Cluster and Cloud Computing COMP90024
Assignment 2 Report

TEAM 16

Zenan JI, 1122396 - city: Nanjing

Weijie YE, 1160818 - city: Fuzhou

Wenqin LIU, 807291 - city: Guangdong

Jinhong YONG, 1198833 - city: Kuala Lumpur

Zixuan ZENG, 1088297 - city: Melbourne

May 21, 2021

Github: https://github.com/williamjz/COMP90024Group16

Youtube: https://www.youtube.com/playlist?list=PLHqviwfB_6ymNOHqE0RpAlzlxJ1D0GwiL

Abstract

As Twitter is among the most widely used social media platforms in recent days, the analysis of twitter data can reveal societal patterns in cities or countries. Understanding the feeling and distribution of cities is critical for a better understanding and, as a result, a better management strategy and plan for the future. Utilising Cloud computing, this project focuses on the analysis of twitter data across five Australian cities: Adelaide, Brisbane, Canberra, Melbourne and Sydney, visualising data on Total Tweets, Tweet Time Distribution, Sentiment Analysis, Language Distribution, as well as Top-30 Hashtags. Automation of instance creation, management, as well as security group configuration are facilitated by Ansible interacting with the OpenStack API exposed by the Melbourne Research Cloud (MRC). Moreover, deployment of the entire system onto four virtual machines (VMs) on the MRC was also done via Ansible. We containerized all services and applications by harnessing Docker. In regards to data collection, we utilised Tweepy and Aurin. Data and results were then stored in CouchDB databases in the CouchDB cluster. In addition, we have developed a RESTful realtime web application, powered by Flask as the backend, where communication with the frontend is achieved via HTTP GET requests to query data from URLs defined by the backend server running on the MRC instance.

1 Introduction

As an innovative internet-based computing platform, cloud computing provides a pool of scalable and virtualized resources as a service, such that user's data and apps are stored on cloud servers in a cloud environment, solidifying the fact that availability, scalability, flexibility, service composition, and quick service conformation are among the distinctive advantages of Cloud computing. In addition, social media has gained popularity among the public in recent years. Among them, Twitter is widely accepted across the range of age groups, varying from teenagers to the elderly. Thus, by exploring the tweets conveyed by users on Twitter, we could potentially gain valuable insights into the habits, routines, activities and perhaps even emotions of users, opening the gateway to better management approach and future planning for urban cities. Exploiting the advantages brought about by Cloud computing, this project aims to create a Cloud-based approach that takes advantage of allocated virtual machines (VMs) on the UniMelbResearch Cloud to gather tweets across cities in Australia using Twitter APIs, in which coupled with a front-end web application will deliver data visualisations for various social media analytics scenarios to be compared with the data available on the Australian Urban Research Infrastructure Network (AURIN), thus enforcing our understanding of life in Australian cities. The focus of analysis for this paper will be on the tweet distribution as well as various content across five major cities of Australia: Adelaide, Brisbane, Canberra, Melbourne and Sydney.

The rest of this paper will be organised as follows: Section 2 discusses the System Architecture behind our application. Section 3 presents our cloud-based solution utilising the Melbourne Research Cloud (MRC) platform, in conjunction with Ansible and Docker. The various content of data and its collection are further described in Section 4, followed by the storage of data in databases using CouchDB demonstrated in Section 5. Section 6 addresses various designed scenarios for this project as well as insights gained following their respective analysis. Section 7 portrays the front-end web application's structure and its communication with the back-end, ending with data visualization techniques allowing for data analysis and better user experience. Section 8 runs through error handling methods and Section 9 briefly describes the deployment guide for our developed application. Finally, Section 10 ends with a summary presenting an overview of the lessons learnt from this project along with conclusions on the analytical results.

2 System Architecture

This assignment requires us develop a rounded system on the Unimelb Research Cloud with the following functionalities:

- Harvesting tweets (both stream data and historical data) through the *Twitter APIs*
- Store the data into the *CouchDB cluster* database

- Using various data analytics (e.g. *MapReduce*) to explore a variety of analytics scenarios across the cities of Australia and also compare tweets data with the official data from *AURIN* to improve our knowledge of life in the cities of Australia
- Deploy a *webapp* to visualize our analytics result in *real-time* which provide a better understanding of the data

In order to align with all the requirements, we build up a data pipeline and responsibilities of each part are Data Collection, Data Storage, Data Analysis and Data Visualization as shown in figure 1. We first crawled the twitter data using the Twitter API and also crawled official Australia statistics data from AURIN. After that, data were stored in the CouchDB cluster with 6 databases including databases for all cities and one database for the result analysed using MapReduce and sentiment analysis. After that, data from the analysis stage and the result database will be retrieved to display in the web-based visualization frontend.

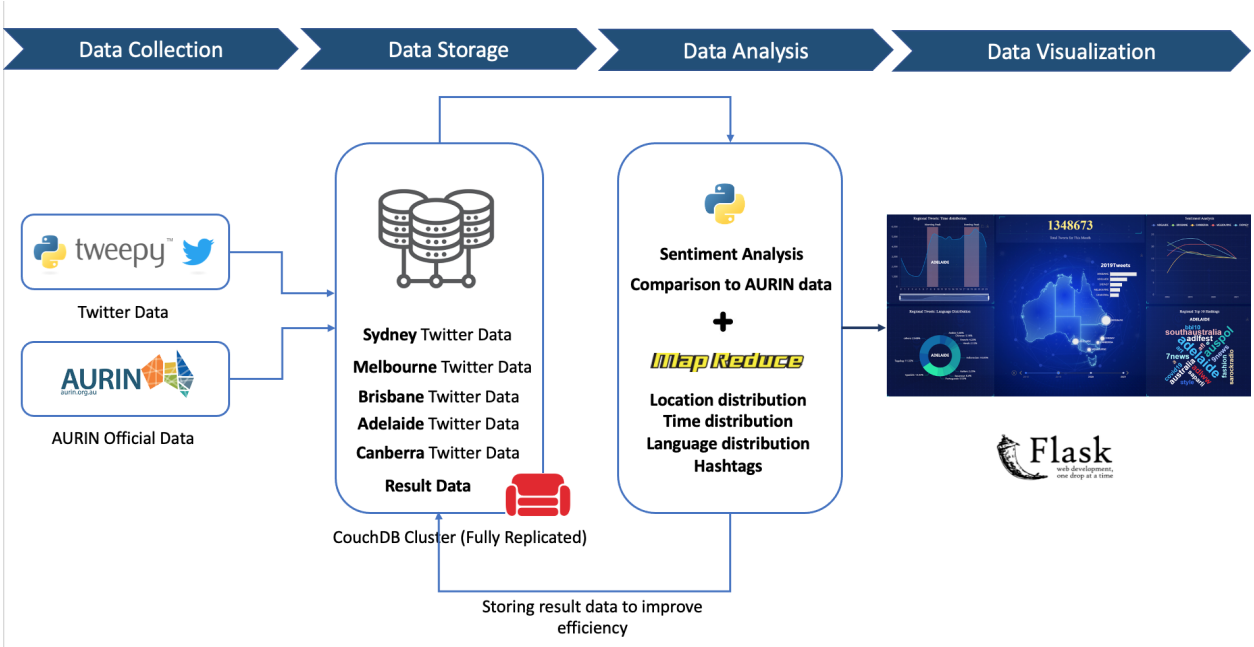


Figure 1. Data pipeline in our application

Based on the data pipeline, the system architecture was developed based on the following principles: 1) All the resources are virtualized in order to manage and allocate these resources easier once they are set up; 2) all the services should form a highly available cluster to achieve fault tolerance and reliability. The entire system is deployed on a total of four virtual machines running on the Melbourne Research Cloud 24/7. Different virtual machines/instances are served for different purposes. There were three instances (server1, server2, server3) designed as the data nodes which are mainly responsible for Twitter Harvesting and CouchDB cluster. The fourth node was designed as the processing node which not only processes and analyzes the data, but also runs the web-based visualization front end. Docker containers were used to run all the

applications (e.g. Twitter Harvesting, CouchDB, WebApp), therefore providing the opportunity for future scaling in case it is required. The entire instance creation and all services deployment on all instances are automated using Ansible. Figure 1 provides an overview of the architectural layers and their corresponding tools adopted for implementation. Further detail of each component in the system will be discussed throughout this paper.

