Cluster and Cloud Computing Assignment2 Report

City Analytics on the Cloud

Team 8

Xiaoyi Han 1130315 Chongqing
Ka Hou Hong 1178062 Perth
Ziyuan Xu 1124236 Taizhou
Xiaoyue Lin 924655 Melbourne
Xingyang Shen 928713 Chengdu

Keywords: CouchDB, Docker, AURIN, Flask, React, MRC, vulgar tweets

Contents

Abstract

In order to facilitate communication, social software is getting closer to people's lives. The information expressed on social software can also reflect people's true feelings and thoughts. In addition, as the popularity of social software increases, tweets that are from some users can be obtained. This paper will study real-time tweets in Australia, and crawlers are used to obtain these tweets. The keywords in the tweets and external datasets will be used in this project. The research will be conducted to observe whether there will be exciting phenomena or results.

1. Introduction & links

With the development of technology, online social media has become very popular. The amount of data for social media also becomes very large. New tweets are produced even every second. By analyzing the data in Australia, some essential information such as location, time, and emotion can be extracted. To combine the information from tweets with external data from other sources like AURIN, some interesting conclusions and phenomena may be gained from this project. Additionally, real-time tweets are used in this project which can make the study meaningful. Due to the size of the tweets data is quite large, a cloud solution architecture that satisfies Openstack protocol is used to deal with the data.

As for this project, crawlers are used to obtain tweet data. The data obtained through the crawler and the data of aurin are combined for analysis and processing. There are the 250 GB Couchdb and three other virtual machines for data storage and deployment. This project mainly focuses on swear words and other attributes which are related to Australian people's life. This project will focus on five main scenarios.

1) Compute the percentage of explicit tweets in each state.
2) Use a bar chart to study the top 3 commonly used vulgar words in each state.
3) Make a word cloud of vulgar words to detect which swear words are more commonly used in the whole Australia.
4) Compare the word cloud of hashtags with swear words tweets to that without swear words tweets in each state.
5) Compare the vulgar percentage to the average of annual personal income of each state.

1.1 Links

- Github link: https://github.com/erichong0815/COMP90024_Assignment2
- Video link 1: Cloud-based solution with dynamic deployment and crawler explanation https://www.youtube.com/watch?v=Ut3a5CFF46E
- Video link 2: CouchDB utilization and frontend scenarios visualization https://www.youtube.com/watch?v=xXDzmZHzU_Y
- Demo Slide link: https://docs.google.com/presentation/d/1gnVthEg6l9-BD5qJXGccGCIrEmOR3MMtdziJS-EXzd4/edit?usp=sharing
- Other links:
    - Front-end: http://172.26.131.136:3000/
    - Back-end: http://172.26.131.136:5000/

- CouchDB: http://172.26.131.136:5984/

2. User Guide

2.1 Ansible Deployment - how to run

To run the ***1-deploy-vm.sh**:* (1) cd OpenStack_Docker_Couchdb
(2) sudo ./1-deploy-vm.sh
(3) Enter the OpenStack Password

To run the ***2-deploy-setup.sh**:* (1) cd OpenStack_Docker_Couchdb
(2) sudo ./2-deploy-setup.sh
(3) Enter the OpenStack Password

To run the ***3-deploy-crawler.sh**:* (1) cd OpenStack_Docker_Couchdb
(2) sudo ./3-deploy-crawler.sh
(3) Enter the OpenStack Password

2.2 Deployment details

The OpenStack_Docker_Couchdb folder contains all the scripts used to deploy instances and this project. This project divides the deployment into three steps:

File name	Description
1-deploy-vm.sh	Instances deployment
2-deploy-setup.sh	CouchDB, front-end and back-end deployment
3-deploy-crawler.sh	Crawler deployment

Table 1 - deployment files

2.2.1 Instances deployment

To deploy instances, we execute the shell script file ***1-deploy-vm.sh*** which runs ***1-deploy-vm.yaml*** ansible playbook, which works with an instances configuration file **host_vars/work.yaml** to manage the configuration of instances and cloud infrastructure.
The following shows the detail of 1-deploy-vm.yaml ansible playbook:

Hosts	vars_files	roles	Description of roles
localhost	host_vars/ work.yaml	openstack-common	• Install and update pip, install openstacksdk
		openstack-images	• Retrieve and show all available Openstack images
		openstack-volume	• Create volumes on NeCTAR

		openstack-security-group	● Create security group and rules
		openstack-instance	● Create instances on NeCTAR

Table 2 - “1 - deploy-vm.yaml” ansible playbook

After the instance deployment is successful, we fill the IP address into inventory/ application_hosts.ini for the further deployments (CouchDB, front-end, back-end and crawler).

2.2.2 CouchDB, front-end and back-end deployment

For the deployment of CouchDB, front-end and back-end, ***2-deploy-setup.sh*** is used to run ***2-deploy-setup.yaml*** with application_hosts.ini, this play book works with host_vars/work.yaml and host_vars/workremote.yaml.

The following shows the detail of 2-deploy-setup.yaml ansible playbook:

hosts	vars_files	roles	Description of roles
couch	host_vars/ work.yaml	openstack-proxy	● Add proxy in /etc/environment
		docker-configuration-couchdb	● Install dependencies using apt ● Update pip ● Install couchdb using pip
		docker-installation	● Uninstall old version of Docker ● Install dependencies using apt ● Add Docker apt repository key ● Add Docker apt repository and update apt cache ● Install Docker ● Install docker-compose
		docker-proxy	● Create proxy directory for Docker ● Create proxy file for Docker ● Add proxy for Docker
		docker-create-network	● Create docker network, which is used between CouchDB, front-end and back-end
	host_vars/ workremote.yaml	docker-pull-image-couchdb	● Pull an image of couchdb
		docker-create-container-couchdb	● Create a docker container from existing couchdb image
		couchdb-setup	● Check if couchdb is created ● Create a database "clean_tweet_by_search"

			● Create a database "vulgar_tweet_by_search" ● Copy mapreduce file from local to remote hosts ● Copy crawler file from local to remote hosts ● Create clean_tweet_by_search design document ● Create vulgar_tweet_by_search design document
		deploy	● Copy front-end from localhost to remote host ● Copy back-end from local to remote hosts ● Run docker compose front-end ● Run docker compose back-end

Table 3 - "2-deploy-setup.yaml" ansible playbook

2.2.3 Crawler deployment

For the crawler deployment, we execute the shell script file ***3-deploy-crawler.sh*** which runs ***3-deploy-crawler.yaml*** ansible playbook with application_hosts.ini, the play book works with host_vars/work.yaml and host_vars/workremote.yaml.

