THE UNIVERSITY OF MELBOURNE

COMP90024 - Cluster and Cloud Computing

Project 2020

Team 53

Sania Khan (1045290)

Udit Goel (1042890)

Gaurang Sharma (1041953)

Kanav Sood (1057606)

Jack Crellin (1168062)

Table of Contents

I. Introduction

Australia has far been known as one of the world's most adaptable countries to work and live. The country has always managed to maintain a respectable position in global employment index over the years. The spike in the average number of people coming to study or work in Australia every year from around the world bears testimony to the fact that how lured they are by the employment status of this Commonwealth country as a whole.

Melbourne, situated in the southern part of Australia, is one of the world's highest rated city in terms of living and working. The city has always been praised for its beautiful weather, scenic beauty, finest beaches, top-class educational institutions and the most successful firms from all over the world. Not only Australians, but people from different countries never lose out an opportunity to work in Melbourne. Promoting itself as a hub of multicultural communities, Melbourne has never shied away from welcoming people to live or work in the city.

Apparently, not many people have been fortunate enough to see the fruitful results once they dive into the pit. On one hand, where Australia enjoys the status of being a stable economy with a good employment rate, a good amount of people has not been able to achieve the mental satisfaction, which they were sure of getting when they first thought of applying to the Australian job market. The much talked about global economic recession of 2008 was somehow avoided by Australia but, hiding behind the curtains, unemployment has since taken a toll on some industries, the fact to which the general public has never been made aware of. A small section of the people who have been a part of the gloominess, have been vocal about the issues through various mediums like newspapers, magazines, social media and television.

Twitter, one of the most used social media across the world, has always served as a great platform for all sections of people to opine their thoughts and views. The microblogging service, since its advent, has seen a great rise in the number of people posting photos, videos, and expressing their thoughts on any topic. Due to its free public API, the academic community from around the world have always been interested in drawing meaningful inferences from the tweets related to their respective studies. Several insights can be generated by scrutinizing the tweets of the general public, thereby integrating their thoughts and viewpoints.

In this project, a study has been conducted for the people residing in and around Melbourne, based on the content of their posts on Twitter. The study aims at interpreting the tweets of the target location and exploring the proximity of the tweets based on their sentiments. The major objective lies in exploring the people's views on unemployment in Melbourne, and in Australia as a whole, and later integrating their sentiments regarding unemployment. This project wishes to dig the hidden aspects associated to the Australian job market, and relate the inferences drawn from the people's views on unemployment with the actual rate of unemployment in Melbourne. The sentiments of the people are examined to give an overview of how they have been affected by unemployment, thereby providing an estimate of the unemployment rate. To verify the results of the study, a dataset from AURIN would be used which would ultimately give an idea about the accuracy of the study that has been conducted.
Taking note of these underlying specifications, the project leverages a huge amount of textual information from Twitter, along with the Melbourne Research Cloud, which

provides similar functionality to commercial cloud providers such as Amazon Web Services. Using additional tools like Docker, Ansible, CouchDB and Python, the team for this project has been able to build a simple software application which analyses the twitter data to extract information about unemployment and infer its relationship with the augmented data from AURIN.

II. System Architecture

The architecture is simple as shown in Figure (XX) deployed. The system was deployed on the NeCTAR Research Cloud, with three 'uom.mse.1c4g' instances running Ubuntu 18.04 (with Docker).

One CouchDB instance contains database 'twitterdata','twitter_time_series', location', 'aurin'. The second instance is the replication of the data of the first instance. with the third acting as a web server and delivering the front-end using Flask. Each of the two instances running CouchDB has attached a 70 GB volume attached, web server instance has 30GB of volume attached.