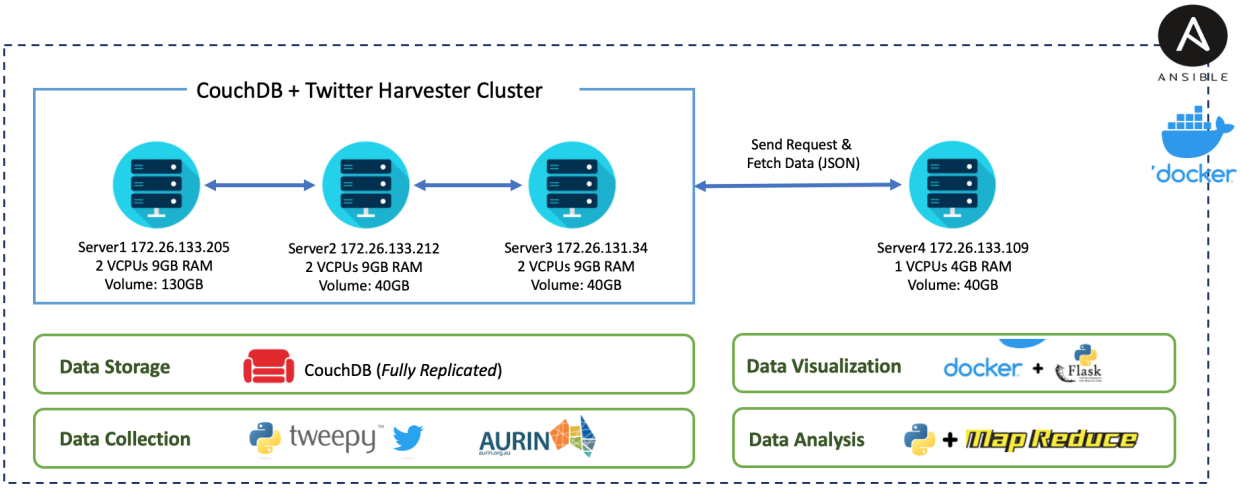


Figure 2. Overall System Architecture and relevant technology stack

3 Cloud Infrastructure

Our system was deployed on the Melbourne Research Cloud which is a free cloud service provided to us by the University of Melbourne. Furthermore, cloud management and orchestration tools such as Ansible and Docker were used to facilitate the setup and deployment of the system architecture.

3.1 Melbourne Research Cloud Infrastructure

Melbourne research cloud, a free cloud service provided by the University of Melbourne, is used in this assignment. It is used for building cloud applications and for leveraging cluster computing capabilities. The cloud infrastructure itself is built using the open-source implementation of OpenStack which allows the usage of existing libraries to access the OpenStack API. Melbourne research cloud provides us with enough features to implement our analysis system. It can run tasks remotely within a reasonable time and is compatible with other parts of our system architecture (i.e. CouchDB, Ansible, Docker, Twitter API and our web application). Overall, it has been a pleasant experience to use the Melbourne Research Cloud. Detailed pros and cons of the use of the Melbourne Research Cloud discovered during the development of our system are summarized as followed:

Pros:

- The created instances were able to run 24/7 without having to encounter significant stability problem
- Free, while major cloud providers like AWS charge on an hourly basis, and the internet traffic is metered and tolled
- The automation of instance creation & management as well as security group configuration is facilitated by Ansible interacting with OpenStack API exposed by the Melbourne Research Cloud
- A web-based dashboard with GUI is provided to monitor the instances and resources.
- The deployment scripts and source code for this assignment are interoperable with any hosting provider that runs the OpenStack software, thereby limiting the effects and allowing for future migration in case necessary

Cons:

- The resource allocation for the project did not allow the provisioning of computing instances with public (i.e. externally-accessible) IP addresses. The deployed infrastructure is therefore only available within the University network accessed by UNIMELB AnyConnect VPN, hence limiting its potential user base.
- Melbourne research cloud does not support multi-attach of volumes, making data sharing across instances harder. Dynamic resizing of volumes is also not available. To resize a volume on Melbourne Research Cloud, data already in the volume needs to be first manually transferred to somewhere else and then copied back into the newly-formed volume.
- The key-pair bound to instances cannot be changed on the fly. Instances would have to be relaunched for updating with a new key-pair.
- The resources (e.g. available hosts) might have run out.

3.2 Automation with Ansible

To enable dynamic scaling of the computing resources within the cloud environment, Ansible was applied in our automation process of creating and configuring instances on Melbourne Research Cloud by interacting with OpenStack API, as well as deploying various services (e.g. CouchDB, Flask). It also tracks steps and dependencies involved in fully deploying customized systems. Ansible runs scripts in a concurrent manner to all specified servers, and such paralleled deployment is time-efficient. Ansible modules allow executing scripts repetitively with some small tweaks without interfering with already completed actions. Such characteristics make Ansible ideal to use in the cloud ecosystem and also because it can significantly reduce the load of large-scale deployment on system administrators.

Overall, a total of 15 roles were created as part of the playbook to build up the whole system for this assignment. Tables 1-5 provide the detailed explanation of the Ansible playbook roles involved in different steps.

Step 1: Instance Setup	
openstack-common	Install pip and OpenStack on the local host
openstack-volume	Create 5 different volumes (4 volumes of size 40 GB and a volume of size 90GB) in total 250 GB which are attached to instances later
openstack-images	Show all available OpenStack images available in Melbourne Research Cloud
openstack-sg	Creating a security group to allow certain inbound traffic through certain ports (i.e. port22, port 80) from an external network. In order to allow inter-node communication between nodes in the CouchDB cluster, TCP ports 9100-9200 are opened. Also the ports 5984, 4369 for accessing CouchDB server are opened
openstack-instance	Launch instances and attach the volume and security groups to instances using OpenStack API with prescribed public SSH key. The IPs of created instances are returned by OpenStack API and added to Ansible host list. This allows Ansible to connect and to further configure the created instances.

Table 1: Roles in instance setup

Step 2: Mounting volumes to instances	
volume-mounting	Mount volumes created in step1 to instances. Ansible ensures the volumes are in ext4 filesystem before mounting the volumes to instances. Each instance is allocated with a volume of 40 GB in size, mounted to /dev/vdb. The instance designed to run backup db is allocated with another volume sized 90 GB, mounted to /dev/vdb.

Table 2: Roles in Volume mounting

Step 3: Installation and Configuration	
http-proxy-setup	Modify environment variables in /etc/environment to enable http proxy to allow internet connections
docker-installation	Install dependencies for docker, docker-compose, and pip. Dependencies are fulfilled before those packages are installed.

docker-proxy	Modify /etc/stsemd/system/docker_serice.d/http-proxy.conf and .docker/config.json to allow http proxy for docker daemon and containers.

Table 3: Roles in Installation and Configuration

Step 4: CouchDB cluster and Twitter Harvester	
couchdb-setup	Pull the readily-available Docker images provided by IBM and run the containers
couchdb-finish-setup	Create and setup the CouchDB cluster on DataNodes [Server 1-3]
couchdb-replicate	According to the variable couchdb_action in Ansible, the role either backup the all the data from CouchDB cluster to backup databases or restore data from backup databases to the CouchDB cluster
deploy-spider	Deploy the twitter harvest in DataNodes [Server 1-3] in Docker container environment

Table 4: Roles in CouchDB cluster and Twitter harvester deployment

Step 5: Data analytics and web-based visualization frontend	
git-clone-repository	git clone our Git repository and run the docker compose to activate flask web app and also enable the data analytics scripts on ProcessingNode [Server 4]

Table 5: Roles in Data analytics and web-based frontend deployment

3.3 Docker

In this assignment, we light-weight virtualization technology Docker to deploy our applications on. It is because the isolation (independent OS kernel & file system) provided by other virtualization solutions would be an overkill for our use case, and the virtualization overheads required could have more impact on the overall system performance than using Docker would do. Moreover, since the instances created on the Melbourne Research Cloud are already Virtual Machines, it would be more appropriate to use light-weight/OS-level virtualization.