The following shows the detail of 3-deploy-crawler.yaml ansible playbook:

hosts	vars_files	roles	Description of roles
worker	host_vars/work.yaml	openstack-proxy	● Add proxy in /etc/environment
		install-dependency	● Install dependency using apt ● And update pip
		docker-installation	● Uninstall old version of Docker ● Install dependencies using apt ● Add Docker apt repository key ● Add Docker apt repository and update apt cache ● Install Docker ● Install docker-compose
	host_vars/workremote.yaml	docker-proxy	● Create proxy directory for Docker ● Create proxy file for Docker ● Add proxy for Docker
		couchdb-run-crawler	● Copy Crawler folder to Instance ● Install dependencies using apt ● Install docker-compose ● Start the crawler python script using Docker Compose

Table 4 - “3 - deploy-crawler.yaml” ansible playbook

2.3 End user invocation/usage

1. As a user of our system, he/she can get accessed through the URL: http://172.26.131.136:3000/ to see our visualising results in real-time.
2. As a system manager, he/she can get accessed through the URL: http://127.0.0.1:5984/_utils/ to visit and manage the database’s information with username and password.

3. MRC, Resources allocation, Docker

3.1 System Architecture

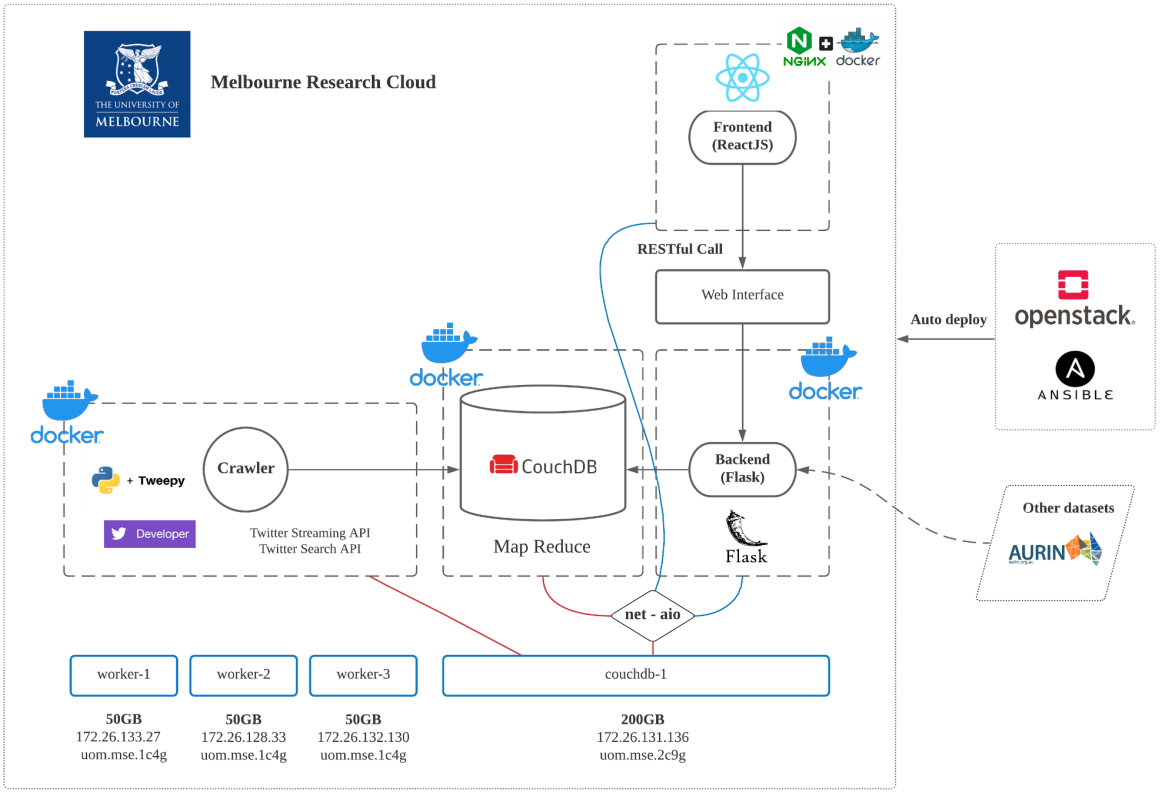


Figure 1 - System Architecture

This is the overall design of our system architecture. The whole System is deployed on the Melbourne Research Cloud(MRC). The 3.2 section will introduce resource allocation and discuss the advantages and disadvantages of using MRC. The 3.3 section will explore the containerization design of our system.

3.2 MRC utilization and discussion of pros and cons

3.2.1 Resources allocation

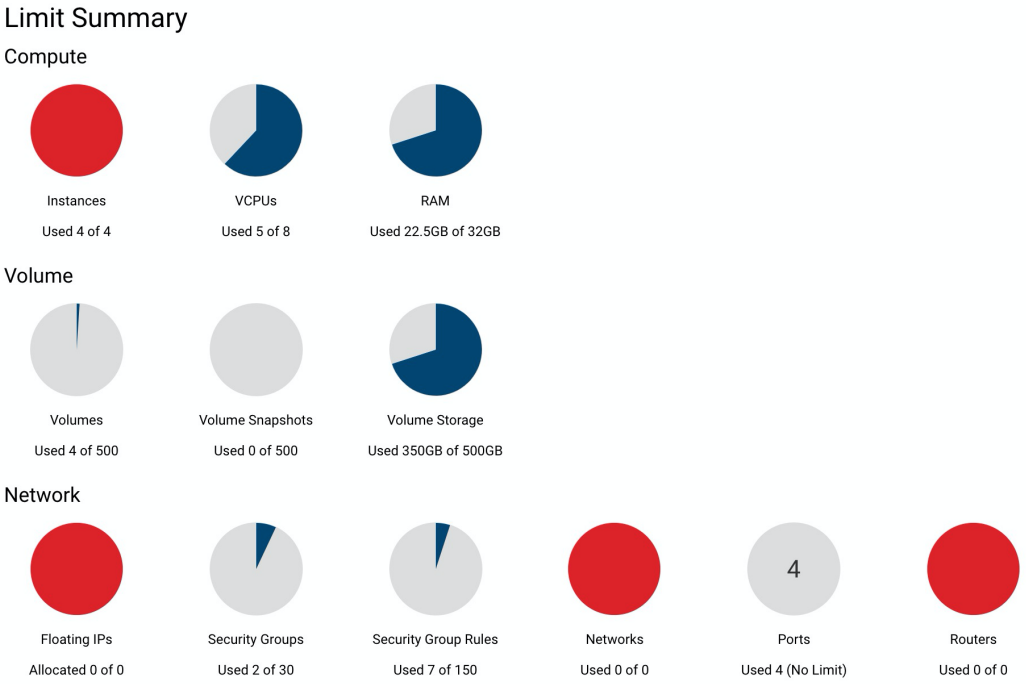


Figure 2 - Resources allocation

1. Instances arrangement and volume allocation

In this project, we allocated four instances with eight virtual CPUs, 22.5 GB RAM (One 2c9g and three 1c4g), and 350GB of volume storage on the Melbourne Research Cloud. The following table will show more details.

