level from -1 to 1 of any given text. This is a very common solution to sentiment analysis and serves as an alternative to very complicated and time-consuming techniques using deep learning and natural language processing.

3.3.6 Geopy

Geopy is a python library used to locate geometry data. In this project, all the geoinformation is in the coordinate form, which could not be directly used to solve our research topic. Hence, Geopy is involved to convert coordinates into human-readable locations.

3.4 Frontend

We develop the user interactable interface using react framework, and embedding the visualization tool Echart and Mapbox component inside it to build an integrated webpage.

3.4.1 React.js

React.js was chosen as the frontend development tool for our platform front end framework. React is an easily used framework that allows us to develop interactable interface.
The nature of using react State allows rendering the page view efficiently. If the data changes only the involved component would be re-rendered.

React applications are composed of multiple components, which allow developers to break down complicated tasks into multiple smaller and separated components easily and clearly when developing, and compose them together eventually. Moreover, the component nature of react allows developers to encapsulate the code and re-used the part of code conveniently.

The React framework was developed by javascript, which is a common tool when developing websites. Therefore, there are plenty of libraries or tools that exist for javascript that would be suitable for developing React applications as well.
In summary, React is well-fitted in our requirement of developing this application.

3.4.2 Mapbox

We used the third-party tool Mapbox to visualize the geodata. It is a powerful geo data visualization tool that could fit multiple platforms. It provides a set of effective functionalities, which allow users to select the desired feature to visualize geo data. And the api for mapbox allows developers to operate the map through the script, which allows users to interact with the app by just visiting the built webpage.

3.4.3 Echart

The Echart library was used to visualize the data without geometry. It has an abundant type of chart, which allows us to simply choose the desired chart type. It is well-fitted in javascript development so that we are able to easily embed the chart in our application.

3.5 Containerization

The following section would discuss the reason why we use the technology of containerization to implement our project, and why we choose docker as our container tool.

3.5.1 What is containerization?

In order to understand containerization, we may first think about the physical containerization in our real world.

Imagine we have lots of different goods which need to be shipped to another location. However, we only have one ship, how should we manage our goods on our ship?
If we put it all into only one container, it would become a mess and some fragile goods would be damaged. The best way is to store each kind of good in different containers. So that containers would not affect each other. Same as the container in the real world, developers often encounter various problems when they write code and deploy it to a test or production environment after testing in their own local environment. There are many reasons why there are many bugs after deploying the code that obviously runs perfectly locally: different operating systems, different dependency libraries, etc. In a word, it is because the local environment and remote environment are not consistent.

Containerization technology solves this key problem by separating the software program from the underlying environment in which it runs. Developers package their programs into a container image after coding, which details the dependent environments and runs the standardized images in different containers, fundamentally solving the problem of inconsistent environments.

3.5.2 Virtualization Vs containerization

Each VM contains a complete operating system and binaries, which can cause duplications among the virtual machines. In addition, deploying lots of virtual machines is a waste of server CPU memory and disk space, which can eventually limit the number of virtual machines you can run on a physical server. The concept of the container allows the virtual instance to share a single host and also the drivers and the binaries and the host OS. Therefore, this approach further reduces the wasted resources, as each container now only holds the application and its unique binaries.

3.5.3 Why Docker?

Docker is required to implement the couchdb database in this project. Although there are other tools that may achieve the same goal such as Tectonic, docker is easier to learn and get some progress on it. This is suitable for this short-period project. In addition, Docker is the most

popular container tool in the world, which means lots of problems we encounter can be easily solved by googling the answer.

3.6 Dynamically scaling

3.6.1 Add IP addresses to inventory

Once Ansible creates instances and assigns IP addresses to each instance, the IP address would follow the template and save it in "*./mrc/inventory/_inventoy.ini*".
By doing this we can get instances' IP addresses easily from *_inventoy.ini* and assign them to different groups later.

3.6.2 Assign roles to different hosts

For this Project, the compilation of crawler, backend, database and frontend have been assigned to the 4 initiated instances respectively. However, if there is a change in the cloud resources, our program can still assign roles to different hosts. The way of doing this is to make the number of instances, crawler, backend, database, and frontend as a command-line argument, the program would dynamically assign roles to hosts based on these numbers.

4 Data Flow

In this section, the entire data journey from retrieval and visualization would be discussed in detail.

4.1 Data Retrievement

All the sentiment data we retrieved in this project are from twitter platform, complemented with census data that was published in aurin to analyze the correlation between Health, Environment and Entertainment factor and twitter sentiment result.

4.1.1 Twitter Data Retrievement

The twitter data would be retrieved from twitter api by using the tweepy library as indicated in Section 2.2. Both streaming and searching methods were involved, which retrieve the most recently posted tweets and the tweets for latest week respectively. Only the tweet located in Melbourne would be saved.