Virtualizing, or ‘containerizing’ our applications with Docker comes with a few advantages: Docker container packages up code and all its dependencies and hence the application runs quickly and reliably. Each service in our application runs within its own container and exposes specific functionality to the applications that are interacting with it. Having all the components that are required for the deployment of the system available in isolated Docker containers further facilitated local development work. Since part of the system could be quickly set-up for testing and troubleshooting purposes. For example, in the case of CouchDB, we could enable a

CouchDB instance directly with one line of commands, by downloading the CouchDB image from the official repo and running it as a container. No manual installation was required. This approach also allowed us to frequently deploy new increments of the individual services as more and more features were being implemented. It also provides the opportunity for future scaling in case it is required. Docker also made our system more migratable as Docker is widely supported among all kinds of Linus distributions.

4 Data Collection

There are two parts involved in this data collection stage, Twitter data collection and Aurin data mining. We are collecting all the tweets of the five cities we are interested in (i.e. Melbourne, Sydney, Brisbane, Canberra, and Adelaide). The Aurin data is used for analysis and comparison to the tweets data we collect in the data analysis session.

4.1 Twitter Data

By the 20th of May, the twitter harvester had crawled over **4 millions** tweets in Australia in the past 3 weeks. The period covered by the tweets is from January 2018 until now. Twitter officially provides a Tweepy library based on Python, which can reduce the cost of data collection. Moreover, tweepy provides a variety of API methods, and different methods can be freely combined to provide us with the tweets we need. These tweets can be crawled in real time, and then those tweets are processed and loaded into CouchDB for storage.

Due to there being too many tweets in the network, Twitter limits the number of tweets that can be crawled. We cannot crawl all tweets indiscriminately, so we propose solutions for each different situation, including the content of the tweets, the number of tweets, and the judgment of tweet repetition.

4.1.1 Content/Number of tweets

a. Content screening

The aim of data analysis is about the cultural distribution of different regions. If we only use StreamListener provided by Tweepy, some specific contents will affect the accuracy of our data analysis. Because of the influence of language on culture, we have to analyze all languages in different regions. Listening to some key words will make data analysis biased.

b. Data quantity

In this part, we used two methods.

1) We need to collect five cities in Australia; they are Melbourne, Sydney, Brisbane, Canberra, and Adelaide. We took the center coordinates of different areas of these cities, and each city took 5 representative center coordinates. We use the 20km range as the limit. Tweepy will crawl all tweets with the center coordinate 20km as the circle area.

2) We use the box model and do not consider the boundary between cities. Melbourne as an example, we used a square with a length of 10km and a width of 10km. The start position is the leftmost side of the Melbourne area. We move the square to the right, and the distance of each move is 10km. We also crawl 2000 tweets in the square area each time, until the square arrives at the rightmost side of the Melbourne area to stop.

c. Time interval

Our group mainly crawled all the tweets in the required area from January 2018 to the present.

d. Account switching

The number of our accounts is limited, and two accounts cannot use the same api. If the same account in one api is used for crawling, the error will occur with connection failure. For different hosts, we have fixed three accounts that can only be used for the current host. When crawling, randomly select one of the current fixed accounts.

e. Thread and host settings

1) Host settings: We used three hosts to deploy crawler applications. Instance1: This host mainly crawls tweets from Melbourne and Canberra. Instance2: This host mainly crawls tweets from Sydney instance3: This host mainly crawls tweets from Brisbane.

2) Thread settings: Due to account restrictions, it is easy to connect incorrectly for continuous crawling with the same account. Therefore, we only use dual threads to speed up the number of crawls, and we define two functions, one of which is for moving right with the box model; the other thread is for downwards.

4.1.2 Identify and discard duplicate tweets

We set the Tweet ID as the index in CouchDB, which can greatly improve query performance. When we set the couchdb id, we used the tweet id provided by Twitter. Each id is unique for a tweet. Before putting tweets into the CouchDB database, we check the existence using its tweet id. Before processing each tweet, we will determine whether the current id already exists in couchdb. If the current tweet already exists, skip to the next tweet.

4.2 Aurin Data

AURIN provides a wide range of data within Australia including details regarding Australia's demographic distribution, population projections and health care services availed by its citizens. In this project, language distributions and demographic distribution of cities in interest are downloaded from AURIN and used for comparison. This data is provided by the Australian Bureau of Statistics that was released in the 2016 census data. This data allows us to identify the language people speak in each city and they provide most of the significant language distribution, e.g. Japanese, Tagalog, and so on.

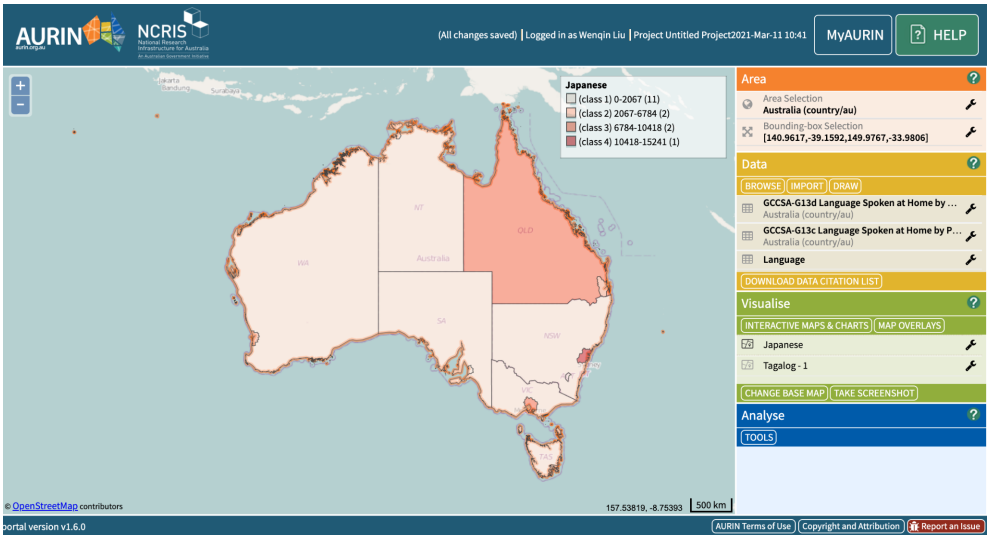


Figure 3. Example of Japanese distribution in Australia (source: AURIN)

5 Data Storage

In this application, we chose CouchDB as our database. Reason why we were using the CouchDB database and the overview of our data storage are described in this session.

5.1 CouchDB Cluster

CouchDB is an open source, document-oriented NoSQL database with high capacity and high availability released by the Apache organization. It supports storing unstructured data individually and independently which is very suitable in our case. Furthermore, the function MapReduce available in CouchDB is essential in our data analysis stage. With tons of data, we are unable to extract data each time and analyse it. However, we only need to create views based on the keys and conduct operations like sum, count, stats, and so on using MapReduce. Additionally, CouchDB provides services in a RESTful format and can easily interface with clients in various languages. Furthermore, it is easy to set up CouchDB as a cluster using Docker. The detailed configuration and installation of CouchDB are deployed using Ansible where

detailed explanation is available in session 3.2 with Ansible tasks. Last but not least, data is stored and retrieved as JSON format which simplifies our work as the front end is working with JSON which requires no additional processing step. Overall, CouchDB serves as an ideal database technology in our case.

5.2 Database Overview

There are 6 main databases in our CouchDB cluster where tweets are stored in its corresponding city and also a database to store the results of real-time analysis. Running the analysis in frontend takes up a lot of time to display the result, while directly retrieving the result data from the result database can dramatically improve the efficiency. All the databases are updated in **real-time** with a time lag of 3 minutes.

