THE UNIVERSITY OF MELBOURNE

COMP90024 Cluster and Cloud Computing

Assignment 2 City Analytics on the Cloud Report

Group 29

Hongwei Yin 901012

Cheng Sun 900806

Xinyi Xu 900966

Yiran Yao 1144268

Xiaotao Tan 1032950

Table of Contents

1. Introduction

From bushfires to COVID-19, 2020 is probably the most challenging year for the Prime Minister of Australia - Scott Morrison. Morrison suffered a huge loss in approval rate during the bushfire disaster at the beginning of 2020, however, the political winds have shifted within such a short period. The Newspoll published on Apr 24th [1] shows that voters overwhelmingly support his work, which made him become the most popular PM in more than a decade. During these periods, many Australians expressed their opinions for him on different social media platforms, including Twitter. The 2018 Sensis social media report suggested that 19% of Australian social media users accessed Twitter and one in three users tweet daily. Therefore, we plan to conduct a Twitter based sentiment analysis about Scott Morrison.

In this assignment, a cloud-based system is developed, including a Twitter harvesting application, a CouchDB database to store tweets and perform analytics with MapReduce, and a front-end web application for visualization. Sentiment analysis will be applied to relevant tweets, which involves classifying opinions in text into categories like “positive”, “neutral” and “negative”. With such data, it is then possible to understand which city in Victoria has the highest approval rate for Scott Morrison and how does this correlate with the income level, education level and overseas migrant population in the chosen cities, which are Melbourne, Geelong, and Bendigo.

The rest of this report is organized as follows: section 2 gives detailed description about the system architecture and design. Section 3 will discuss advantages and disadvantages of the UniMelb Research Cloud. We present our system functionalities in section 4, including four scenarios, data analysis, graphical results, and errors handling. Section 5 and 6 offers a simple user guide for testing, including system deployment and end user invocation. Our work is concluded in section 7.

2. System Design and Architecture

2.1. System Design

In this section, details of the design and architecture of the system will be discussed, including each component of the system, and why this was chosen. To support a range of analytic scenarios, the team is expected to have several instances running on the UniMelb Research Cloud along with an related CouchDB database, containing the collection of related Tweets from the harvester application. Besides, to visualize these scenarios, a front-end web application is required. The chart of the system design is shown as below (Figure 1).

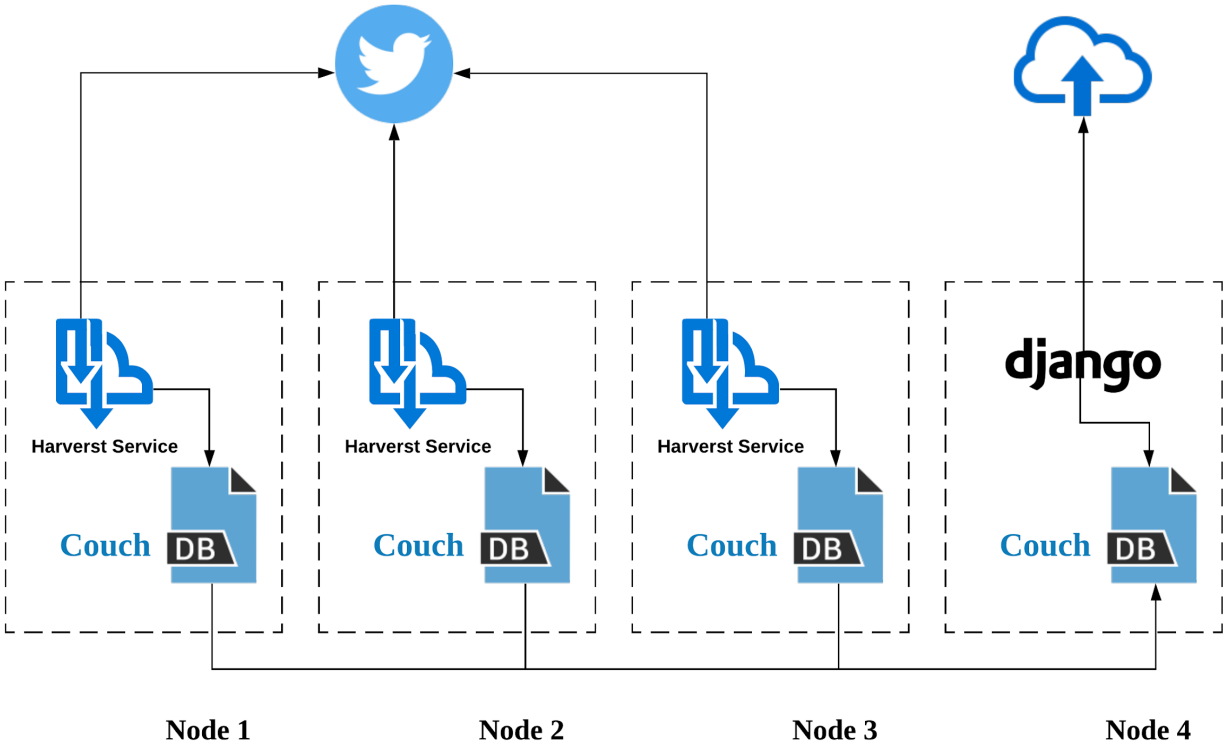

Figure 1: The diagram of the system architecture

The system occupied 4 nodes on the UniMelb Research Cloud, with 3 out of 4 nodes are used for harvest service. And the fourth node is used for RESTful service, which takes the data in CouchDB, and displayed at the frontend page of the website.

2.2. System Components

In this section, the key components of this system are discussed with a detailed analysis of benefits and deficiencies, which includes tweets harvester, CouchDB, web server and sentiment analysis, as shown in Figure 2.