Instance name	IP Address	Flavour	Volume size	Deploy portion
couchdb-1	172.26.131.136	2c9g	200GB	CouchDB, front-end and back-end
worker-1	172.26.132.130	1c4g	50GB	Crawler
worker-2	172.26.128.33	1c4g	50GB	Crawler
worker-3	172.26.133.27	1c4g	50GB	Crawler

Table 5 - Instance configuration

For couchdb-1, because it needs to store many tweets and also supports front-end and back-end, couchdb-1 requires a large volume and more RAM. In terms of other instances (worker-1, worker-2, worker-3), they are responsible for collecting tweets and importing the processed tweets into CouchDB. For each worker instance, a Flavor of 1c4g and volume size of 50 GB is sufficient.

2. Instance Image and Security Group

We choose a relatively new NeCTAR Ubuntu 20.04 LTS (Focal) amd64 image, qh2-uom-internal, melbourne-qh2-uom availability zone for all instances and set up a security group named ubuntu_security_group.

3. Proxy Settings

We also add proxy settings followed by our workshops to access the public network. Docker-related files add the same proxy settings as well.

3.2.2 Advantages and disadvantages of MRC

The Melbourne Research Cloud (MRC) gives a private cloud model. It organizes on-demand computing resources for researchers. It can provide similar functionality as commercial cloud providers such as AWS, Azure.

Advantages:

1. MRC supports OpenStack, which provides lots of associated virtual servers and other resources. OpenStack offers a free and open-source software platform for cloud computing for Iaas mainly.
2. MRC has the ability of elastic scaling, and users can deploy computing problems on the MRC. If the CPU has limitations, it can automatically upgrade, even without reinstalling the software.
3. Many available image options in MRC; when we design our system, we are able to choose suitable images to start research.
4. Costless

The cost is one of the significant pros of UniMelb Research Cloud. Our project also benefits from this because the initial investment of hardware and software is very high. Compared with other public cloud services, we do not need to pay for daily maintenance costs. MRC offers enough resources for us to start research.

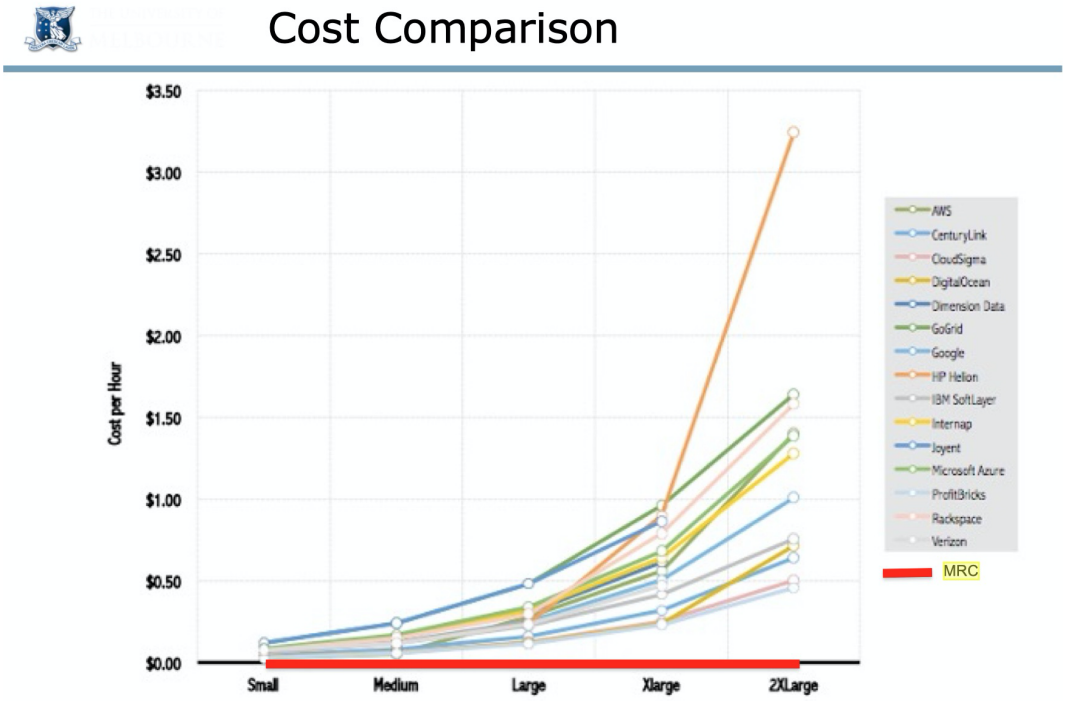


Figure 3 - Cost Comparison of Different Cloud

5. It is possible to access the space and same resources on MRC as a team, which is suitable for team development and research.
6. MRC provides Snapshot backup, which can avoid data loss.
7. Users can creatively customize their cloud service, including software installation and configuration on MRC.

Disadvantages:

1. Not all the OpenStack services are available on the MRC.
2. MRC web web-based dashboard without guidelines in detail, which can be a little hard for novice users to follow.
3. Snapshot creation must wait a lot of time, which is regretful to conduct data back-up.
5. MRC does not officially support AWS EC2.

3.3 Containerization with Docker

The primary benefit of containerization is to keep the consistency between production and deployment. Teams use different operating systems, including OSX, Windows, Ubuntu, which cannot ensure the runnability of each piece of the code. It may take a lot of time to debug. The container provides a new solution that the program can run anywhere on the container.

For development, we can input “docker-compose up” to set up all the environment and test codes. When deploying the service to the server, we use ansible scripts to build images. Besides, containerization also has better performance in terms of security. Different containers communicate through virtual networks. The advantage is that we can design to deploy a database and back-end on a single server; only the back-end can access the relative database. This kind of network isolation lets the load-balance server listen to bind to 0.0.0.0 and listen to all the ports within the virtual network and only expose the port we would like to accept requests from the container. Hence, we have more flexibility and scalability about the ports that we want to accept, instead of changing the source code, but only the container settings. In our system, we take advantage of this feature to accomplish communication.