The following table summarises the format of docs store in the cities database and an example is provided:

Attribute	Description
_id	Tweet ID instead of the auto-generated id by CouchDB
date	The date when the tweet was post
time	Specific time when the tweet was post
timezone	Time zone of the region where the tweet is located at
user_id	ID of the user
place	Latitude and longitude when the tweet was post
tweet	Content
language	The language used
hashtags	Specific topic, event, them, or conversation which the current tweet relates to
sentiment	Sentiment score of the tweet

Table 8: Attributes description of docs in CouchDB database

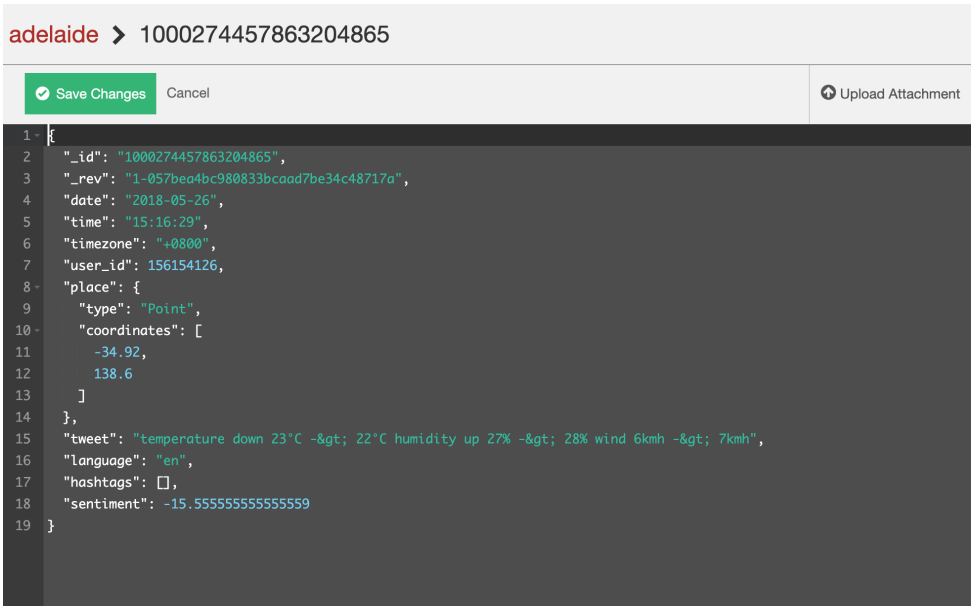
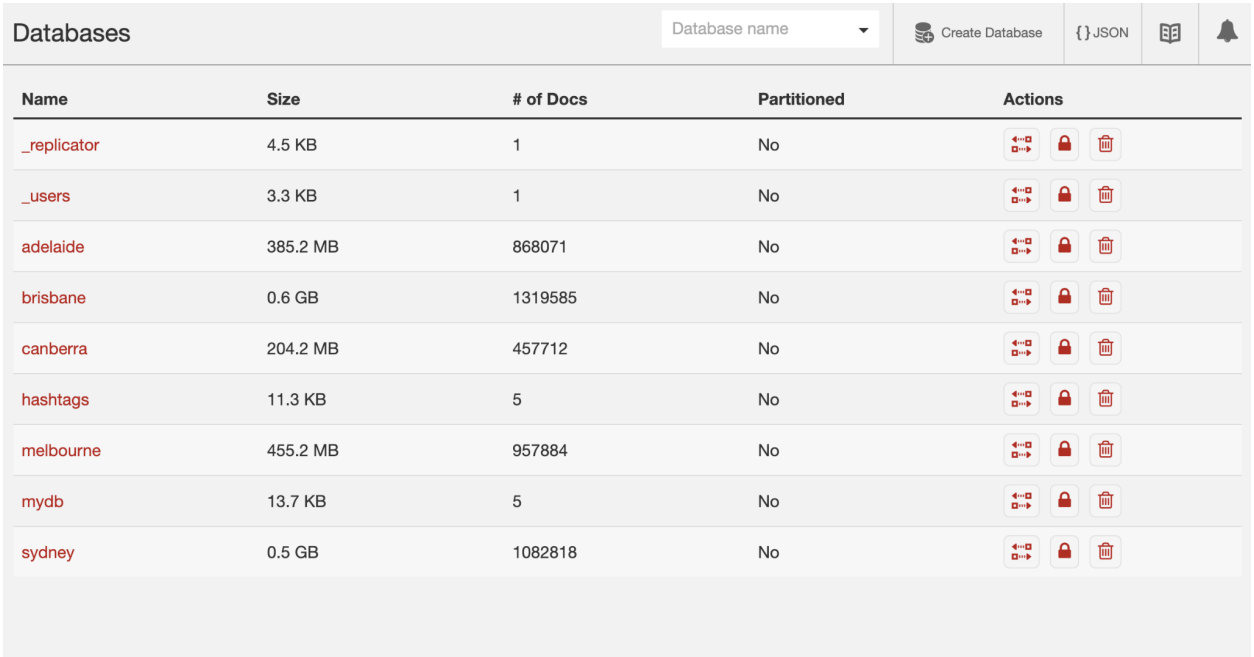


Figure 4. An example doc in CouchDB Adelaide database



Name	Size	# of Docs	Partitioned	Actions
_replicator	4.5 KB	1	No	
_users	3.3 KB	1	No	
adelaide	385.2 MB	868071	No	
brisbane	0.6 GB	1319585	No	
canberra	204.2 MB	457712	No	
hashtags	11.3 KB	5	No	
melbourne	455.2 MB	957884	No	
mydb	13.7 KB	5	No	
sydney	0.5 GB	1082818	No	

Figure 5. Overview of databases in CouchDB

6 Data Analysis

During the past decade, a plethora of social networks and applications have grown increasingly omnipresent within our communities. Twitter has become one of the most popular social media

platforms and a massive amount of twitter data is generated within a matter of seconds. It is vital to analyse these data which brings about the unearthing of important and valuable insights. Session 6.1 describes the MapReduce analytics to extract important information and session 6.2 discusses the results obtained from the analysis.

6.1 MapReduce Analytics

MapReduce is used to do some basic statistics about the tweets. Our statistics have six parts, thus six design documents for each database were created. In the end, we use the view to access and organize these analysis results. Python module cloudant is used to access the Couchdb databases, create design documents, and collect view results. This process is written as Python functions, returning a Python dictionary to store the statistics, which can be called for data visualization. We'll elaborate on the meaning of the statistics in the following paragraphs.

1. current_twts. This is how many tweets we have harvested this month (currently it's May 2021) till now. The number includes tweets from five cities, Melbourne, Sydney, Adelaide, Canberra, Brisbane. In the map function, the first seven characters of date, including the year and the month, are emitted. The reduce function is the built-in _count. The results are grouped by the string about month and year.

2. total_twts. This is about all tweets stored in the databases. The numbers are grouped by city and year. In the map function, the first four characters of date, indicating the year, are emitted. The reduce function is the built-in _count. In the end, the function returns a nested Python dictionary.

3. lang_dis. We are interested in the language distribution of tweets in each city. In the map function, the language is emitted. The reduce function is the built-in _count. In the end, the function returns a nested Python dictionary.

4. senti_score. Sentiment score of each tweet is calculated before the tweet is stored. We need to further compute the average score of each year per city. Two views are created to complete the analysis. First, sentiment_sum. The map function emits the first four characters of date, which mean year, and the sentiment score. The reduce function is the build-in _sum. The results are grouped by year. Then, another view en_count is used to count the number of tweets in English. This is because our sentiment analysis package is only valid on English text. Average sentiment score is the sentiment_sum result divided by the corresponding en_count result. The results are returned as a Python dictionary as well.

5. time_dis. We are interested in the time people in different cities are active on Twitter. In the map function, the first two characters, which indicate the hour, are emitted. The

reduce function is the built-in _count. In the end, the function returns a nested Python dictionary.