4.1.2 Aurin Data Retrievement

All the Aurin data have been manually downloaded from the aurin platform, and are stored locally to wait for further processing and analysis.

4.2 Twitter Data Analysis

4.2.1 Reduce Duplication

When storing twitter data into a database, the document id would be replaced by the corresponding twitter id. To check whether the current tweet has already been stored, the system would send a http get request to couchdb to obtain the twitt with its id from the corresponding database. For example, to check if the tweet with the tweet id *493802997793570817* exists in the *search_and_stream_tweets* database by compiling the following code.

```
requests.get(url=('http://admin:admin@172.26.134.66:5984/search_and_stream_tweets/493802997793570817')
```

If the response successfully received, then it shows the twitt already exists and it would not be stored to the database.

4.2.2 Twitter data processing

The stored twitter data would then be processed line by line continuously. Sentiment by using TextBlob would be calculated, and the coordinate would be converted to LGA in the same time. These 2 attributes would then be appended to each tweet. Only the *tweet id*, *sentiment* and *Lga* would be stored in the database to be analyzed as this is the only attribute we need for this project. Also, only tweet that with the LGA within metropolitan Melbourne would be stored in this database as we only going to analyze this area in this project

4.2.3 Mapreduce

In this project, the sentiment data for each LGA is required. To analyze the enormous database for twitter data, MapReduce functioned by couchDB was used. View is created in the desired database, which is used to count the number of tweets with positive, negative and neutral sentiment from each suburb in Melbourne. Then the result of count could be obtained by sending http requests in any part of this application.

4.2.4 Scenario analysis

A sentiment map would be produced from MapReduce. Then the sentiment map would be analyzed together with the aurin data to meet the demand of producing insights for chosen scenarios. The final result would be saved to the result database.

4.3 Data visualization

The live website would obtain the final result by sending http requests to the result database. The chart would be rendered once the response from the database has been retrieved.

5 System Functionality and Scenario

5.1 Scenario and rationale

To analyze the livability of Melbourne for the scales of Health, Environment and Entertainment, the scenarios chosen were the capacity of the hospital, the pollutant emission and the seat of the restaurant.

5.2 Data Interpretation

5.2.1 Overview

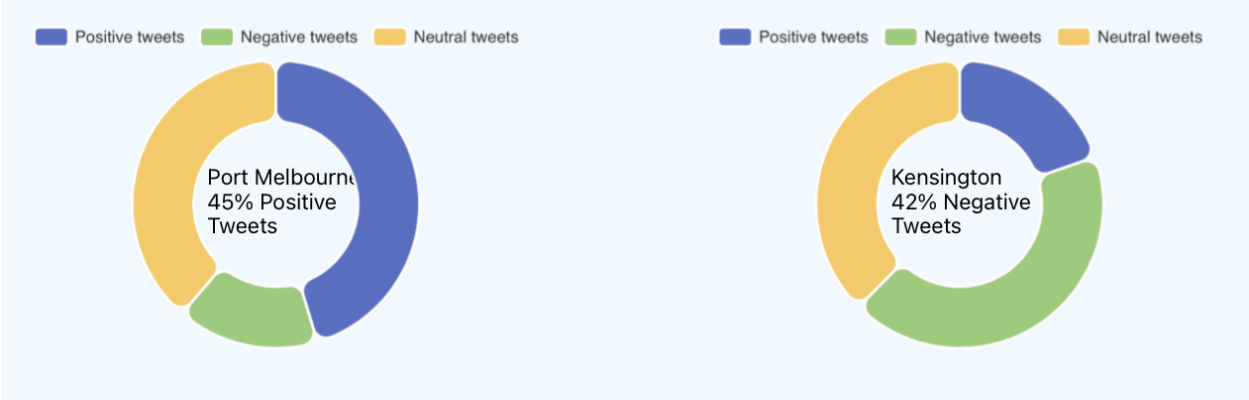

Figure 2. The suburbs have the most positive and negative tweets

Port Melbourne has the most positive tweets out of all the suburbs we investigate, which consists 42% of all the tweets being positive.
Kensington has the most amount of negative tweets out of all the suburbs, which consists 42% of the tweets being negative.