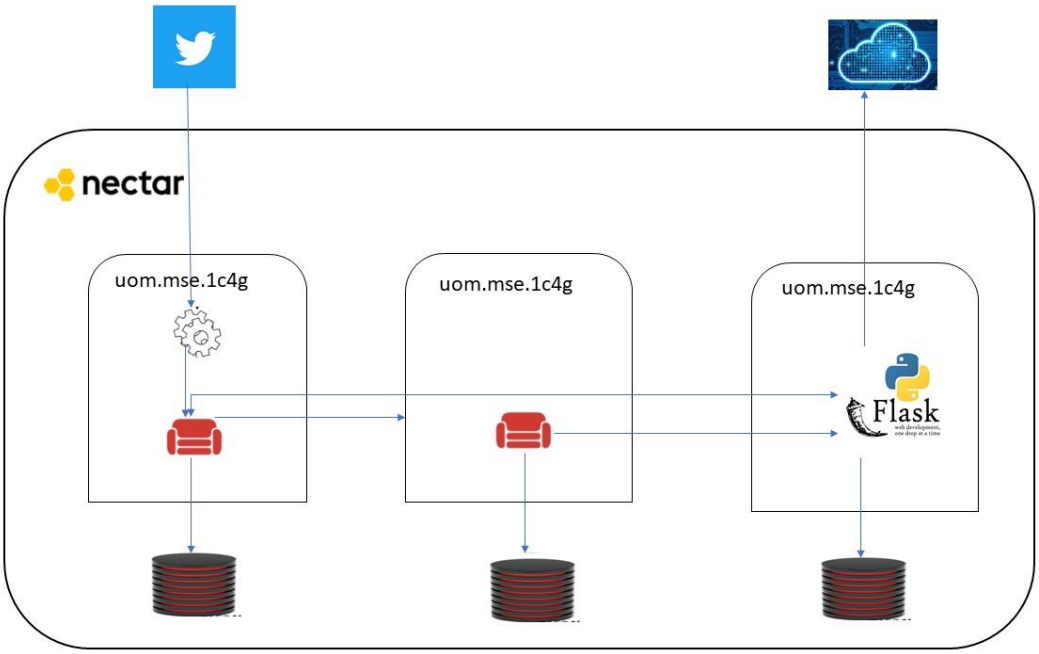


Figure 1 : System Architecture

The system architecture in Figure 1 includes CouchDB running on two instances where one is considered as a prime location for tweet harvesting and other is used to keep a copy of the documents. Third instance is for hosting the web server that displays the various analytics performed in CouchDB through views and displaying it on the UI. Sharing of the data between the CouchDB views to the web app is done By REST API calls. Making such calls enables the data to be changed at the client side without changing any code at the

server side. Thus, ROA calls are the most suitable to use in light weight approach, easy to implement and agility of the design. While SOA is service based approach with a more mature tool thus can be changed over time but this is not needed in the project which we have implemented. The ROA architecture enables the system to run in RESTful web services. Changes can be made in a manner we want the data to look at client side without affecting the data in the volumes stored at CouchDB server. Requests are made using HTTP request GET to display the data stored in views made by map reduce feature of CouchDB.

One instance is set up as a backup to the data giving the fault tolerant system thus, we can call to two instances with CouchDB set up. By default, the shards are n= 3 and q=2 are set up for the database having 3 shards and 2 replicas.

In order to collect the tweets, the harvester is set up which first collects the through Stream API by passing all members keys thus running a process for 25 minutes to process the data every time. Search API is used to collect the tweets from the user timelines and the historic data is collected from the UniMelb Research Cloud. All the tweets are then stored in the JSON files which are passes through the Condenser. In condenser, the tweets are restructured where the ID is set as the document ID for the database. Tweet has fields like text, created at, sentiment score attached to it.

While storing the retweets filed is not considered and is trimmed at Condenser level thus ensuring no display in the from end application.

Fault tolerant:

Currently our architecture is developed so that it can be fault tolerant and be available around the clock. We have been taking backups of our database at regular intervals to prevent the crashing of the app in case of any deployment issues. Also snapshots of the VM's are taken periodically to keep our data backed up in case of any issues.

Single Point of failure:

There are no single points of failure in our current application apart from Nectar.

III. Deployment

Ansible Playbook

Ansible is an open-source software configuration management, provisioning, and application-deployment tool. In this project it is used to automate the creation of instances, attach volumes, configure basic libraries for the application, run the harvesters, install CouchDB and run the web app.

We choose to create multiple ansible files for the deployment as in case few changes made in the system requirements make only run the ansible scripts required and not unnecessarily run the entire process again.

a. nectar playbook

   The nectar playbook creates the instances, volumes, security groups and common dependencies.

b. set-env playbook

   The set-env attaches the volume to the container and mounts the volume, builds the docker image, runs the docker compose, installs the CouchDB on required instances deploys the harvester.

c. web-app playbook

   It installs basic dependencies to host the flask server in Python. Flask is a web framework for Python, used in building web application, which is hosted on the web server instance, managing HTTP requests, and displaying various scenarios for our story along with charts, graphs etc.

Docker

Docker is used in the project to provide application to be packaged inside the container with all the dependencies associated with it thus, making it independent of the operating system to be deployed on. This ensures more efficient and less overhead of resources for the underlying system.