3.3.1 Why Docker?

We adopt Docker as a containerization solution in our projects. All parts of deployment are based on Docker, which includes CouchDB, web app, and crawler deployment. Docker is one of the most popular containers and takes up the dominant position in containers. Compared to other container providers, Docker has valuable learning resources online, offering us a possibility to get familiar with them in a limited time. Docker community DockerHub provides many extended versions of the docker image but still retains the customizing ability. Docker has a building-by-stage feature; different parts can share the same environment, which reduces scripts, and keeps the container lightweight.

3.3.2 Dockerfile and Docker-compose

Our system set up three Docker files and corresponding docker-compose.yml files to customize the runnable environment. Docker-compose uses single-host deployment. It uses Yaml scripts, which can be quick and easy to configure. Docker-compose also has high productivity. It reduces the time to perform tasks, and it is easy to start and shut the program with one single command.

- Front-end

Our ReadJS front-end application uses a node environment to build and then choose a more lightweight and high-performance web server Nginx to quickly serve static files and send requests to proxied web servers or act as a load balancer. Nginx exposes the 80 port and mapping to 3000 when using docker-compose.

- Back-end

The Flask back-end is based on python 3.7, and sets FLASK_RUN_HOST as 0.0.0.0. It also set up pip to install all packages in requirements.txt.
It exposes the 5000 port.

- Crawler

Crawler's Dockerfile defines python 3 for the environment. It sets up pip to install packages directly.

3.3.3 Setup docker virtual network

Our system chooses to use three docker-compose.yaml files to start corresponding applications, and CouchDB is also docker-based. In the couchdb-1 instance, we create a virtual network named "net-aio", which enables the communication between CouchDB, web app, and crawler. To avoid creating a new default network for each docker-compose. We assign the network "net-aio" for each of the three docker-compose files so that the back-end and CouchDB can connect. The front-end and crawler can also point to the instance's exposed host and communicate with the back-end or CouchDB.

4. Tweets collection (crawler)

These tweets were collected from Twitter using tweepy, and the filter was set to collect only English tweets from Australia using the bounding box location ***[72.25, -55.32,168.23, -9.09]*** and ***['en']***. Each tweet was preprocessed and checked if it was not a duplicated tweet. Only the unique tweets were imported into CouchDB; other duplicates were discarded.

4.1 Tweets preprocessing

Before the web crawler is started, an id_list is initialised to store all the unique tweets ids. Also, 1670 vulgar words are stored in the vulgarword_list to find the offensive words in the tweet.

4.1.1 Streaming API interface

When the tweet is received, the program firstly extracts the ***'id_str'*** in the tweet and checks if the id already exists in the id_list. If the tweet is unique, we will use the vulgarword_list to find any vulgar word that matches with the word in the list. if any vulgar word is found in the tweet, the program uses a new list to store all the profanity words that have appeared in this tweet. After the inspection step is completed, a new dictionary key will be generated to record if the tweet contains a profanity word and what profanity words appeared in the tweet. Finally, the tweet is saved into the database. Besides, we prepared two databases to classify vulgar tweets and clean vulgar, increasing the efficiency of finding nasty tweets and explicit tweets.

4.1.2 Search API interface

When we use the search API interface, we first need to enter the parameters about the desired tweets, which includes (i) query string, (ii) geocode, (iii) language, (iv) result type, (v) start date, and (vi) end date. When the search request is made, we can receive about 100 tweets each time. The

way to preprocess and classify the tweets is the same as the way we implemented in streaming. Although the search API can only crawl a limited number of tweets, streaming cannot crawl to previous tweets, but the search can do.

4.2 Other tools or materials in tweet harvesting

Bounding box tool: https://boundingbox.klokantech.com/
We used the bounding box tool to find the Australian bounding box, enter the keyword ***Australia*** and convert the result format to ***CSV,*** the result 72.25,-55.32,168.23,-9.09 will be shown.

Profanity list: The profanity-list.txt is found in:
https://github.com/whomwah/language-timothy/blob/master/profanity-list.txt
It initially had only 816 lines with a lot of word sets, after we split the word sets and removed some words that we think are not profanity, the total lines of the file became 1670 lines (words).

5. Database Component and AURIN Data

5.1 CouchDB

Tweets are stored in CouchDB, which is a document-oriented NoSQL database [3]. CouchDB stores data in JSON format and supports parallelized execution of MapReduce. The built-in Web application, Fauxton, makes the database convenient for administration and testing. Collected tweets are stored in two partitioned databases based on whether the tweet was labeled as an explicit tweet or not. Hence, the two databases each contain explicit and clean tweets. By separating the tweets when storing them, the summary operations of MapReduce can be done separately and faster when summarising vulgar word frequency, for example. Tweets from different states are stored in different partitions in the databases. By storing data in logical clusters, restricts each query to one single partition at a time. This makes the data query faster and easier to test since the data analysis will be done state by state. Each tweet in the databases has a unique document ID, which is made up of the partition ID (i.e., the name of the state it is from) and the original unique identifier of the tweet. As each tweet only comes from one state, this further ensures that no duplicates are stored in the databases. Tweets were regularly backuped in local CouchDB databases. The CouchDB databases were deployed using the Docker container.

5.2 MapReduce

Data analysis was performed using CouchDB's built-in MapReduce capabilities, which can be used for document filtering and data extracting. CouchDB provides HTTP APIs for querying MapReduce Views and uses JavaScript as its query language [3]. Mappers are used to filter required documents and key/value pairs, while Reducers perform a summary operation such as counting and generating word frequencies and return results. In our implementation, three MapReduce functions were designed to suit our analysis topic for the two databases, each storing explicit and clean tweets:

- **countTweetByStates**: counts tweets from each state.
- **vulgarWordFreqAU**: yields used vulgar word frequencies in tweets in Australia
- **hashtagFreq**: yields hashtag frequencies in each state

MapReduce functions were written in JavaScript in a design document called design.json, which was deployed and added to both databases using the Docker container. Views were constructed by the query functions, which were rows returned by Reducers. Each row contained aggregated information of each state, for example. Views were called for data analysis using Python client API for CouchDB in later steps.

For the configurations of CouchDB, *reduce_limit* was changed to false because reduce overflow errors occurred when viewing the aggregated hashtag frequency in Victoria and New South Wales. The errors occurred because these two states had a large number of tweets with many hashtags that had only appeared in the same state of data once. Thus, even after summarising the frequency of each hashtag, the size of the reduce output might be almost as large as the partition of documents and didn't shrink rapidly. Therefore, we changed the configuration to avoid this error and it has been tested that the output was expected.