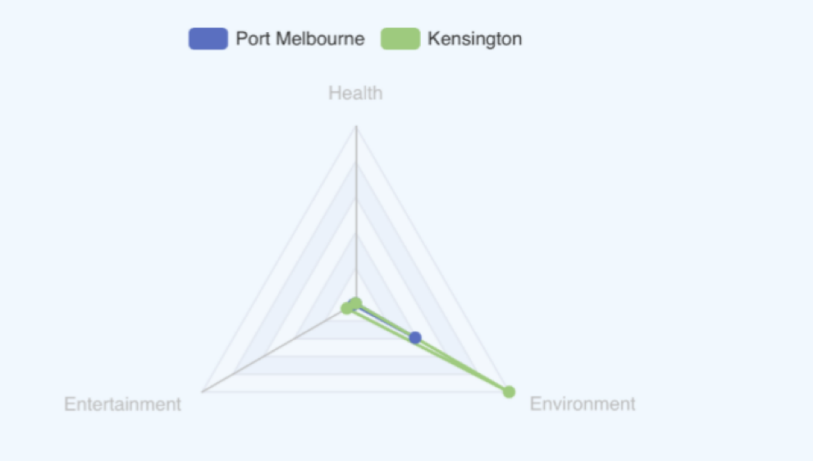

Figure 3. Livability Scales for the most positive and negative suburbs

This diagram shows how each of three factors affects the two suburbs that are at the extremes at each end of the scale.

It is clear that environmental factors are contributing to both suburbs, but the other two factors are absent within both suburbs.

5.2.2 Environment

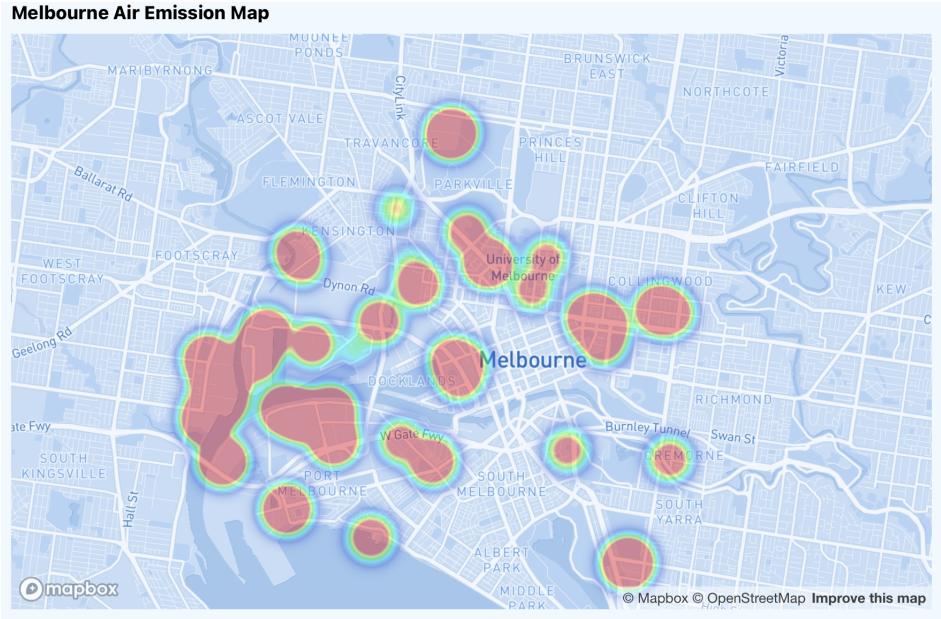

Figure 4. Pollutant emissions Map

From the map of “Melbourne Air Emission”, it is clearly shown that the pollution hotspots are consistently concentrated within the inner Western Suburbs of Melbourne, which contains the suburbs of Port Melbourne, West Melbourne, Kensington, Flemington and Jolimont.