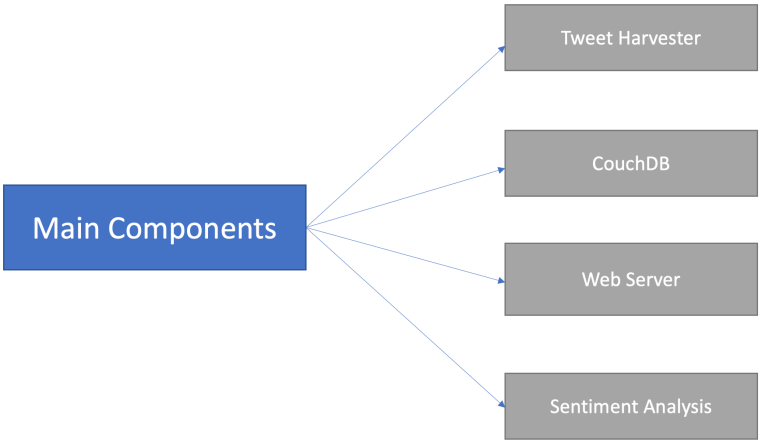

Figure 2: Main System Components

2.2.1. Tweets Harvester

Historical tweets are extracted using the Tweepy package and its search API. The API, however, is only limited to retrieve tweets posted in the past 7 days. In order to overcome this restriction, a 2-stage extraction strategy was employed. 4 sets of API keys were used in different stages of the extraction to minimise the impact of hitting rate limit and being forced to wait for the limit to replenish. The benefits of this strategy are two-fold. As the full usage of twitter native API requires subscription, the number of tweets that one account is allowed to harvest each month is extremely limited for free accounts. This is bypassed by utilising Tweepy. Secondly, the depth of extraction can be further exploited by examining the user timeline in which a tweet is posted, thus expanding the raw data volume available for analysis.

Real time tweets are streamed using the StreamListener class from the Tweepy package. The same filter criteria as harvesting historical tweets are applied, apart from date range. The streaming is divided into 3 groups based on user locations (Melbourne / Victoria / Other cities), with different API's assigned respectively. The 3 streams are run on all 3 instances that make up the database cluster. More details about how to obtain relevant tweets will be discussed in section 4.

2.2.2. CouchDB

CouchDB is chosen for data storage and processing, and it is a NoSQL database that supports MapReduce. Through map() and reduce() functions, it can process data in a parallel way and therefore provides a more effective way to deal with a large amount of data. Also, it offers data in JSON-based document format for storage, which means that the data can be stored directly, rather than being split into smaller parts and placed in different tables. As a document-oriented database, it stores raw tweets from the harvester as documents and supports easy replication and fast retrieval, which provides lots of benefits for the distributed system. After processing by the MapReduce in CouchDB, the tweets will be delivered to the virtual machine of the analyser then conduct the sentiment analysis.

2.2.3. Web Server

a) Django

Django, as a high-level Python web framework, has been selected to build the web server for this project. A RESTful framework is implemented using Django, it monitors the result in the database, and reflects the changes on the frontend display. It is capable of coping with several effective external tools, including Google Map API, Apex chart, which are used to display the data to users through an intuitive graphical interface. With its comprehensive functionalities, it would be more manageable when the tight time schedule has been taken into account. Django provides a pre-built framework leaving the component as blank, thus developers could take the advantage of its one-stop solution from template, ORM, session, authentication and so on. Meanwhile, Django has perfect online documents, which can provide great help for the problems in the project, so Django becomes the most appropriate option.

However, the Django framework also has its own disadvantages. Django framework works in its own way of defining and executing tasks. With its logical file structure, it leaves little space for developers to change its default structure. And all developers are expected to follow all the predefined rules. Therefore, the learning cost for this framework is relatively high. Another shortcoming of Django framework is it requires the time and space for server processing, due to the large number of codes.

b) Google Maps API

Google Maps API is the most powerful SDK for embedding a map into a web application. One may modify almost all visual components of the map. Customizable markers and shape layers help developers to visualize datasets from multiple sources. To access the API, one may simply add a script tag to their HTML file. However, since Google Maps API is not open source, one may have to work around some

of its rigid design. For example, the API itself does not have the “popup window” functionality. To add popup windows to the map, users may need to create an extra info window to contain the link of the popup window. Also, since the layer functionality does not take multiple layers into account, developers cannot optimize the map’s usability in various aspects.

2.2.4. Twitter Content Sentiment Analysis

Twitter offers the stream API for developers to collect the post online, and further analysis could be conducted using these data. The final solution of our project to analyze the polarity of twitter content is using Valence Aware Dictionary and sentiment Reasoner (VADER), developed by Natural Language Toolkit (NLTK), which is a lexicon and rule-baled sentiment analysis tool. It not only shows the result being positive, neutral or negative, but also reveals how positive or negative it is. With this numerical result, when developers apply the sentiment analysis, the result would provide more information that could be further looked into.

3. UniMelb Research Cloud

We have been allocated 4 servers with 8 virtual CPUs and 250 Gb of volume storage on the UniMelb Research Cloud in this assignment. We summarized several advantages and disadvantages of the UniMelb Research Cloud while working with it and they will be explained in this section.

3.1. Advantages

- Dashboard

  The research cloud provides a well-organized dashboard, and it is convenient for users to make use of its different sections, especially creating, managing and terminating instances, without writing a complex configuration script. It also provides an overview for user to keep on track of current resource usage and all VM instances.

- Free & on-demand

  Like other commercial cloud services, the research cloud offers a similar functionality, but it is free to student. This allow student to test their project anytime, without considering the cost of the service. This is a great boost for student academic performance.

- Remote control & Accessibility

  The cloud service allows students to access the service anyplace around the world, and not restricted by the local copies. To manage the specific instance, the user just needs its IP address, the private key and an SSH client. With a remote-control development environment, it provides a good platform for cooperation among several developers.

- Reliable backups

  It is easy to create backups for the volume and virtual machines on the research cloud, which is really useful since the project is involved with working on distributed systems. And it is likely that some components fail, and the system terminates, backups can help to quickly recover the system, which ensure that the team can work effectively and smoothly. The cloud is also a stable platform and the team never encountered any failure on it during the entire assignment.