5.3 Aurin data utilization

Alongside the Twitter data, Statistical Area Level 4 (i.e. SA4) Estimates of Personal Income data was retrieved for analysis from the Australian Urban Research Infrastructure Network (i.e. AURIN). The data set contains estimates of the mean personal income of each SA4 in the financial year of 2013-2014. The state then grouped the data to obtain each state's average annual income, which was treated as an approximation of the current income status in each state. We are interested in seeing any relationship between the prevalence of swearing in tweets and income in each state.

6. Back-end Component

6.1 Back-end design (Flask)

The back-end server is deployed on the instance named "couch-1" based on docker. The server is developed in the Python language with the Flask web framework. Flask provides the service to complete RESTful request dispatching. After receiving ReSTful API resource calls from the front-end, our back-end server will return the required data result about life in the cities of Australia to visualize. This kind of data is stored in the CouchDB database with MapReduce data analytics.

Flask is a micro web framework written in Python microframework because it does not require particular tools or libraries. First, developing a system with Flask is easy to start. Team members are familiar with the Python language and can quickly move around and contribute to the Flask application. Due to the flexibility and minimality of the Flask framework, we can easily modify existing systems. Compared to Django (one of the Python web frameworks), Flask has better performance because of its fewer abstraction layers, the requests, the cache, etc. Flask can be thought of as a micro framework being slightly more "low-level" than Django. Flask is also a free, open-source framework. Our application uses some URLs to get required resources across the web, and Flask is supportable.

6.2 RESTful design

The back-end server adopts a RESTful design to expose data to the client. Using the RESTful pattern, we can separate the user interface concerns from the data storage concerns, enabling the user interface's movability across multiple platforms and improving scalability by simplifying the server components.

REST is the most logical, efficient, and wide-used standard in creating APIs for Internet services. In short, it is an interface between systems using HTTP to obtain data and generate operations on data in all possible formats, such as XML and JSON, which is flexible. REST also has stateless, cacheable, uniform interface and layered system principles. Compared with SOAP, and WSDL-based kind, RESTful provides a way to design Web services with less dependence on proprietary middleware.

Considering the advantages of RESTful design, we use it in our web application to exchange data and better display specific Tweet scenarios.

7. Front-end Component

7.1 Front-end Design (React)

The front-end server is also deployed on the instance named “couch-1” based on docker and Nginx. The front-end web application is established based on ReactJS and uses a standard template provided by material UI. For better visualization of datasets and scenarios, MatrialUI, react-google-maps, Tableau JavaScript API, react-word-cloud are utilized in the web design. React-router can provide declarative routing for React web to accomplish a better page switch for different scenarios.

React.js is an open-source JavaScript library for developing user interfaces, and it can be used in single-page or mobile applications. There are many libraries currently for free use, and it is friendly for fresh users to implement web applications. Therefore, we choose to React framework to establish our front-end application.

Our application chooses Axios to make HTTP requests, which can support older browsers, and automatic JSON data transformation. It enables our application to exchange data in a RESTful way.

7.2 Visualization

The front-end web is composed of several pages. As a user, he/she can switch pages by clicking left-side buttons or directly input different routes:

1. The homepage displays the overall Google map focusing on Australia because our social media data analytics scenarios are based on Australian cities.
2. When the user clicks the Report button, he/she can download the completed report, which gives users a comprehensive understanding of our system and analysis result.
3. The front-end also has five significant pages to display the five scenarios respectively; each of them has visualisation, description, and analysis.

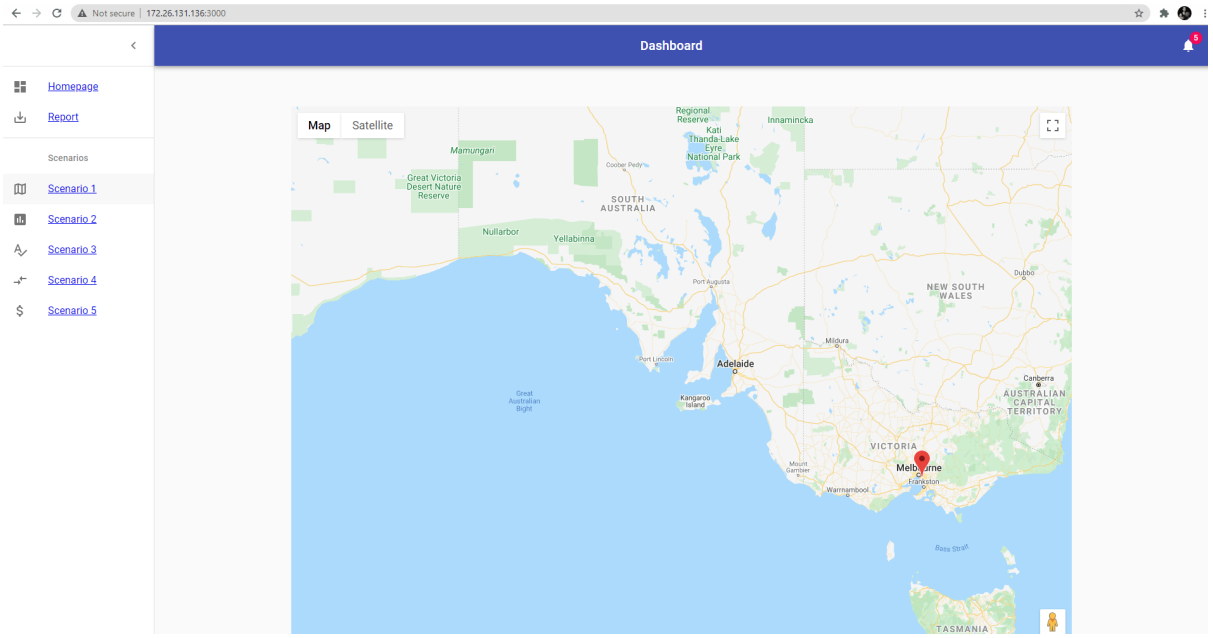


Figure 4 - Frontend web visualization example

8. Data Analysis and Scenarios Discussion

When collecting sample tweets from Australian cities by crawling, we find the language used is more casual than real daily life. It would be interesting to explore vulgar tweets and gain more understanding of Australian city life. So, we collect more tweets that include offensive words and also use other datasets from Aurin to process data analysis and find the results behind these explicit tweets.

To analyse the tweets, views constructed by MapReduce functions were called using Python client API for CouchDB. Before the visualization, a few processing steps were taken on the views, such as reorganizing data and sorting, to make it adaptable to our visualization formats. As the databases continue saving new streaming tweets, we use the Twitter data that were crawled up to 23rd May 2021 for analysis. Descriptions and analyses of five data analytics scenarios are presented below.