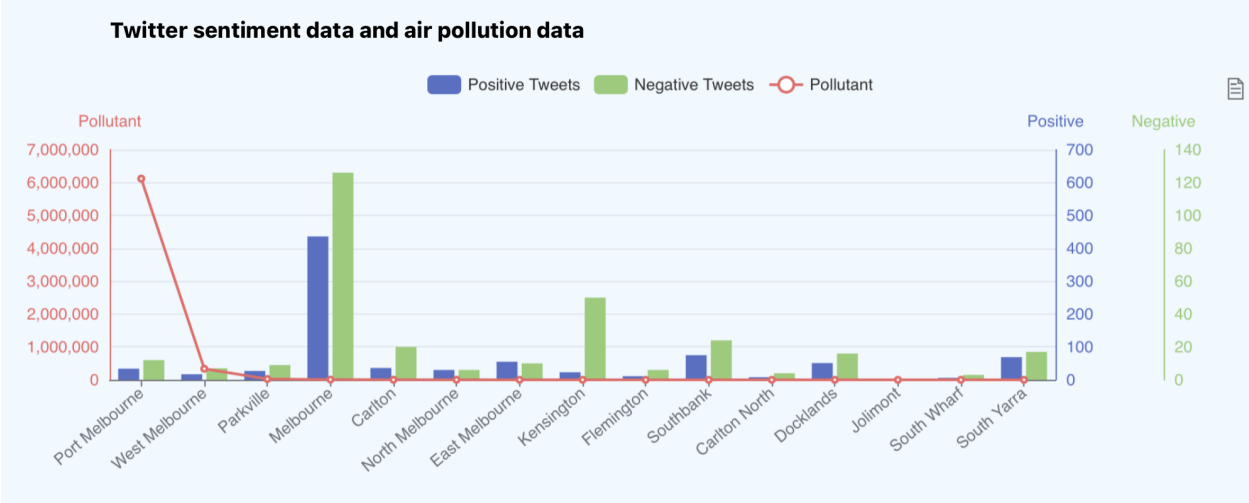

Figure 5. Pollutant Emission data and tweets sentiment data

The graph we have generated contains pollutants per suburb and compared against positive and negative tweets, the suburbs are ranked from the one that have the most air pollutant emission to the suburbs with the lowest emissions.

The negative tweets are relatively higher within (from highest to lowest) Kensington, Port Melbourne, Carlton, West Melbourne.

From the data of pollution compared with the negative and positive tweets, we find that pollution does correlate with the positive and negative tweets, with examples such as Kensington and Port Melbourne being pollution centers by area, there are significantly more negative tweets from Port Melbourne. areas such as South Yarra that are less polluted have significantly less negative tweets.

5.2.3 Health

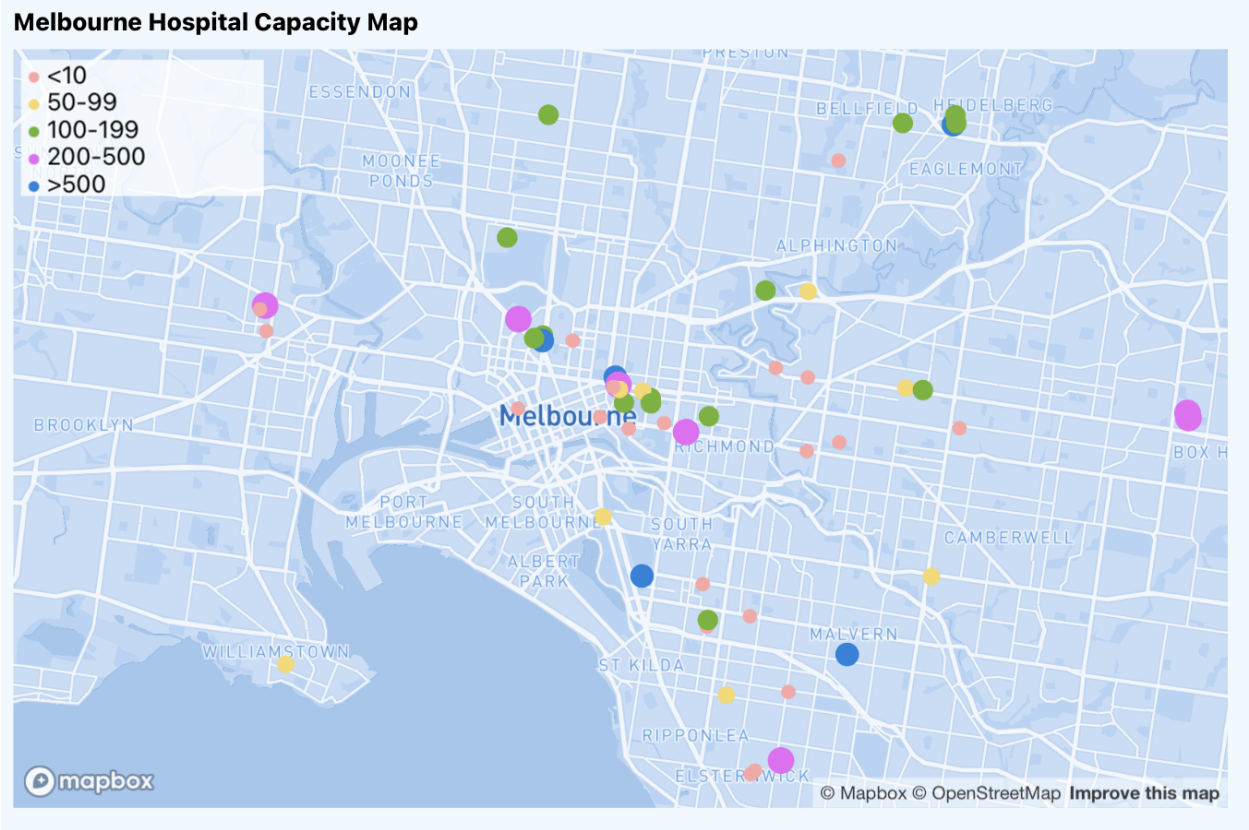

Figure 6. Melbourne Hospital Capacity Map

The Melbourne Hospitals are significantly more dense on the side of the east half of the map, hospitals are in clusters at Parkville, East Melbourne, South Yarra, etc.