6. Top_hashtags. In this part, we aim to find the most frequently used thirty hashtags of each city per year. This can be regarded as a summary or description of life in different cities. The map function is to emit all valid hashtags. After we collect all the hashtags from the view result, we count the frequency of each hashtag, and then sort them by frequency. Then we extract the most frequently used thirty hashtags and store them in a new database.

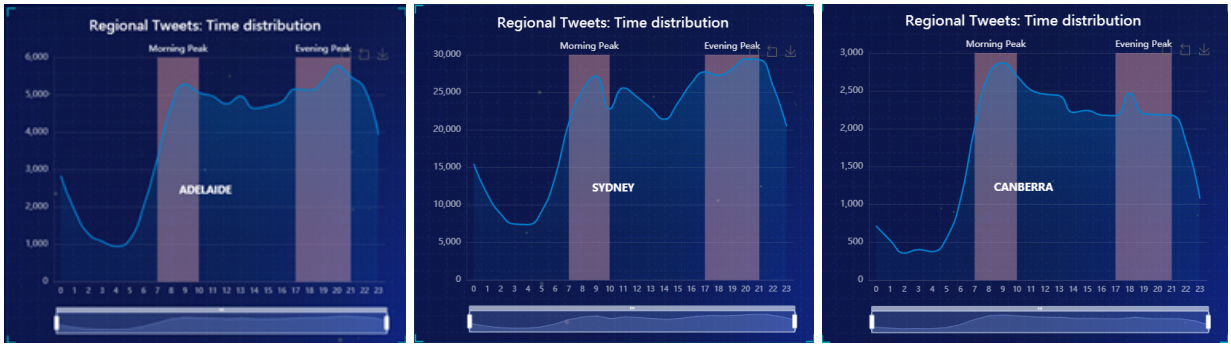
6.2 Scenarios Analysis

In this study, we analysed the overall pattern of tweets across the five cities mentioned above supported by the Melbourne Research Cloud platform. To be specific, we analysed the multiple distributions of tweets which contain information of 1) the major time when the users tweet the most; 2) the cities that have most tweets; 3) sentiment analysis of the tweets; and 4) most popular hashtags. All these analyses are conducted across the five cities during the period 2018-2021. Moreover, the language distribution is compared to the official AURIN data.

6.2.1 Distributions of tweets

Time distribution of tweets

The number of tweets posted in a day varies in time. Figure 2 shows the time distribution of the number of tweets in five cities. The number of tweets starts rising after 5am which might be because people in the cities began their day at 5am. At that time, a large number of people wake up and start posting tweets, or check the news and trend of the day. This situation lasts until about 10am and the peak of tweets posting is between 7 am and 9 am. However, the number of tweets posted in Adelaide is relatively constant from 7am to 12am compared to other cities. The most prominent one is Canberra, where the number of tweets posts peak at 8-9 am in the morning, then the number of tweets starts to decline. It is estimated that people in this region are not as concerned about news and trends as other cities.



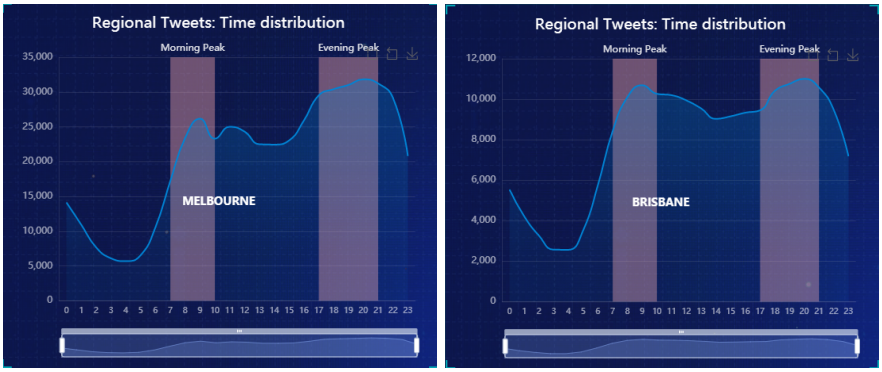
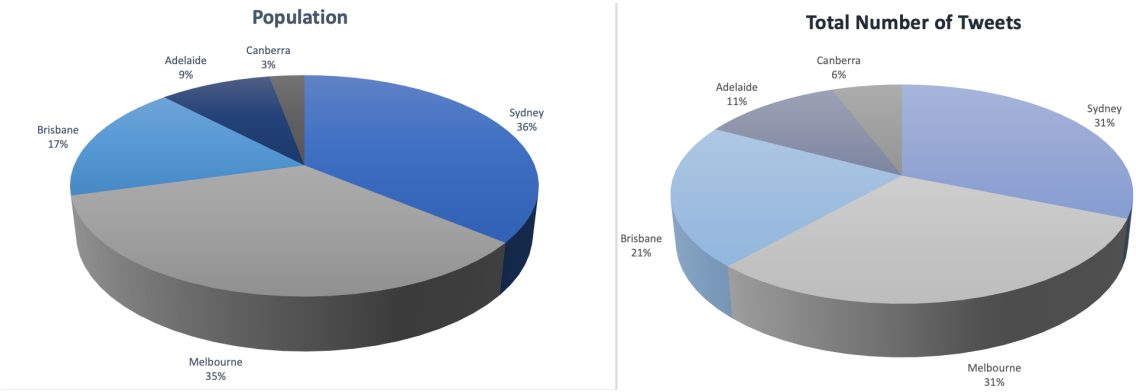


Figure 6. Time Distribution of tweets in all Cities

Location distribution of tweets

Figure 6a shows that Sydney and Melbourne have the highest population followed by Canberra and Adelaide, with Brisbane having the lowest population of them all. The demographic distribution of the number of tweets also corresponds to this same order as shown in figure 6b. Out of the major cities in Australia, Sydney and Melbourne account for more than half of the country's active tweeters. While changes may have occurred since then, the statistics in figure 6b showed that 62% of all considered cities (i.e. Melbourne, Sydney, Canberra, Brisbane and Adelaide) tweets originate from these two cities. This might be because there are more young people living in Melbourne and Sydney so we might expect a proportionate increase in the number of tweets.



Left: Figure 7a. Demographic distribution (data from AURIN)
Right: Figure 7b. Location distribution of number of tweets