- APIs

  A great range of command line, application interfaces are available in UniMelb Research Cloud. Therefore, it is able to auto deploy. Developers are able to manage instances completely through scripts, including deployment, configuration and terminating instances, etc.

- Security Groups

  A security protocol could be allocated to instances when they are created. They control the opening and closing of operating systems ports.

- Free detailed handbook offered

  UniMelb Research Cloud provides a detailed handbook on some operations, and a trouble shooting page on popular operation mistakes. A developer without any technical background could follow the instructions on handbook to access the basic services.

3.2. Disadvantages

- Limited resources to student project

  The research cloud service is not providing floating IP service to student projects. Therefore, it is required to ensure the personal computer is connected to the university's VPN if the user wants to access the instance remotely, otherwise the connection will timeout. However, the VPN is sometimes unstable and causes connection issues, which brings inconvenience to the team and affects the developing efficiency.

- Restricted user guide in the handbook

  Although we say the handbook provides a detail instruction on basic operations, the instructions for complex services are not written in the handbook. And some instruction given is complicated and frustrating for students without programming or relevant background use some complex features.

- Instability under huge demand

  It is time consuming when processing with large volumes of data and displaying the web interface of the cloud due to the limited given resources and massive enrolments of the subject.

4. System Functionalities

4.1. Data

In this subsection, details about two main data resources - AURIN and tweets will be discussed. Also, error handling such as removal of duplicate tweets, and initial data processing steps will be demonstrated.

4.1.1. AURIN Data

The analysis covers five cities of Victoria, Australia, covering Melbourne, Geelong, and Bendigo. Therefore, data related to income, education, overseas migration population for each city are generated from Australia Urban Research Infrastructure Network (AURIN) to support different scenarios analysis. The average income for 2014-15 financial year, the percentage of persons with post school qualifications postgraduate degree and the number of overseas migration in 2016 work as indicator respectively for comparison.

4.1.2. Tweets

a) Filter Criteria

Keywords: Scott Morrison, Morrison, #ScottMorrison, @ScottMorrisonMP

Language: English

Location: Victoria (Profile location within Victoria but does not mention the other 3 cities explicitly) / Melbourne / Geelong / Bendigo

Date range: 1/11/2019 - 22/5/2020

b) First Stage

Tweets are filtered with keywords in interest, namely **Scott Morrison**, **Morrison**, **#ScottMorrison**, **@ScottMorrisonMP**. Only tweets written in English (with the '**Lang'** attribute detected as **'en'**) are included in the analysis. For each tweet in the result, the **Location** attribute in the user profile is inspected to determine if the user is located in the State of Victoria. Although the geo location information would be a more precise determinant of user's location, the percentage of extracted tweets that contain such information is too low to fit the purpose of analysis, thus user's profile location is used as a backup choice.

If a user is indeed located in Victoria, the API then proceeds to probe the user's timeline for tweets posted in the set date range. Any tweets that include keywords are added to the database.

c) Second Stage

For each tweet added to the database, further investigation is conducted to check if said tweet is a retweet rather than an original post. If yes, API will look for the original tweet and attempt to extract a list of users who have retweeted. It is worth to note that if the number of retweets exceeds 100, only the first 100 retweets will be extracted. Among those retweets, the API continues to probe each individual user timeline for any posts containing keywords.

There are 2 checks involved in filtering location data. The first, also preferable method to obtain location is to inspect coordinates or the **'Place'** attribute of where a particular tweet is posted from. This information however is not always available, therefore a second check is implemented as a remedy, where the program looks into the user profiles in an attempt to discover the location entered by users. As the analysis depends greatly on geographic information, the second check is crucial in building up usable data, although at a cost of losing accuracy, as a user might neglected to change location after moving to a different State or using 'Melbourne' as a general term for residing in Victoria, even though the actual location might be outside of metro Melbourne area.

4.2. Error Handling

4.2.1. API rate limit reached

Twitter search API has a limit on how many tweets can be harvested in a 15-minute window. If the rate limit is hit before the interval lapses, the API will be forced to go into sleep mode and terminate the API connection. Two approaches are taken as compensation. The first is to leverage the **Application only Auth** instead of **Access Token Auth**, which grants a rate of 45,000 tweets per 15 minutes, 2.5 times more than the Access Token limit.

This however, is still not an once-and-for-all solution to avoid hitting the rate limit. The **wait_on_rate_limit** argument can be set to True when calling Tweepy search API, which allows the API to automatically wait for rate limits to replenish, then pick up where the search left off rather than starting over.

4.2.2. Duplicates

With the aforementioned extraction strategy that searches repeatedly, it is highly possible that the same user timeline will get tapped more than once. To eliminate the waste of rate limit and improve the efficiency of harvesting, a list of user ID's is recorded to keep track of users whose timelines have been harvested already.

Since each published tweet is assigned with an unique identifier, this ID becomes the document key when saving to the database. As soon as the API detects a tweet that meets all filtering criteria, a check will be initiated to investigate whether it has previously been saved to the desired database. Saving will proceed if no such document is found, however the tweet will be skipped if a document with the same ID is found to exist. Duplicates are thus eliminated by employing additional procedures, boosting the overall data quality and accuracy, as well as productivity of the harvester program.

4.3. Data Processing

4.3.1. Data Cleansing

Raw twitter data extracted contains a whole range of related information, which a number of them are not of interest in this analysis. The key values to preserve through the cleansing process are: unique ID, date, tweet content, user profile, geolocation, language. By cleansing the data, the database volume required to store all tweets can be reduced significantly. Yet, the loss of information makes it impossible to expand the analysis into other aspects, for instance, comparing the number of users with their own opinions toward the Prime Minister and the government against the number of users who only retweet.
The time information provided by Twitter encompasses precise minute and second, as well as timezone. As the analysis only focuses on aggregating the number of tweets published in a day, the Created_at value is pre-processed to retain only date before adding to the database.

While the user object from each tweet data is retained, not every attribute within the object is useful to the analysis. User description, URL, and entities are removed in the cleansing step.