8.1 Scenario one: Computing the percentage of explicit tweets in each state

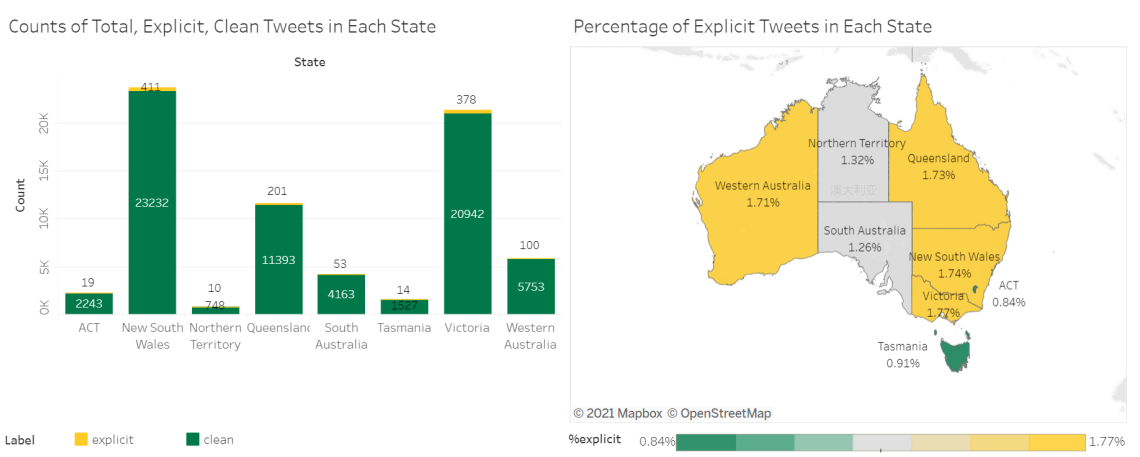


Figure 5 - Graph of Scenario one

The first scenario is computing the percentage of explicit tweets in each state. The explicit tweets represent the tweets that have vulgar words, and the clean tweets mean the tweets which do not have vulgar words. As the figures are shown above (Figure 5), the left one is a bar chart that shows the number of clean tweets and explicit tweets, respectively. From the bar chart, it infers that the number of explicit tweets for each state is commonly small. In addition, the right figure is a geomap that can suggest the percentage of explicit tweets in each state. From this graph (Figure 5), it can be seen that the state with the highest percentage is Victoria (1.77%), and the lowest one is ACT (0.84%). It can be concluded that people send tweets with swear words much less frequently than those without swear words. This shows that people rarely use swear words when communicating on the Internet.

8.2 Scenario two: Using barchart to study the top 3 commonly used vulgar words in each state

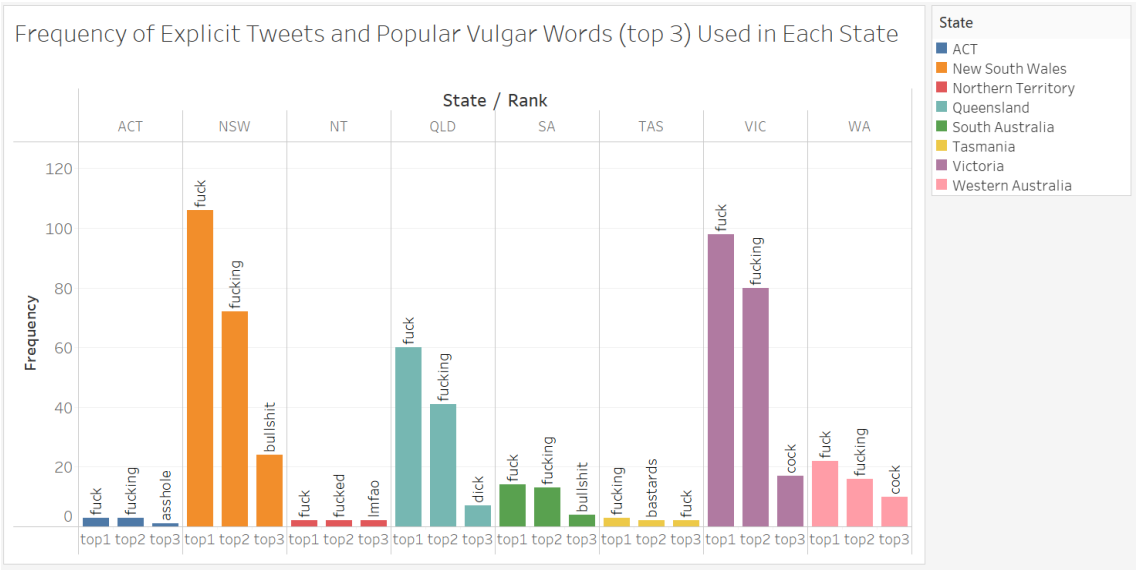


Figure 6 - Graph of Scenario two

The second scenario is to find out the top 3 commonly used vulgar words in each state and count out the frequency of these words being used. From the figure above (Figure 6), different colors represent each state. It can be easy to point out that Victoria, New South Wales, and Queensland are the three states which have higher vulgar words used frequency. Perhaps this phenomenon is related to the populations. Because these three states have larger populations than other states, then the frequency of using swear words will be relatively higher. Furthermore, from the figure (Figure 6), in each state, the top three words in the frequency of swear words have “fuck” and its derivatives “fucking”, and the rest one of the top 3 vulgar words for each state may have some differences. It explains that the word "fuck" and its derivatives have become the most popular vulgar words in the whole of Australia. For the differences between the rest of the top 3 vulgar words for each state, it may be caused by the local dialect or the age of the population being different in each state.

8.3 Scenario three: Making a word cloud of vulgar words to detect which swear words are more commonly used in the whole of Australia

Figure 7 - Graph of Scenario three

This scenario makes a word cloud to find out which swear words are more commonly used in Australia. As the figure shown above (Figure 7), this is a word cloud that consists of the vulgar words that are most widely used in Australia. The word cloud suggests that the word "fuck" and its derivatives are the most commonly used words in Australia; it also supports the phenomenon and conclusion in scenario two.