The map does not show the amount of total beds within a certain area, but does show a few clusters of hospitals and the distribution of hospitals in Melbourne, which it shows where the hospital resources are more concentrated.

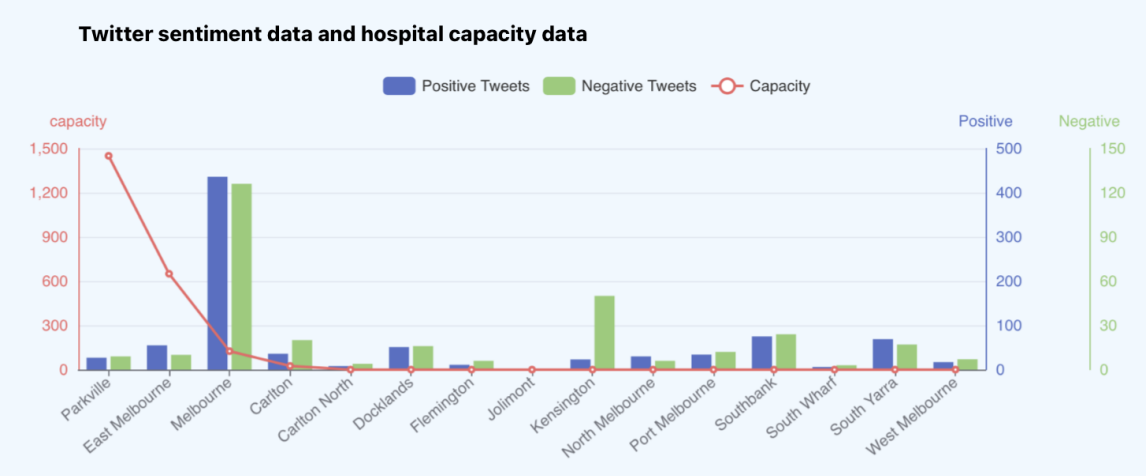

Figure 7. Hospital capacity and tweets sentiment data chart

Excluding areas are population factors, the beds are concentrated as mentioned above, it is mostly concentrated within Parkville and East Melbourne.

Suburbs within the western areas such as Port Melbourne, West Melbourne, Kensington, and Flemington have significantly less hospital resources.

The tweets are clearly showing that the eastern half of the map is having a lot more positive tweets such as East Melbourne, Melbourne, South Yarra, then the western half.

Hospital resources may correlate with the wealth distribution as well as other distributions, which makes eastern half of the map's tweeter sentiments than the western half.

From this set of data, we do find a correlation between positive tweets with the hospital capacity.