4.3.2. Data Preprocessing

In our project, the actual content of the tweets is used for Sentiment Analysis. The retweets content is also be treated as a share of personal opinion. The key information of the twitter content for Sentiment Analysis including the words in the tweets, the emoji used in the post, and the referenced tweets. Several steps are applied to clean the text:

a) Lower the content, and remove all punctuations

b) Remove words that contain numbers, stop words

c) Move empty tokens

d) Lemmatize text content

e) Remove all the words with only one letter

4.3.3. Sentiment Analysis

With the cleaned tweet text, the noise for analysis are mostly removed. The VADER Sentiment Analyzer is used for analyzing the tweet text. The numerical result for showing how positive or negative the text are recorded.

According to the results, if the result was greater than 0.05, the text could be identified as a positive text. If it was lower than -0.05, the text could be treated as negative text. And any text with result locating in (-0.05, 0.05) zone is regarded as the user is expressing neutral opinion.

All the steps are packaged within the function called "sentiment test", it could take text as input and output the final result of the Sentiment Analysis result.

The examples of the input and output are shown as below:

```
text = 'I hate you'
sentiment_test(text)
```

```
['nagetive']
```

```
text = 'I cannot hate you'
sentiment_test(text)
```

```
['positive']
```

Figure 3: Examples of Sentiment Analysis

4.3.4. CouchDB Views

3 views are created to manipulate data by grouping them based on different attributes.

- Count of total number of tweets posted

  Map:

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

  Reduce: Inbuilt _count function

- Count of daily number of tweets posted and percentage of tweets with Positive sentiment

  Map:

```
function(doc) {
   if(doc.sentiment[0] == "positive") {
       emit(doc.created_at, 1);
```

```
    }
}
```

Reduce: Inbuilt _count function

- Total Positive sentiment ratio

  Map:

```
function(doc) {
   if(doc.sentiment[0]) {
       emit(doc.sentiment[0], 1);
   }
}
```

Reduce: Inbuilt _count function

4.4. Scenario Analysis

This study is intended to investigate the change of how people react with the news associated with Scott Morrison. Sentiment analysis is done using the Python Natural Language Toolkit. Each text content is classified to have a positive, negative or neutral sentiment, indicating the user's attitude towards Scott Morrison and major decisions and actions initiated by the Federal government. The approval rate is defined as the percentage of positive tweets among all tweets related to the Prime Minister.

The entire dataset is grouped first by user location, and subsequently by creation date. A daily count is obtained for the number of tweets posted in a particular day, along with the percentage of tweets in each sentiment class. Total count of tweets, average income information, percentage of migrations from overseas, and education level are taken into consideration as different scenarios. Due to the fact that cities in regional Victoria are less populated with fewer active Twitter users than major cities, data obtained from those areas is unlikely to produce meaningful and valuable results. As such, this study only considers three cities in Victoria, namely Melbourne, Geelong and Bendigo. Those users who locate in other areas other than the 3 aforementioned cities are treated as a collective that represent regional Victoria. Total count of tweets is restricted by the API of Twitter, and other factors are location properties of each city.

4.4.1. Total Count of Tweets

Total count of Tweets is the first factor that might affect approval rate. With more people expressing their personal opinion, this study could extract more information from them, thus a more accurate result of approval rate.

Nowadays, people are paying extra attention to protect themselves from private information leaking. They are unwilling to be exposed and keep their location information as confidential information. As a

result of that, only a small proportion of data collected from Tweets contains location information. It would be difficult to map the tweet content to a relatively accurate city.

The time period this project selected is from November 2019 to May 2020. During this period, the number of tweets is counted with city information associated and calculate the approval rate accordingly.

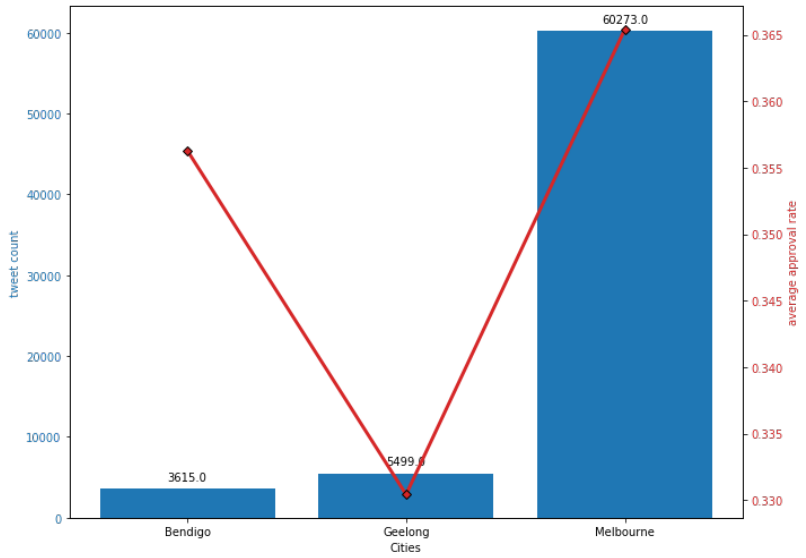

Figure 4: Total Count of Tweets VS Approval Rate

From the result shown in Figure 4, the count of tweets varies from around 4,000 (Bendigo) to over 60,000 (Melbourne). And the approval rate changes from 33% to 36.5%.

As the approval rate is an indicator of people's reaction to the news or policies associated with Scott Morrison, with less total count, this indicator could only be referenced as a general indicator. At Melbourne, with over 60K tweets, we are able to conclude the approval could reflect the opinion of populations, and the approval rate is more objective and fairer, compared to the result from Bendigo and Geelong.

4.4.2. Income Level

Does the attitude towards Scott Morrison, e.g. people support him more or less in wealthy or poor areas?

In this study, a city is defined as groups of different people with several properties. Income is one of the levels of these properties. The average income and approval rate at different cities are shown as below (Figure 5).