8.4 Scenario four: Comparing the word cloud of hashtags with swear words tweets to that without swear words tweets in each state

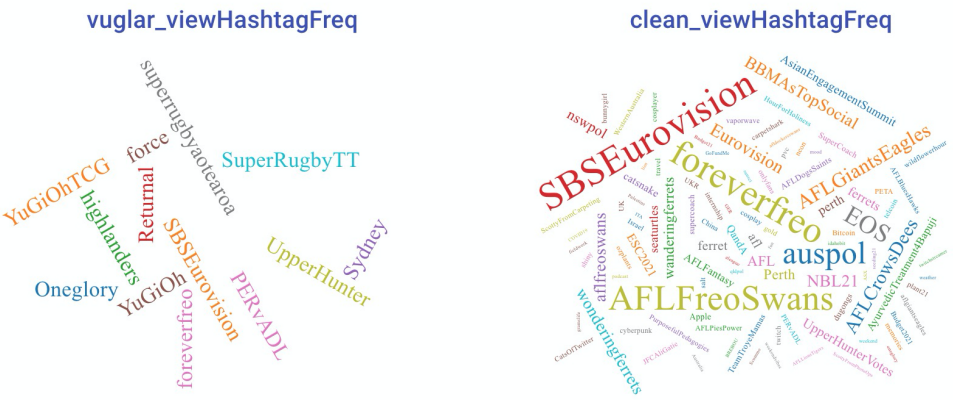
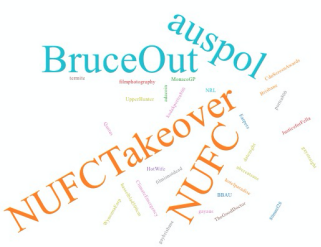


Figure 8 - vuglar vs clean Twitter's hashtags in Western Australia

vuglar_viewHashtagFreq



clean_viewHashtagFreq

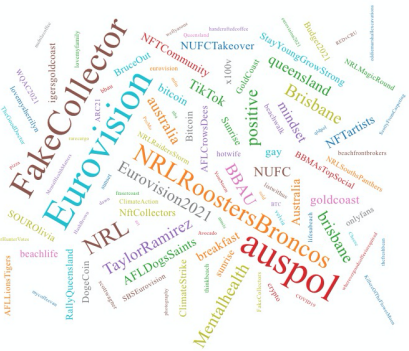


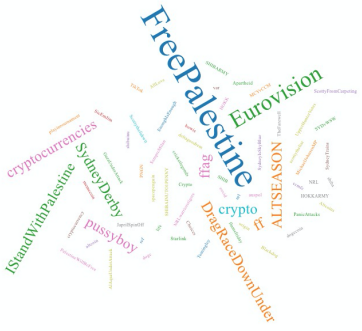
Figure 9 - vuglar vs clean Twitter's hashtags in Queensland