5.2.4 Entertainment

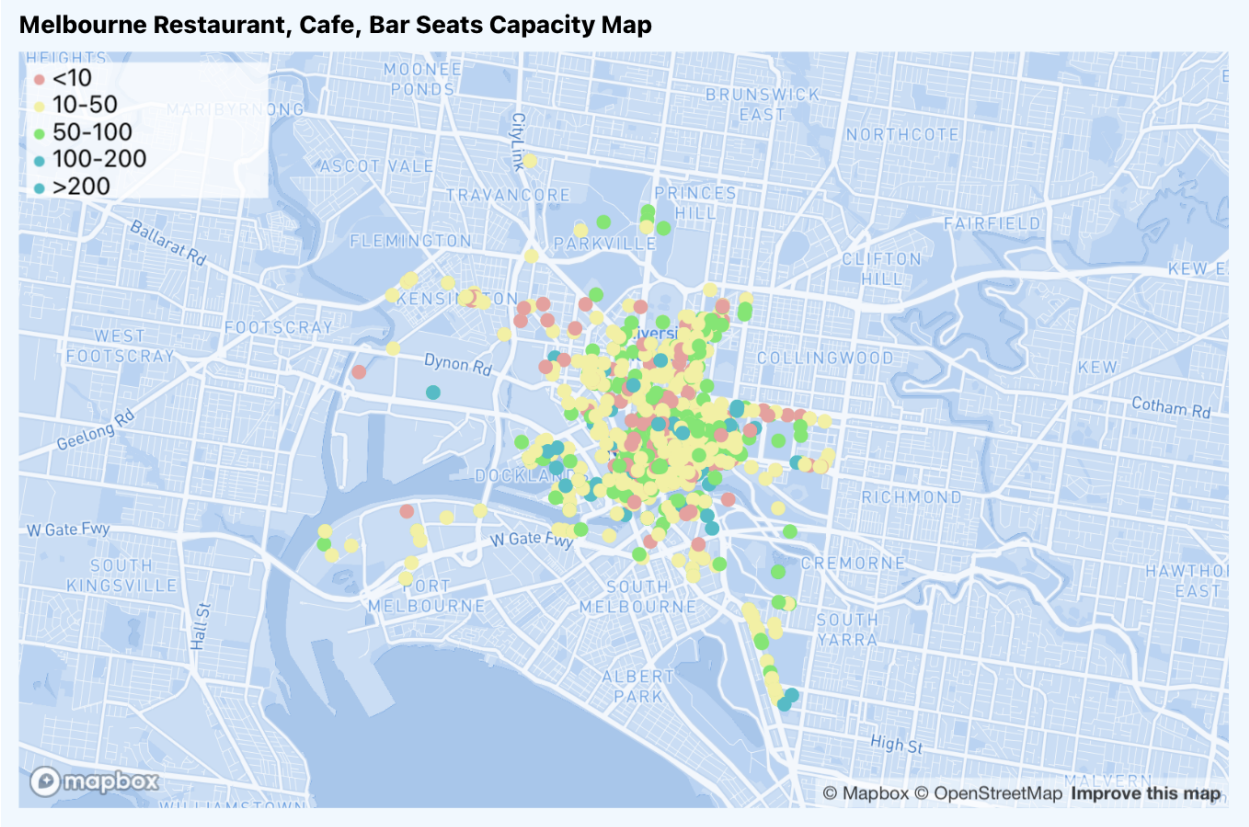

Figure 8. Melbourne Restaurant Capacity Map

The map of restaurants consistently concentrated along the city, the number of seats in restaurants, cafes and bars are very much a “city feature”.

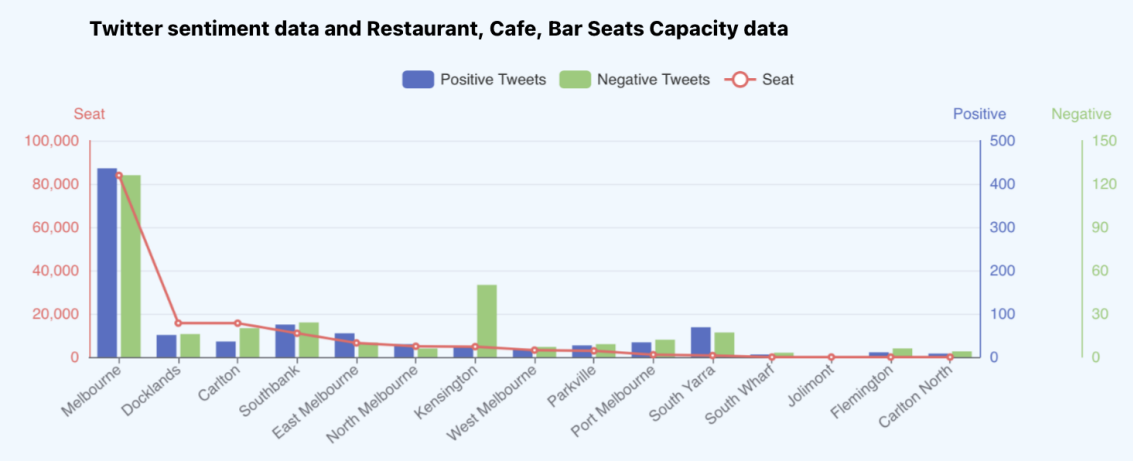

Figure 9. Melbourne Restaurant Capacity and Tweets sentiment data chart

From the twitter sentiment data we collect and the seats per suburb, we can see the seats do concentrate within suburbs close to Melbourne City, such as Melbourne, Docklands, Carlton, Southbank, etc..

South Yarra, also being a population center, has a small amount of restaurant seats, but has a very positive tweets sentiment.

Kensington at the middle has a really negative sentiment, but as an area that small as Kensington should have a lot more restaurant seats per capita, the result is not showing it translating it into more positive sentiments.

It is hard to conclude restaurant seats have correlation with tweets sentiments.

5.2.5 Result

Two of the three factors, Education and Environmental, may be part of the reasons that influences sentiments of tweets from certain suburbs, on the other hand, Entertainment (seats per suburb) may not be that important in affecting the twitter sentiments.

6 MRC Discussion

6.1 Overview of MRC and its role in our project

Melbourne Research Cloud (MRC) is a private Infrastructure-as-a-service cloud computing platform which provides free on-demand cloud computing resources for researchers at the University of Melbourne. It is based on the open-source OpenStack cloud technology.

Our resource allocation on MRC is shown in the following figure.