Language distribution of Tweets

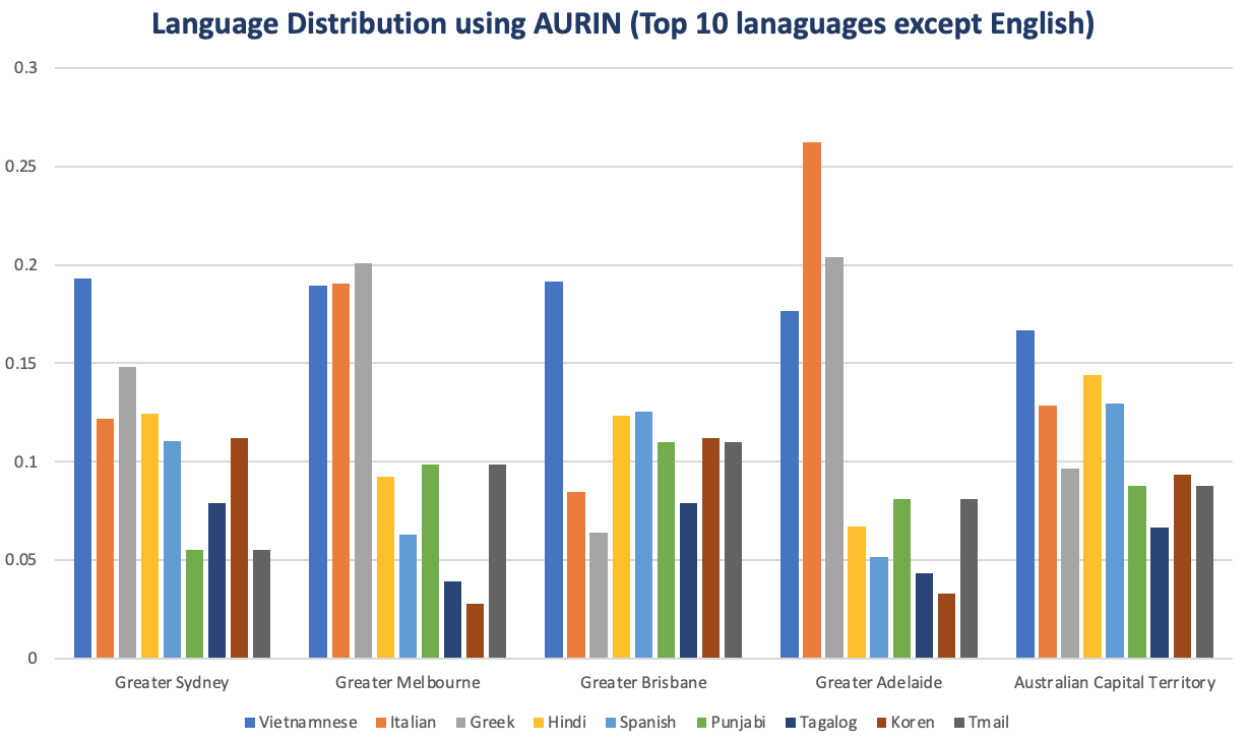


Figure 8a. Language Distribution from AURIN Dataset

Figure 8b. Language Distribution from Twitter Data

AURIN data Top 10 languages used at home (except English) are shown as above. They are common among the five cities, including Vietnamnese, Italian, Greek, Hindi, Spanish, Punjabi,

Tagalog, Korean, Tmail. There are some slight differences. Vietnamnese the frequently used Asian language among all these cities. There are more people speaking Italian in Greater Adelaide. The proportion of Greek usage is less in Greater Brisbane.

However, according to Twitter data, in Adelaide, the most commonly used languages(except English) are Indonesian, Spanish, Tagalog, Japanese, Arabic, Portuguese, Chinese. In Brisbane, there are more people using Polish. In Melbourne, Thai is the fourth most frequently used language except English. In Sydney, Portuguese, Spanish and Japanese are the top 3. Indonesian has a better proportion in Adelaide, Brisbane and Canberra, compared to Melbourne and Sydney.

There are distinct differences between the AURIN data and Twitter data. That's because the AURIN data is about language used at home, but the Twitter data is about language used on social media. We can see that some people are not very active on social media, such as people using Greek, Hindi, Punjabi and Tmail. Or the other possibility is, bilingual or multilingual people would choose to speak English on social media to be involved in a wider discussion.

6.2.2 Tweet Sentiment Analysis

Sentiment analysis of tweets provides significant insights about the cities and countries as it allows us to gain an overview of the wider public opinion behind certain topics. Knowing the overall sentiment score of the cities allows better strategies and plans for the future.

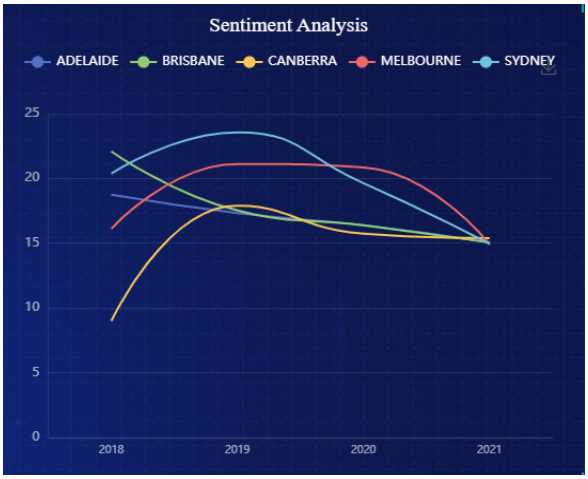


Figure 9. Sentiment scores across five cities

Figure 8 outlines the average sentiment score in five Australian cities between 2018-2021. Overall, the sentiment scores are all positive, but the sentiment scores of all cities have declined over time. This might be because coronavirus disease (COVID-19) has swept the world and caused great panic since December 2019. This might result in a decline in sentiment overall. We also analyze some other reasons for the movement in sentiment scores in different cities.

Adelaide In 2018, the sentiment score was the highest in the past three years. According to data from the Australian Bureau of Statistics in 2018, Adelaide is facing an increase in unemployment, slow population growth, and serious brain drain. These economic factors have led to a decline in the sentiment scores of people living in this area since 18 years.

Brisbane The overall trend of sentiment scores is similar to Adelaide. However, due to a series of vicious events that triggered public discussion, such as strawberry hiding needles. These events have caused people's sentiment scores to be in a state of decline.

Canberra The overall sentiment has risen sharply from 2018 to 2019 which might be because it is the end of elections and some other unknown factors. However, the sentiment decreased in the second half of 2019 because of the emergence of wildfires. The environment and standard of living have dramatically worsened which results in a decline in sentiment scores.

Melbourne & Sydney The sentiment score for both Melbourne and Sydney has increased slightly from 2018 to 2019 and stays stable in 2020. People living in Melbourne might have a happier life than the others. However, due to the close of the city during the pandemic, people's sentiment scores plummeted for a while.

6.2.3 Keywords Analysis & Top Topics

Lifestyle in Melbourne shifts a lot these years. Before 2020, topics like art, photography, love, travel, coffee, street art, music, food, sunset, sunrise, artist, cartoon, fashion, fitness are most frequently talked about. In 2020, among them, travel failed to enter the top 30 frequently used hashtags, however, topics related to covid-19 such as covid19, lockdown, stay home, social distancing emerged. Also, the Australian Open is the big event that year. When it comes to 2021, instead of general topics about lifestyle, people are talking about more specific events. Covid-19 remains a highly concerning issue. Awareness in politics increased a lot. Many events triggered heated discussions on Twitter, such as AFL, Myanmar coup, TRON(also known as TRX, a kind of cryptocurrency), BBAU (a popular TV show), MasterChef Australia(a popular TV show), Tigray Genocide, Wynonna Earp(a popular TV show), Spring St crisis, and so on.

Traffic in Victoria has been constantly a hot topic in Melbourne through these years. Besides, our data does indicate a big change. People used to talk about outdoor activities a lot, but now indoor and online entertainments are more discussed. In 2021, trending topics are far more specific than years before. People are using Twitter in a different way. Rather than sharing their personal life, they tend to express their strong opinions on some big issues now. Organizations are also cleverer now. They learned to use Twitter as an important advertising tool.

In other cities, the trend is similar, although the big events people care about might be slightly different. There are some interesting discoveries. Unlike people in Melbourne, who love entertainment TV shows, people in Adelaide pay more attention to news channels such as 7news

and 9news. They also stress more about family in their life, love rock music, are addicted to coffee, and care more about climate change. People in Sydney are active about LGBT rights. In 2018, Canberra people were obsessed with instagram, whereas, in 2021, Brisbane people are interested in writing blogs.