vuglar_viewHashtagFreq

clean_viewHashtagFreq

Figure 10 - vuglar vs clean Twitter's hashtags in Northern Territory

vuglar_viewHashtagFreq



clean_viewHashtagFreq

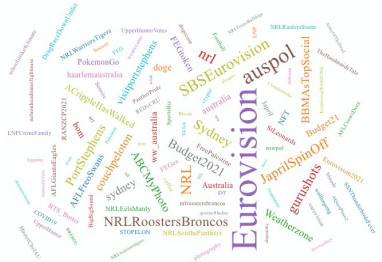


Figure 11 - vuglar vs clean Twitter's hashtags in New South Wales

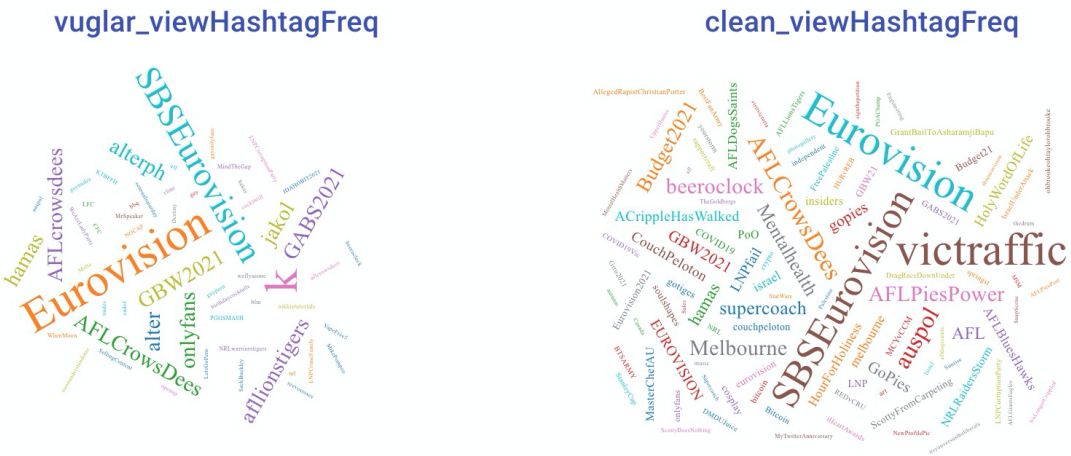


Figure 12 - vuglar vs clean Twitter's hashtags in Victoria

Figure 13 - vuglar vs clean Twitter's hashtags in Tasmania

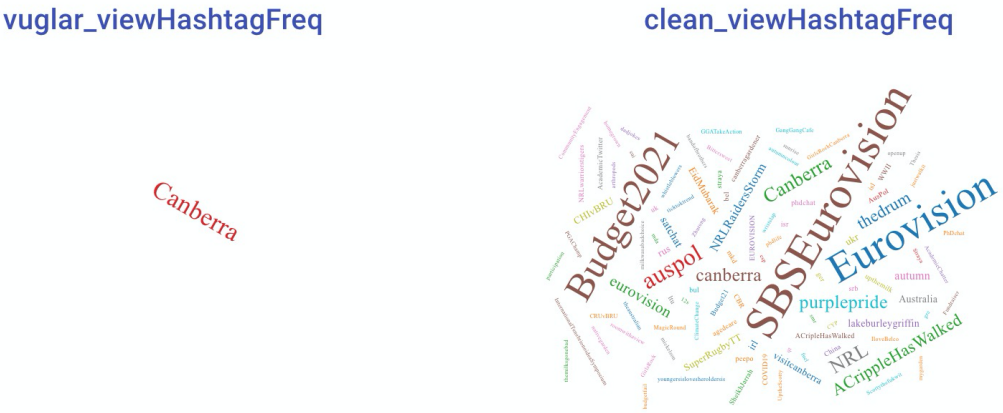


Figure 13 - vuglar vs clean Twitter's hashtags in South Australia

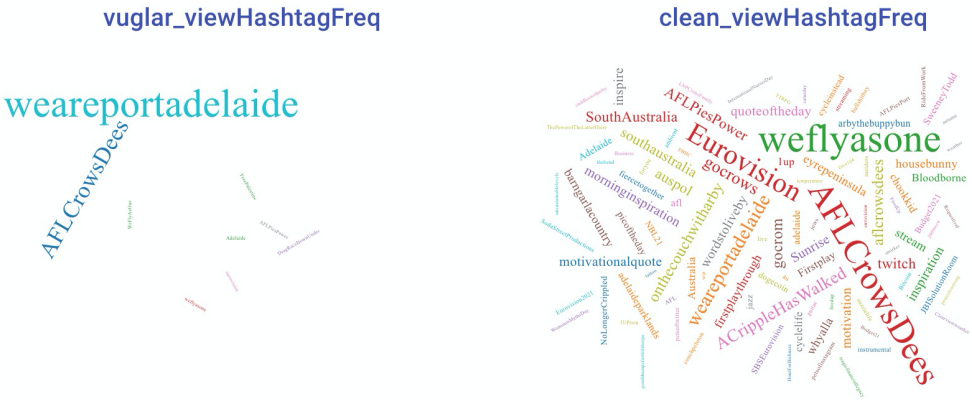


Figure 14 - vuglar vs clean Twitter's hashtags in Australian Capital Territory

This scenario talks about the word cloud of hashtags with swear word tweets and that without swear words, tweets in each state. There are two figures for each state (Figure 8-14); the left one is the word cloud of hashtags with swear words in tweets, the right one is the word cloud of hashtags without swear words in tweets. This scenario can infer the topics that may make people send vulgar words when talking about them. Furthermore, people's emotions can be inferred when they are talking about some topics with swear words; it can also show people's attitudes towards this topic. From these graphs (Figure 8-14), it can be seen that the few states have fewer hashtags with vulgar words. The reason is that the total number of tweets in these states is relatively small; it should take more time to get many tweets to solve this problem.

8.5 Scenario five: Comparing the vulgar percentage to the average of annual personal income of each state

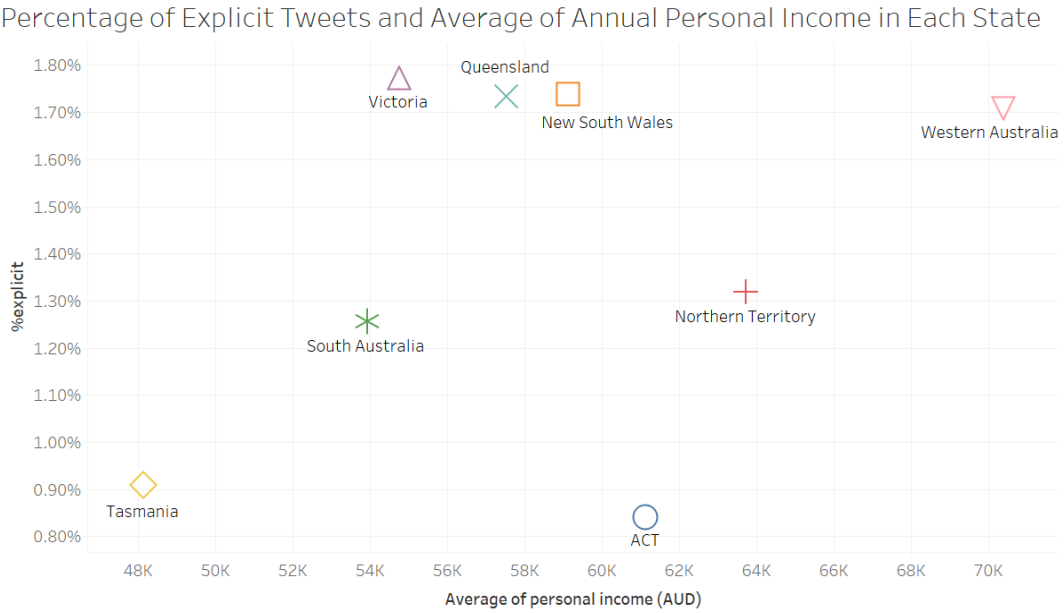


Figure 15 - Graph of Scenario five

The last scenario is about the relationship between swear word frequency and the average annual personal income. From the figure above (Figure 15), different symbols represent different states. The x-axis represents average personal income, and the y-axis represents the percentage of vulgar words. Among these states, ACT and Northern Territory have tiny numbers of tweets, and the small sample size may cause inaccurate results. Suppose the two records of ACT and Northern are eliminated. In that case, it can be seen that Tasmania has the lowest value with both vulgar percentage and average personal income; Victoria, Queensland, New South Wales, and Western Australia have the highest vulgar percentage; however, the average personal income for these four states are different. It can be predicted that the rate of tweets with swear words may increase with the increase of average personal income. When the average individual income increases to a specific value, the percentage of swear words may be stable and no longer continue to increase significantly. Additionally, the rate of tweets with swear words may be influenced by other factors. Deeper research can be done in the future.

9. Findings and Discussion

9.1 Findings and Conclusions

For our team, the biggest challenge is that this project requires us to learn multiple new knowledge in a limited time quickly. We were especially learning Ansible and Docker. Since we had not used Ansible and Docker before, we spent a lot of time in the early days learning the deployment of instances, including the syntax of yaml, resources allocation, and how to define roles. Whenever there was a problem with the yaml code, we had to spend a lot of time dealing with and redeploying.

Besides, we also spent a lot of effort in understanding the concepts of Docker image, container, network, and inventory, and how to use docker to deploy on the instances. However, we are satisfied and gained a lot of new techniques from this project.

For the data analyzing part, we find that the frequency of using swear words may have relationships with some factors like average income and different areas. Also, we find out the most commonly used swear words in other regions of Australia. However, we think the data size in this project is not big enough to make the result more accurate.

9.2 Discussion

9.2.1 Dynamically scale application

We used three instances to deploy the crawler script; the advantage for doing that is that if one or even two of the worker instances are corrupted, the remaining instances can still collect tweets into CouchDB.

9.2.2 Future improvement

In my system, we use data harvesting twitter and adopt some preprocessing data methods.
When the data becomes more extensive, we should filter data, adopt data cleaning, and form a data warehouse, which needs more professional data processing progress.

10. Working Allocation

Name	**contribution**
Xiaoyi Han	Writing Ansible scripts
Ka Hou Hong	Writing streaming crawler and search crawler Writing Dockerfile and docker-compose for crawler Writing report
Ziyuan Xu	Writing front-end and partial backend application Writing Dockerfile and docker-compose for web app Writing report
Xiaoyue Lin	Writing MapReduce functions Visualising and processing data for analysis Writing report
Xingyang Shen	Using crawler to collect tweets Analyzing the visualization result Writing report and checking grammar

11. Reference List

[1] Richardson, L., & Ruby, S. (2008). *RESTful web services*. " O'Reilly Media, Inc.".

[2] Grinberg, M. (2018). *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

[3] Apache CouchDB. (2021) *Apache CouchDB© 3.1.1 Documentation.*
https://docs.couchdb.org/en/latest/intro/index.html