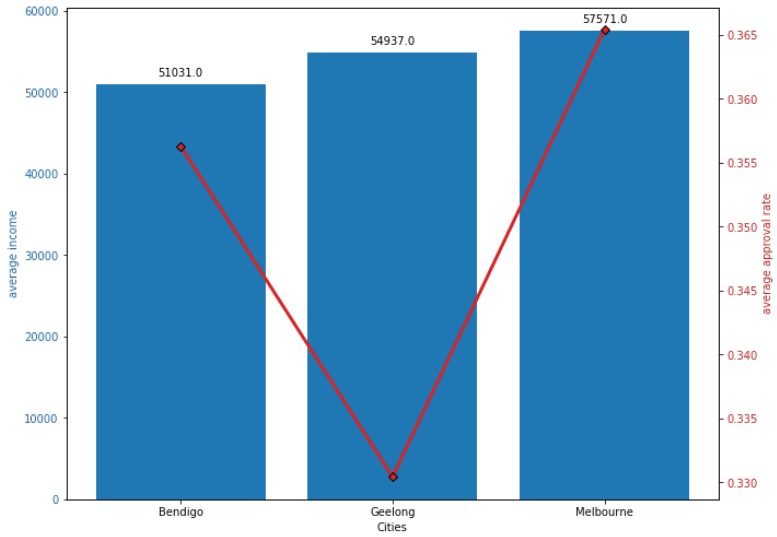

Figure 5: Income VS Approval Rate

From the chart above income and periodical average approval rate shows no obvious relationship. Melbourne is both high at the average income and approval rate. Bendigo and Geelong, which is a rural area, show less average income compared to Melbourne, and their approval rate is low as well. From this, it could conclude that people living in the metropolitan area have a higher approval rate to Scott Morrison.

4.4.3. Overseas Migration Level

For cities with higher number of overseas immigrants, will Scott Morrison have more supporters?

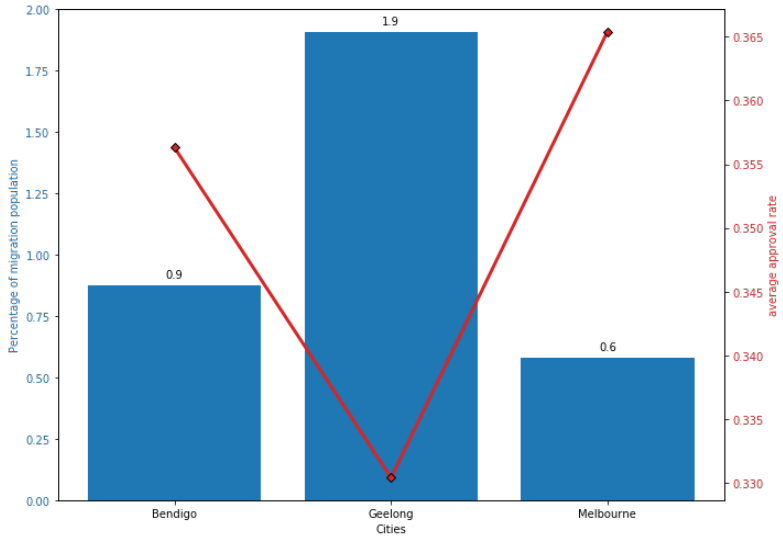

Figure 6: Percentage of Migration Population VS Approval Rate

Bendigo and Geelong, as rural areas, have higher migration levels, which could be explained as that people are easier to be granted a Permanent Resident Visa. And higher migration percentages of the whole population usually affect how oversea residents think about current government policies. From the result above, the percentage of migration population and periodical average approval rate shows inverse relationship, which indicates the population who was born in Australia expresses higher approval rate.

4.4.4. Education Level

Is there a correlation between responding positively to Scott Morrison and the education level of citizens across the cities in Victoria?

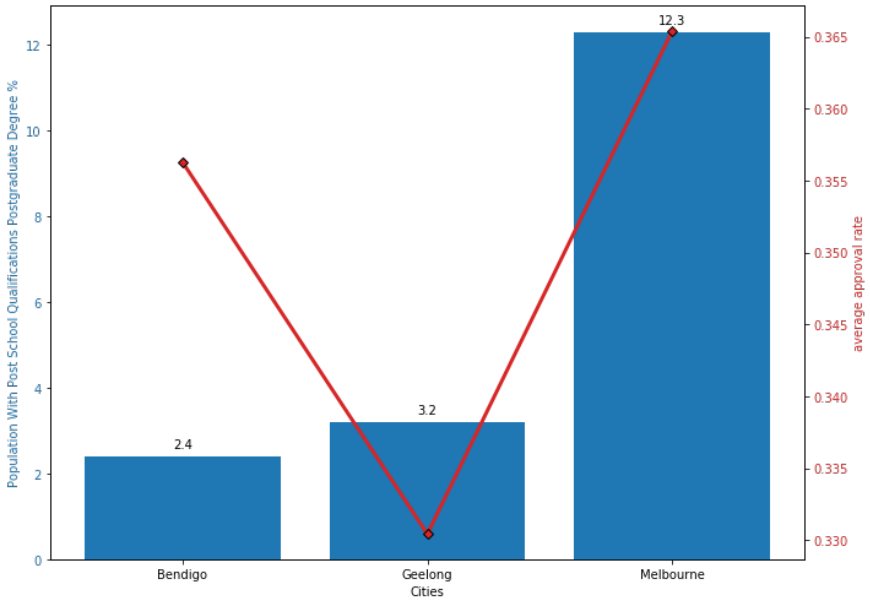

Figure 7: Education level VS Approval Rate

In this study, the percentage of people with post school qualifications and postgraduate degrees has been selected as the indicator of educational level among the area. People living in Melbourne show the highest percentage of people with postgraduate degrees, and this is understandable as Melbourne has more working opportunities.

As for the relationship between the educational level and average approval rate, the rural areas still show less approval rate.

5. System Deployment

The main focus of this project is to automate each component as much as possible. To achieve this goal, scripts are used to build up the instances of UniMelb Research cloud, and the process of deploying CouchDB, Twitter harvesting, and Django Framework are automatically conducted.