Docker Compose is a tool for running and defining and multiple containers with Docker applications. The **docker-compose -up** single command makes the YAML file to configure our application's services. We create and start all the services from our configuration. Compose use prevents every volume used by our services and it is always easier to make changes by reusing the existing containers. This can be used to configured for different environment or to the different users. This makes the rerunning of the existing services by other developers as it comes with web service APIs, databases, caches, queues etc.

IV. Twitter Harvester Architecture

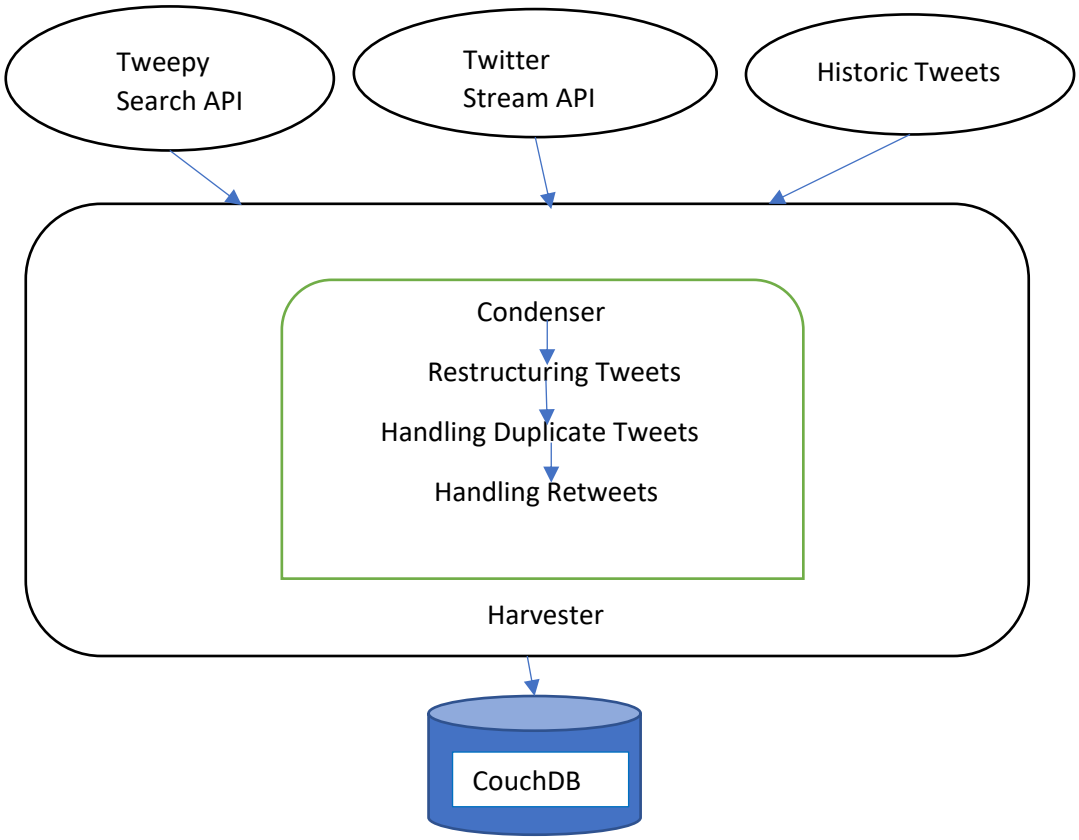


Figure 2 : Twitter Harvester

Tweets data set is collected using the three methods using Twitter Stream API , Twitter Search API and Historic tweets from UniMelb Research Cloud.

Twitter Stream API pushes the session to persistent state thus allowing a lot more real time data to flow in than the REST APIs. Stream listener call method on_data fetched the tweets according to the message type. The full text of the tweet is fetched by mentioning tweet_mode='extended' while streaming.

Pagination through user timelines and lists with requires lot of loops so to make it easy twitter uses Cursor object.

Historic data has been collected from university of Melbourne Research Cloud where data from 2016, 2017,2018 are fetched to obtain better results over the past few years.

The stream data fetches the tweet from Melbourne area with the latitude and longitude range. The search data fetched the data from the from Melbourne area with the range of 300 km to consider scenarios for various cities near Melbourne.

The flow of the harvester is designed in such a way that the tweets are cleaned and only desired field such as ID, text, created at, coordinates and sentiment score is used.