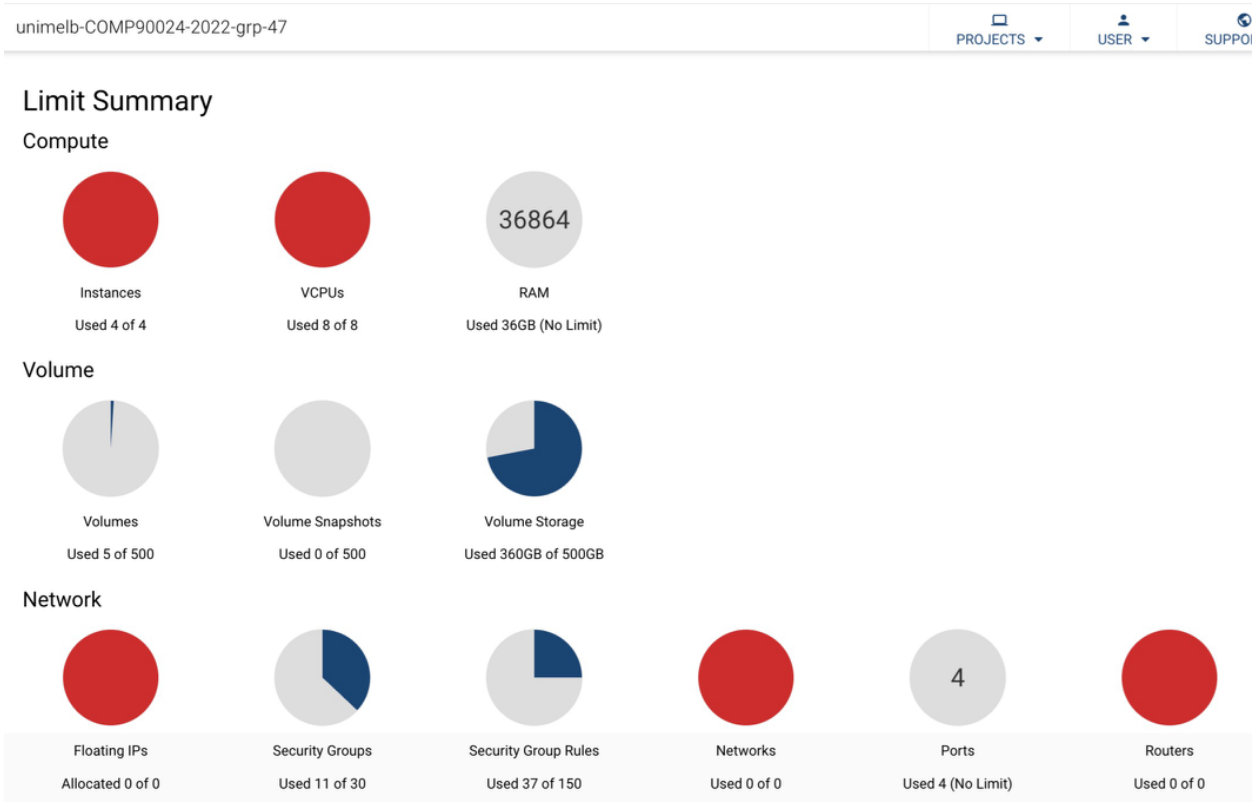

Figure 10. MRC usage overview

6.2 Advantages

There are many advantages of using MRC for this project compared to other solutions such as owning hardwares or other cloud computing platforms like AWS or Azure. These advantages could be summarized as follows based on five different dimensions: cost, security, controllability, availability and performance.

6.2.1 Cost

Cost for a web-based research project could be significant. These costs include initial investment into necessary infrastructure and management overheads as well as continual maintenance and operating costs. Due to its free to use nature for University of Melbourne researchers, MRC is a very attractive solution compared to alternatives. We do not need to buy any hardwares upfront and there are no operating expenses for using the services.

6.2.2 Security

Security is another attractive feature of MRC. As MRC is a private cloud platform, all data is stored locally and shielded from the public network. Key management tools such as encrypted key-pairs allow users to access all instances and data securely. MRC also offers security group tools that handle the user access issues among team members. Port management services are also provided that serve as an important firewall against malicious attacks and unwanted traffic such as a DDoS attack.

6.2.3 Controllability

Users have a high level of control in the MRC system. Users have full control over their allocated resources. Images could be easily built and restored and data could be easily backed-up and restored.

6.2.4 Availability

MRC offers on-demand services around the clock to users all around the world as long as they have the proper access. Users could easily build any instance with detailed tutorials and support. Industry leading backup solutions allow users to easily restore instances. Various popular pre-built images help users to bootstrap their systems with ease and speed.