4 instances are created with an equal number of CPUs, size of RAMs, and storage assigned. 3 of which are set up as a CouchDB cluster, and the 4th instance is configured to be the web server that the application will eventually be running on. Among the 3 database cluster nodes, one node is elected to be the manager. Ansible and shell scripts are used throughout the entire deployment process, which encompasses the following steps:

1) Create instances

The role for creating instances is made up of the following 5 sub-tasks:

```
- include_tasks: instance_prep.yml
- include_tasks: images.yml
- include_tasks: mount_volume.yml
- include_tasks: security_groups.yml
- include_tasks: instance.yml
```

Running Ansible playbook:

```
. ./openrc.sh; ansible-playbook --private-key=~/.ssh/team29.pem --ask-become-pass
launch_instance.yml -vv
```

- Configuration variables

  Image: NeCTAR Ubuntu 18.04 LTS (Bionic) amd64 (with Docker)

  Security groups: Apart from the standard security groups that allow inbound communication through port 22 and 80, an additional security group is added to all instances, opening up port 5984 to allow reading from and writing to CouchDB databases. Moreover, since CouchDB uses Erlang-native clustering functionality to achieve a clustered installation, port 4369 is required to be open, enabling all nodes in a cluster to speak to each other.

- Install necessary packages

  Core tool and dependency to create instance on the Research Cloud are **pip** and **openstacksdk**.

- Create volumes and security groups

```
name: create volumes
os_volume:
display_name: '{{ item.vol_name }}'
size: '{{ item.vol_size }}'
availability_zone: '{{ availability_zone }}'
wait: yes
timeout: 600
state: present
loop: '{{ volumes }}'
register: os_vol
```

4 volumes are created to be attached to each instance. **vol_name**, **vol_size** and **availability_zone** are variables stored in a separate file.

```
name: create security group
```

```
os_security_group:
name: '{{ item.name }}'
description: '{{ item.description }}'
state: present
loop: '{{ security_groups }}'

name: configure rules for each security group
os_security_group_rule:
security_group: '{{ item.name }}'
protocol: '{{ item.protocol }}'
port_range_min: '{{ item.port_range_min }}'
port_range_max: '{{ item.port_range_max }}'
remote_ip_prefix: '{{ item.remote_ip_prefix }}'
state: present
loop: '{{ security_groups }}'
```

Similarly, variables required to create security groups are stored separately.

- Launch instances

```
name: create instance
os_server:
name: '{{ item.name }}'
image: '{{ instance_image }}'
key_name: '{{ instance_key_name }}'
flavor: '{{ instance_flavor }}'
availability_zone: '{{ availability_zone }}'
security_groups: '{{ sg_name }}'
volumes: '{{ item.volume_names }}'
auto_floating_ip: yes
wait: yes
timeout: 600
state: present
loop: '{{ instances }}'
register: os_instance
```

Ansible output:

```
PLAY RECAP *********************************************************************
localhost                  :
ok=22   changed=5    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

2) Set up instance environment

A number of libraries and tools are to be installed across all instances. In terms of the 4th instance - webserver, on the other hand, few more dependencies are necessary to successfully deploy and run the Django application.

```
- hosts: all
 gather_facts: true
 roles:
   - role: setup-env-all

- hosts: webserver
 gather_facts: true
 roles:
   - role: setup-env-webserver
```

Running Ansible playbook:

```
. ./openrc.sh; ansible-playbook -i ./inventory/hosts --private-key=~/.ssh/team29.pem
--ask-become-pass setup_env.yml -v
```

- Set up instance proxies on all 4 instances which enable them to communicate with the outside world

```
- name: create
 become: yes
 file:
   path: /etc/environment
   state: touch
   mode: 0777

- name: add ins env proxy
 become: yes
 blockinfile:
   path: /etc/environment
   block: |
     HTTP_PROXY=http://wwwproxy.unimelb.edu.au:8000/
     HTTPS_PROXY=http://wwwproxy.unimelb.edu.au:8000/
     http_proxy=http://wwwproxy.unimelb.edu.au:8000/
     https_proxy=http://wwwproxy.unimelb.edu.au:8000/
no_proxy=localhost,127.0.0.1,localaddress,172.16.0.0/12,.melbourne.rc.nectar.o
rg.au,.storage.unimelb.edu.au,.cloud.unimelb.edu.au

- name: reboot
 become: yes
 reboot:
   reboot_timeout: 3000
```

- Install dependencies appropriate for the purpose of each instance
  Common tools for all instances:

```
apt:
   name: ['vim', 'python3-pip', 'python3-setuptools']

pip:
   name: ['jsondiff', 'tweepy', 'couchdb', 'nltk']
```

  Applicable to webserver instance only:

```
apt:
   name: ['python3-dev', 'libpq-dev', 'nginx', 'curl', 'git']

pip:
   name: ['virtualenv', 'django', 'whitenoise', 'djangorestframework',
'python-dateutil', 'django-jquery', 'django-extensions']
```

Ansible output:

```
PLAY RECAP *********************************************************************
ins1                       :
ok=12   changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ins2                       :
ok=12   changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ins3                       :
ok=12   changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ins4                       :
ok=15   changed=15   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

3) **Install Docker**

Docker is to be installed on all instances. This step consists of 3 sub-tasks:

- **Create volume directory and mount volumes**
  Data generated and used by Docker containers needs some volumes to reside in. **Source** path and **Destination** path variables are stored in a separate file.

- **Install Docker**

  The steps to install Docker strictly follows the recommended process shown on its official website. Prerequisite steps include installing dependencies, adding Docker GPG key, adding Docker apt repository, and finally install Docker, along with an optional component - Docker Compose.

- **Configure Docker to use proxies**
  Proxy file requires to be stored in a dedicated path (`/etc/systemd/system/docker.service.d`). In order to configure proxies, the directory in which the config file is located needs first to be created, before copying the pre-written config file to the directory.

- **Flush changes and restart Docker service for the proxies to take effect**

```
- name: flush changes & restart Docker service
become: yes
 systemd:
   state: restarted
   daemon_reload: yes
   name: docker
```

4) **Setup CouchDB cluster**

Instances 1 to 3 are designed to form a CouchDB cluster. Shell scripts are written for each node and copied to respective instances.