V. Augmented Data Sets

Apart from the tweets that have been loaded onto CouchDB in huge amounts, two other sets of data have been used in the project to map the results of the research with the actual scenario. The two subsections below give a detailed explanation about the datasets used and the source from where they have been extracted.

AURIN Dataset

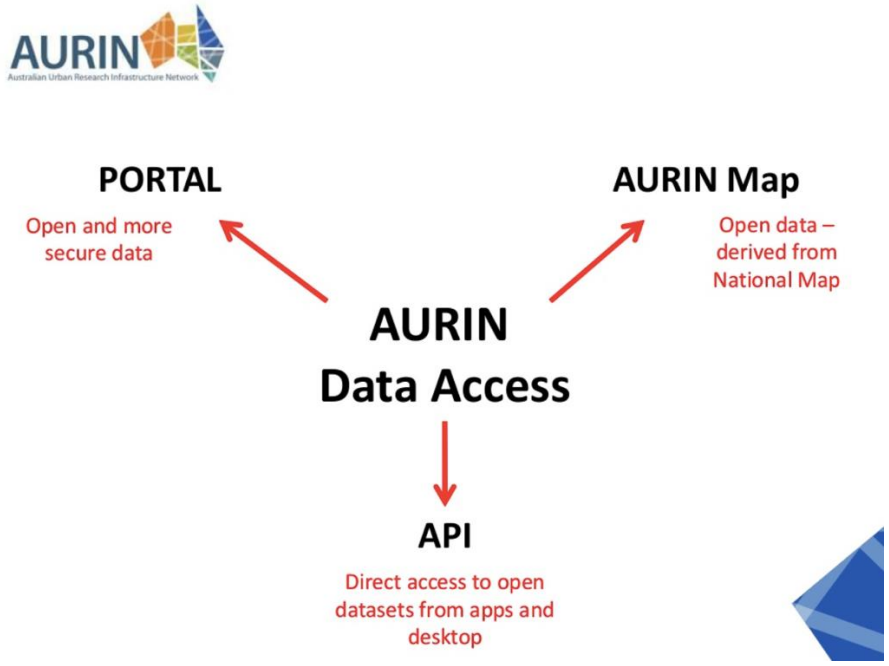


Figure 3 : AURIN Data Access

Australian Urban Research Infrastructure Network, or more commonly known as AURIN, is a platform for a wide community of researchers, government officials and industry partners, that serves as a medium to provide the data and tools to various organizations and/or prospective individuals. AURIN is like a warehouse that stores immensely huge amounts of data from Australia related to various subjects and themes, which gets augmented at a good rate on a daily basis.

For this project, two similar datasets from different time periods have been included in our study from AURIN, which provide us with a numerical information about the unemployment status in Melbourne and the surrounding cities. "*VCGLR – Population Density and Gaming Expenditure(LGA)"* datasets for 2016-2017 and 2017-2018 have been used for the analysis of our project. These data sets include the population and EGM

expenditure breakdowns by Local Government Area(LGA) and demographic statistics, labour statistics, and Socio-economic indexes for areas(SEIFA) LGA score and ranking per LGA. In addition to these parameters, 'Unemployment rate' and the 'the number of people unemployed' remain the most important ones as they cater to the requirements of the information needed to augment the study related to this project. Figure below gives the various factors present in the database being used:

Dataset Attributes (20)

Attribute	Name	Type
Adult Population 2017	adult_population_2017	Integer
Adults per Venue 2017	adults_per_venue_2017	Integer
EGMs per 1,000 Adults 2017	egms_per_1000_adults_2017	Integer
EXP per Adult 2017 ($)	exp_per_adult_2017_aud	Double
LGA	lga	String
LGA Code	lga_code11	String
LGA Name	lga_name	String
Region	region	String
SEIFA ADVDIS RANK COUNTRY	seifa_advdis_rank_country	Integer
SEIFA ADVDIS RANK METRO	seifa_advdis_rank_metro	Integer
SEIFA ADVDIS Rank State	seifa_advdis_rank_state	Integer
SEIFA ADVDIS Score ADVDIS Score	seifa_advdis_score_advdis_score	Double
SEIFA DIS RANK COUNTRY	seifa_dis_rank_country	Integer
SEIFA DIS RANK METRO	seifa_dis_rank_metro	Integer
SEIFA DIS Rank State	seifa_dis_rank_state	Integer
SEIFA DIS Score DIS Score	seifa_dis_score_dis_score	Double
Total Net Expenditure ($)	total_net_expenditure_aud	Double
Unemployed as at June 2017	unemployed_as_at_june_2017	Integer
Unemployment rate as at June 2017	unemployment_rate_as_at_june_2017	Double
Workforce as at June 2017	workforce_as_at_june_2017	Integer