7 Data Visualization

7.1 Application Structure

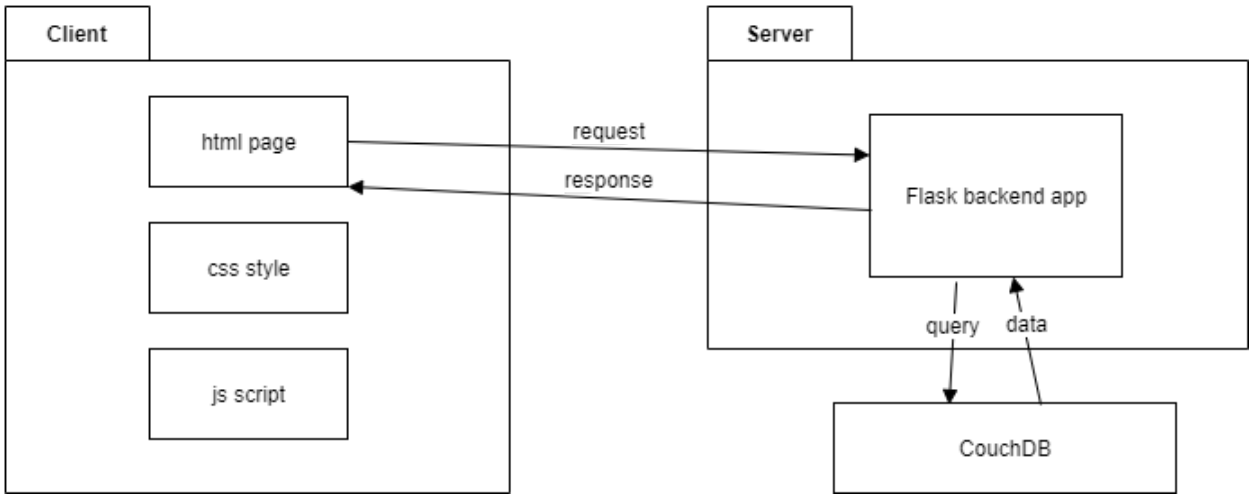


Figure 10. Web Application Structure

The web application consists of two main parts, the first part is the client side which is the frontend of the web application and the other part is the backend which is also known as the server of the program. To start the server, just run the backend app.py which contains the main code of the web application.

Backend of the server is using the flask framework, and this is a lightweight framework programmed with python. The advantage of flask is that it is very flexible and easy to implement. If any additional functionality is required, the corresponding third-party module could be imported to fulfill the requirements or just call the functions defined in other python files at the backend directory. The disadvantage of flask is that when handling tons of requests, it might be slow because it would take turns to handle the requests, once at a time, but for this project it should be good enough.

The frontend of the application is just static files, which is a html page, and it uses the css files to define the style and js files to define the functionality for the displaying charts and the map.

7.2 Communication between frontend and backend

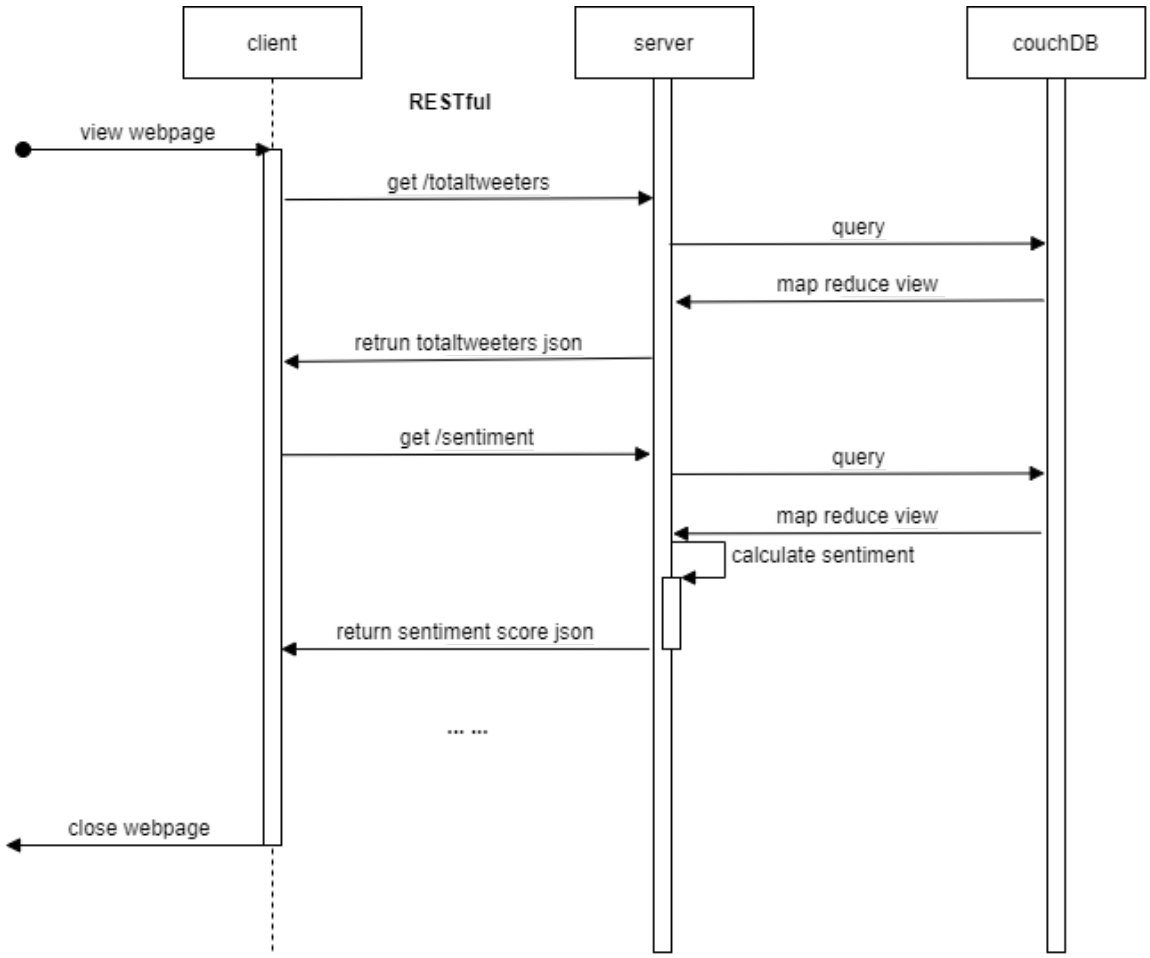


Figure 11. Sequence diagram of the program.

This application is a RESTful application that when the user browses the web page, it makes a request of the html page to the backend server running on the instance. And then every data for showing and analyzing are also achieved from different URLs, for example, if the frontend wants the sentiment score for different country, it could send a GET request to the URL host:/sentiment, and the backend would look up the relating data in CouchDB, and processing the data in the backend, then return the json representation of the data to the frontend. The frontend has multiple charts for displaying different information, and they are received from the data from the database through the GET request to various URLs defined at the backend.

The reason for processing the data at the backend of the application is that the client side of the web is not safe, the data might be easily leaked if the progress of analyzing happens at the frontend.

7.3 Data Presentation: Echarts

In this project, we addressed Echarts, a declarative framework for web-based data visualisation as our charting library to present data. The decision was made based on the fact that various charts have been made readily available by Echarts, not to mention the vast amount of creative examples from the community base, thus eliminating the need for building charts from scratch, saving time for the benefit of development velocity.

Throughout the project, the focus of analytics and visualisation were the five major cities of Australia: Adelaide, Brisbane, Canberra, Melbourne and Sydney. By harnessing the data collected from the GET requests, we display five separate charts in total: geographical map of Australia, scatter and bar chart of total yearly tweets with timeline, time distribution of regional tweets, language distribution of regional tweets, as well as the top-30 hashtags of the respective cities.

7.3.1 Map Chart

Map charts were used for the visualisation of geographical data, allowing us to obtain insights relating to different regions of Australia. With respect to our case, the map of Australia was used as a background, as well as an interactive tool to switch and display data for other visualisations. GeoJSON data was sourced from the web and generated using Echarts, comprising seven geographical regions: Western Australia, Southern Australia, Northern Territory, Queensland, New South Wales, Victoria and lastly, Tasmania, which is shown in Figure \ref{}. For the sake of stability, the map was not zoomable. Since no data related to the seven respective regions were investigated, they will not be displayed. The center coordinates of each respective city mentioned above were used to locate their respective locale on the Map Chart. In conjunction with the data obtained via GET request to the URL on host:/totaltweeters, a scatter chart was created atop of the constructed map chart.