- Shell scripts to create Docker container with CouchDB image for all nodes
  Published ports are chosen as 5984, 4369, 9100-9200 across all nodes.

```
echo "== Get CouchDB =="
docker pull ibmcom/couchdb3:${VERSION}

echo "== Start container Master==="
if [ ! -z $(docker ps --all --filter "name=couchdb${node}" --quiet) ]
   then
       docker stop $(docker ps --all --filter "name=couchdb${node}" --quiet)
       docker rm $(docker ps --all --filter "name=couchdb${node}" --quiet)
fi

docker create\
  --name couchdb${node}\
  -p 5984:5984\
  -p 4369:4369
  -p 9100-9200:9100-9200\
  --env COUCHDB_USER=${user}\
  --env COUCHDB_PASSWORD=${pass}\
  --env COUCHDB_SECRET=${cookie}\
  --env ERL_FLAGS="-setcookie \"${cookie}\" -name \"couchdb@${node}\""\
  ibmcom/couchdb3:${VERSION}
```

```
declare -a conts=(`docker ps --all | grep couchdb | cut -f1 -d' ' | xargs -n3
-d'\n'`)
for cont in "${conts[@]}"; do docker start ${cont}; done
```

- Add worker nodes to the cluster and finish setup on the master node

```
echo "== Add nodes ==="
curl -X POST -H "Content-Type: application/json"
http://${user}:${pass}@172.26.133.229:5984/_cluster_setup  -d
'{"action":"add_node", "host":"172.26.133.240", "username":"admin",
"password":"admin", "port":"5984"}'
curl -X POST -H "Content-Type: application/json"
http://${user}:${pass}@172.26.133.229:5984/_cluster_setup  -d
'{"action":"add_node", "host":"172.26.134.32", "username":"admin",
"password":"admin", "port":"5984"}'

# THis empty request is to avoid an error message when finishing the cluster
setup
curl -XGET "http://${user}:${pass}@172.26.133.229:5984/"

echo "== Finish cluster =="
curl -X POST -H "Content-Type: application/json"
http://${user}:${pass}@172.26.133.229:5984/_cluster_setup -d '{"action":
"finish_cluster"}'
```

- Output from checking the active nodes in the cluster:
  curl http://admin:admin@172.26.133.229:5984/_membership | python -m json.tool

```
{
   "all_nodes": [
       "couchdb@172.26.133.229",
       "couchdb@172.26.133.240",
       "couchdb@172.26.134.32"
   ],
   "cluster_nodes": [
       "couchdb@172.26.133.229",
       "couchdb@172.26.133.240",
       "couchdb@172.26.134.32"
   ]
}
```

5) Start applications

A virtual environment has been configured already for Django backend with all of the dependencies needed to run the server without interruption.

- Prepare files locally before copying to remote
  Core web server files need to exist on the instance for the application to successfully run. Due to the VPN provided by the University is highly unstable, all attempts to transfer the files using scp or sftp mechanisms were failed, so was cloning from GitHub repository. Thus, the entire directory is compressed into an archive, and split into smaller chunks, each of 10 MB in size, before copying over to remote to accommodate the inadequate connection speed.

- The nignx is configured to ensure the website is launching.

6. User Guide

1) Home Page:

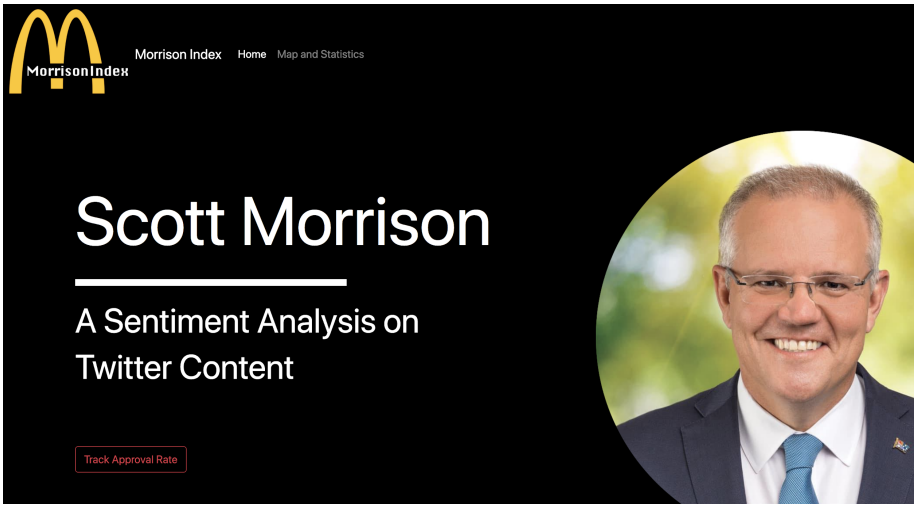

Figure 8: Home Page

Since the home page is more of a ritual than a functionality, it only contains a short description of the web application, a photo of Mr. Scott Morrison, and a navigation bar. By clicking the "Map and Statistics" button on the navigation bar, users may be directed to the map page.