6.2.5 Performance

MRC is an advanced cloud computing platform with high performance. First, it is highly efficient. MRC comprises almost 20,000 virtual cores across a range of virtual machine sizes. These hardwares are located centrally in the data center in melbourne which gives the best network performance. Second, it is very scalable. Script automation tools such as Ansible are available for users to scale their system with ease. Last but not least, it is convincingly reliable. MRC is based on the trustworthy open source OpenStack cloud technology which has withstood the scrutiny of millions of developers and attackers. Besides, MRC is actively being maintained by the leaders and experts in the cloud computing industry due to its significance in research.

6.3 Disadvantages

The main disadvantage of using MRC comes in terms of availability of services. Although MRC is considered very much available in many aspects, it is definitely not the most available cloud computing solution. This is not a surprise considering it is completely free. MRC offers a subset of OpenStack services compared to AWS. For a specific project, the allocated resources are also limited. In our example, the resources are limited to 4 instances with 8 virtual CPUs and 500Gb of volume storage. This poses constraints on the speed of data harvesting and analyzing. Moreover, even though MRC is open to researchers and relevant users, VPN is required to access outside the university network. Finally, as everything is on the cloud, users do not have direct physical access to their data. Transferring data outside the private network could be time-consuming, costly and subject to security issues.

7 Discussion and Future Development

7.1 Issues and Solution

7.1.1 Couchdb deployment

When I create the couchdb container by using Ansible, I can see all the containers are running on a remote server. However, I can not use the curl command to interact with the couchdb container. After all, I found out that the main issue is that the version of docker image which creates couchdb is out of date. Changing the docker image of couchdb fixes it all.

7.1.2 Storing data on Couchdb

Since the dataset is enormous, it is impossible to ensure the quality of data, which means there might be some inconsistent format for a single line of data. This caused the crashing of the whole processing, and it had to be fixed to allow the rest of the dataset to be successfully uploaded. Therefore, a try and catch exception process was added to the program to escape the single point of failure.

7.2 System disadvantage and future development

Although there are a total 4 instances assigned and three docker containers were used to create the setup and deployment of Couchdb, only a single node was used in this project. This might cause the inefficient use of resources since there are 4 instances that could be used in

total. Moreover, this caused the low fault tolerance rate of the application. If the database server we used for setting up the database crashes, the couchDB is not able to run on another server. Therefore, a solution for setting up and deploying the couchDB to compile on multiple nodes should be explored in future.

8 Team

Team No.47
Number of members: 4

Note: Jianian Yang joined the team in the third week because another team dissolved due to team 2 out of 4 members dropping the subject, later Jianian Yang left the subject 2 weeks before the due date (2 days before last day to withdraw) and did not contribute to the project.

Location: 3 out of the 4 members are located within China, 1 out of the 4 members are located within Melbourne. The project was cooperated online via Zoom, WeChat, Google Doc and Github.

8.1 Contribution

Main Part	Contribution	Contributor name
Front-End Server	Frontend Chart Development	Zhiqing Wu
	Frontend Map Development	Zhiqing Wu
	Front-End Deployment	Xuxu Xue
Back-End/ Data analysis Server	Aurin data processing	Leqi Wang
	Aurin data collection	Zhiqing Wu
	Twitter data Sentiment analysis	Zhiqing Wu & Leqi Wang
	Retrieve final result	Zhiqing Wu
	MapReduce	Zhiqing Wu
	Back-End Deployment	Xuxu Xue
Crawler Server	Twitter Search API	Xuxu Xue & Oliver Cheng
	Twitter Stream API	Xuxu Xue & Oliver Cheng
	Remove duplicate Tweet	Xuxu Xue
	Crawler Deployment	Xuxu Xue
Database Server	Create couchdb container	Xuxu Xue
	Set up couchdb cluster	Xuxu Xue
	Database deployment	Xuxu Xue
Report	1. Introduction	Zhiqing Wu & Oliver Cheng
	2. User Guide	Xuxu Xue
	3. System Architecture	All
	4. Data Flow	Zhiqing Wu
	5. System Functionality and Scenario	Oliver Cheng
	6. MRC Discussion	Leqi Wang
Presentation	Video	All
	Video editing	Zhiqing Wu
	Prepare PowerPoint	Leqi Wang

9 Reference List

National Health Performance Authority, 2016, MyHospitals Profile Data - Number of Beds, Retrieved from https://portal.aurin.org.au on 2022-05-16

Local Government of Victoria - City of Melbourne, 2018, City of Melbourne CLUE Cafes, Restaurants and Bistros Seats (Points) 2010, Retrieved from https://portal.aurin.org.au on 2022-05-16

Government of the Commonwealth of Australia - Department of the Environment and Energy, 2017, National Pollutant Inventory - Emissions (Point) 2018, Retrieved from https://portal.aurin.org.au on 2022-05-16