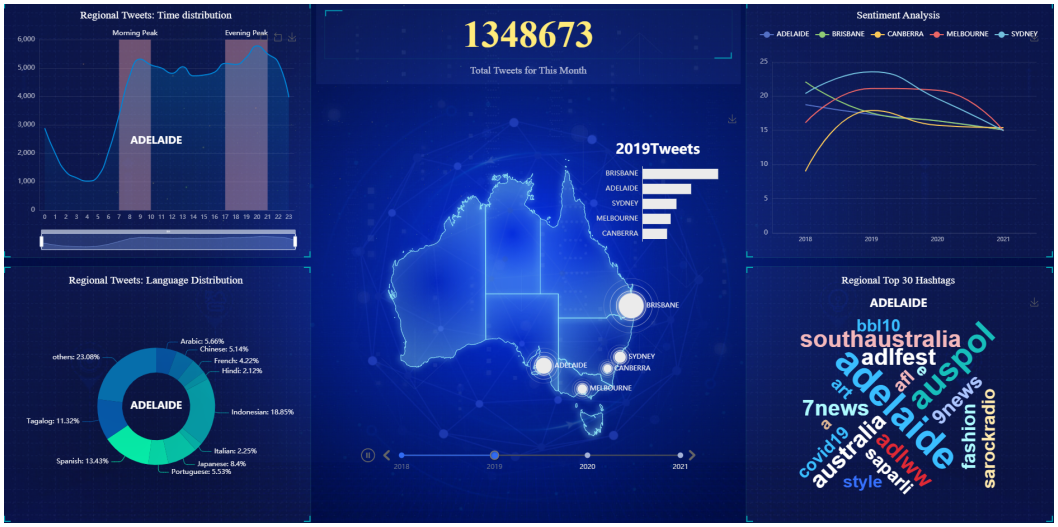


Figure 12. Visualization of the webpage

7.3.2 Scatter Chart and Timeline

Scatter charts are great for visualising data points in an easy yet straightforward manner, allowing the user to clearly distinguish the highs and lows of data points swiftly.. In our case, scatter plots can potentially present a large amount of data points across the map, instantly exhibiting the relationship between various interesting locations in Australia and the total number of tweets in that area, as shown in Figure 11, represented with white circles. This variable can, however, be changed to any other data having a location tag to be projected on the map. For example, we could investigate the population instead of the total number of tweets in a particular area.

Moreover, scatter charts could be used in conjunction with bar charts as portrayed in Figure 11, where the values of data points can be examined in a sorted order, allowing them to be used for further analysis by a user. On top of that, the scatter points on the map chart can be triggered to switch between data in various locations for other charts to be discussed in sections below.

In addition to the presented visualisation, a timeline feature was added to the mix mentioned above, which upon a click event, will conjure the desired data on the map chart regarding that particular year. By default, the timeline is looped infinitely according to a fixed interval, until the pause button is triggered, which stops the timeline from looping. This will allow the user to observe the gradual change in data density while looping, and the data in specific time ticks when paused.

7.3.3 Line Charts

The time distribution line chart can be seen from Figure 11. Line graph was used to portray the activity of twitter users over time, in conjunction with markers which indicate the peak hours in the morning and evening, which could potentially provide insights into the correlation between peak hours and tweets.

Furthermore, stacked line charts were utilised to demonstrate the sentiments of the five cities from across a fixed time range (4 years). This allows better visualisation when comparing the sentiments of cities across the time series.

7.3.4 Language Distribution: Pie Chart

Pie charts were used to characterize the spread in languages across the respective cities, due to their ability to accurately illustrate the percentage held by each language. A color gradient similar to the background was chosen for the user's better visual experience.

7.3.4 WordCloud

A straightforward way of visualising the top-n hashtags across cities would be through a WordCloud, as seen from Figure 11. Hashtags with higher occurrences will appear larger in the wordcloud, instantly diminishing the hashtags will lower frequencies.

8 Error Handling

Two types of error need to be handled in the harvester program: Twitter API error and database error.

In terms of Twitter API error, we assume that there is no error in the configuration file for the search subject. The most frequently encountered error is the connection timeout of the Twitter developer account. For such cases, we have added parameters of the waiting time and the number of attempts to the connection. It will be fault-tolerant to try a few times, if it still cannot connect, or the account is rejected by Twitter to log in. We will quiesce the current account for a period of time, and we randomly select another account to continue crawling. If all accounts are banned, we have a second plan to use our own crawler program to crawl tweets in real time.

In terms of database, each CouchDB server performs as a replica in the cluster which stores the same data as each other to ensure high availability. As a result, when one or two nodes are accidentally down, the cluster is still able to serve the request from the frontend. Moreover, we replicate our data to the backup database with continuous replication provided by CouchDB. This continuously monitors the changes in data in the CouchDB cluster and backs them up in an incremental manner. With this action, if any database is deleted by mistake, we can still recover most of the data from the backup database. In addition, when there is more demand and need to scale up the system, we can just add more nodes to meet the demand of the system using Ansible.

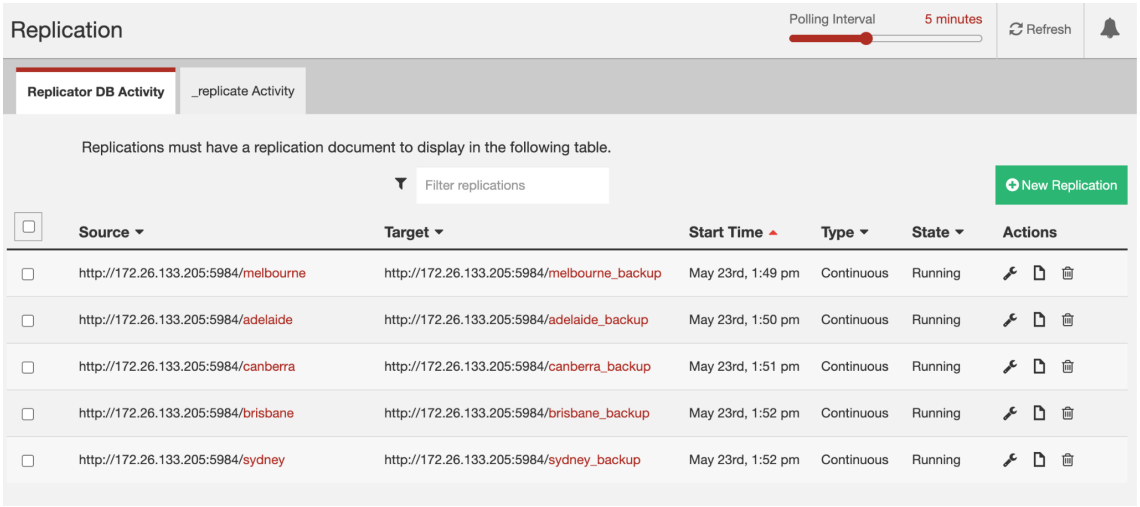


Figure 13. Replication in CouchDB

9 Deployment Guide

- Ansible 2.10.7

1. Make sure you are connecting to the Unimelb AnyConnect VPN even though you are using the campus network. Ansible may hang if being run from the unimelb network without proxy

2. Clone the repo from https://github.com/williamjz/COMP90024Group16.git and change directory to mrc using command $cd Ansible/mrc

3. To deploy, there are two cases:

(1) Create instances and set up remote instances and start the service.

$. openrc.sh; ansible-playbook -i inventory/host --extra-var db-action=backup master.yaml

(2) Remote instances are all set and only run the service.

$. openrc.sh; ansible-playbook -i inventory.host --extra-var db-action=backup webapp.yaml

10 Conclusion

In this project, we have learnt a lot about cloud and clustering capabilities as well as the mechanism behind them. Particularly, as none of the team members has any prior experience in Ansible, automation with Ansible had posed great challenges for the team. Nonetheless, it is overall a pleasant experience and the performance of the overall system is acceptable. With the advantages of the research cloud and the couchDB, the team was able to work asynchronously. Additionally, the MapReduce function in CouchDB has also made analysing data more convenient, saving plenty of time compared to extracting data and processing them externally. As a matter of fact, the twitter data alone is insufficient if we are intending to uncover major trends representing the country. For future improvement, we could apply a similar approach to analyze other social media platforms such as Instagram and Facebook to generalize the application in obtaining a more trustful conclusion for the country.