2) Map Page:

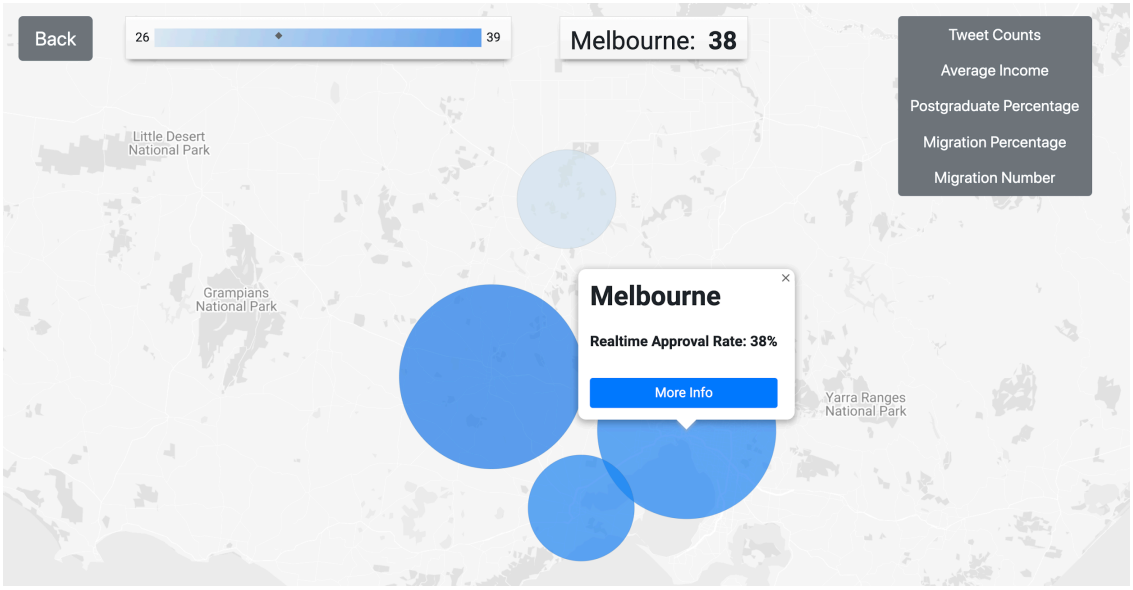

Figure 9: Map Page

In this page, the team has utilized several ways to customize result datasets (based on data gathered from Twitter API and AURIN; calculated using UniMelb Research Cloud) on the map. To increase visibility, many features of the Google map such as landmarks and labels were deleted. Four circular markers representing four of the most populous Victorian cities are

displayed. The map uses information in the form of GeoJSON to define the center of circles. The reason only four have been chosen is that other city-level regions do not have sufficient data. The whole Victoria would have been covered if more tweets were harvested. The circle size increases with the total amount of tweets retrieved. Therefore, populous regions with more tweets are represented as the larger circles on the map, indicating that they possess bigger datasets and thus offer more reliable results. The circles are also drawn with the color intensity relative to the Prime Minister's approval rate. Theoretically speaking, the bluer the circle, the more pro Morrison that region is. The legend above informs users of the current maximum and minimum approval rate. One may select other categories of data other than tweet count from the vertical menu to compare them with the approval rates.

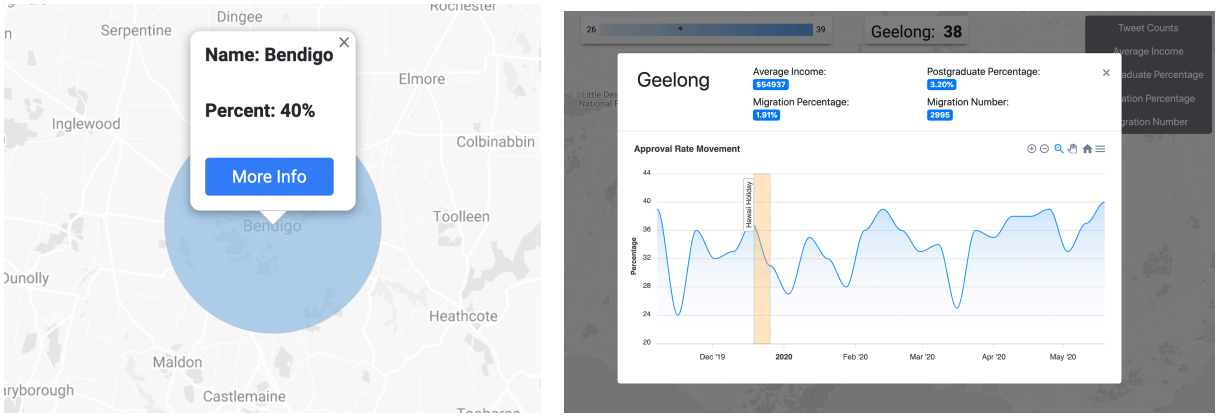

Figure 10: Detailed Statistics Popup Window

One can hover over a city circle to view city-specific information in the data box. An info-box will also automatically pop-up. By clicking the “More Info” button, a more informative pop-up window will open. The information window contains the city’s historical trend of approval rate; financial, educational and demographic statistics are also displayed in the header. The team utilized ApexChart API to draw the Approval Rate Movement. Since the daily fluctuation of the result is very high, the current line chart is drawn using the average approval rate per week.

7. Conclusion

In this report, the structure and architecture of the system which supports city analytics on the cloud are demonstrated. Besides, three relevant scenarios are being analyzed and a front-web application is built for visualizing. We find out that the approval rate of Scott Morrison is negatively correlated with the number of overseas immigrants and there is no clear relationship between the approval rate and the income or education level. As discussed in the analysis section, the approval rate calculating from rural areas can only be interpreted as a general guide, due to the small number of samples. If more samples can be gathered from these areas, a clearer relationship could be investigated.

In terms of future work and improvements: It can notice that the approval rate is time-sensitive. If all relevant tweets data can be obtained, it is able to expand the system to display the real-time approval rate across all cities of Australia, which helps the government to understand how people respond to their policies. Also, our analysis only used tweets in the past six months, and another future improvement to our work should focus on to access the long-term relationship between approval rate and chosen scenarios. This could be done by gathering more relevant data across a longer period of time, perhaps using the premium developer account.

8. References

[1] Theaustralian.com.au. 2020. *Latest Newspoll 2019 | The Australian*. [online] Available at: <https://www.theaustralian.com.au/nation/newspoll> [Accessed 25 May 2020].

[2] Mika V. Mäntylä, Daniel Graziotin, Miikka Kuutila, *The evolution of sentiment analysis—A review of research topics, venues, and top cited papers*, Computer Science Review, Volume 27, February 2018, Pages 16-32, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2017.10.002.

Link to the source code: https://github.com/gusyi/COMP90024_Team29
Link to deployment & website demo: https://youtu.be/3JJsadUnpBY