Figure 4: AURIN Dataset Fields

Location Dataset

Another dataset that is being used for the project comprises of generic location parameters of different cities of South Australia and their famous localities. The dataset, after being loaded onto CouchDB, has been analyzed using the MapReduce technology to create a view containing data confined to cities of Victoria. The sole purpose of using this dataset is to merge it with the location parameters in the tweets, so as to infer the localities from which the tweets have been posted. It helped greatly in deducing insightful results for assessing the sentiments of people based on their tweets from various locations in and around Melbourne.

The Location dataset has been downloaded from:

https://www.matthewproctor.com/Content/postcodes/australian_postcodes.json

VI. Sentiment Analysis

Valence Aware Dictionary and sEntiment Reasoner is a lexicon and rule-based sentiment analysis tool that is *specifically attuned to sentiments expressed in social media*. This gives us a score of positive, neutral, and negative sentiment of the text. Moreover, the essential feature of VADER is the compound score(normalized score of positive, neutral, and negative)that tells how positive or negative the sentiment is.

Compound score <= -0.05 denoting a negative sentiment

Compound score>= 0.05 denoting a positive sentiment

-0.05<Compound score<0.05 denoting a neutral sentiment

VADER has better results than other sentiment analysis tools because of the following reasons:

- Punctuation marks for example, "!", "!!!" intensify the sentiment magnitude without the modification of sentiment orientation. The upper-case letters also change the magnitude accordingly.
- Words like "extremely", "marginally" increasing or decreases the intensity. Use of conjunctions also shifts the polarity of the sentiment. The negated sentences are also handled with great accuracy say, "The job market isn't really great"
- The emoticons and slangs are handled that are most common in micro blogging platforms. For example, it calculates the polarity that includes "😊", "☹","SUX" etc.

Error Handling

The stream API 15 minutes window allowing 15 requests per token limits the rate at which we get tweets. To avoid such scenario, we have implemented 5 mins window for each team ember over five team member tokens thus making it 25 mins call difference between per token call to API. The duplicate tweets are handled in manner such that every unique ID of the tweet is stored as the document ID of the CouchDB database. Retweets field in the tweet JSON format is not considered in this project, omitting such fields by the condenser file. Thus, retweets are not stored in database.

In case, the rate window for the stream process exceeds the limit, the code has the on_error methods that handles the 420-error code making sleep the process for 10 minutes before the next request window starts.

```
# restriction of max rate limit
def on_error(self, status_code):
    if status_code == 420:
            self.on_timeout()
# sleep for 10 minutes
def on_timeout(self):
    time.sleep(600)
```

VII. CouchDB

Apache CouchDB is an open source NoSQL (Non relational) database which has been implemented in Erlang. NoSQL databases are non-relational databases which are designed to be scalable and can handle large amounts of data efficiently, also are horizontally scalable. NoSQL databases follows the BASE consistency model which ensures the following properties:

1) **Base Availability**: This property ensures that availability of database is guaranteed but while it is guaranteed the data maybe in inconsistent state.

2) **Soft State**: This property of the database means that the state of the database is no fixed and can be changing over time.

3) **Eventual Consistency**: Eventual Consistency means that the database might become eventually consistent and data will propagate everywhere at some point the future.

CouchDB uses an HTTP based REST API to communicate with the databases, which are collection of individual documents. In CouchDB data is inserted in Json format where data can take any form. Some of the main advantages of CouchDB includes:

1) **Scalability**: CouchDB is designed to be adaptable as it supports both horizontal partitioning and replication and balances the read and write operation when the load level increases.

2) **No read Locks**: In classical relational databases the rows of the databases are locked while an update is happening. This makes it difficult to users to access the database when the update is taking place. CouchDB uses Multi-Version Concurrency Control which manages the access to databases concurrently.

3) **Easy Indexing**: CouchDB allows indexing which makes to easier and efficient to query large amount of data. Indexing makes it possible to search through every row without examining every row with every query.

Views

Views are the primary tools used for everything and reporting on CouchDB documents.

The data in view is sorted by the key of the row. Views are created using the Map-Reduce technique, which comprises of two techniques Map and Reduce.

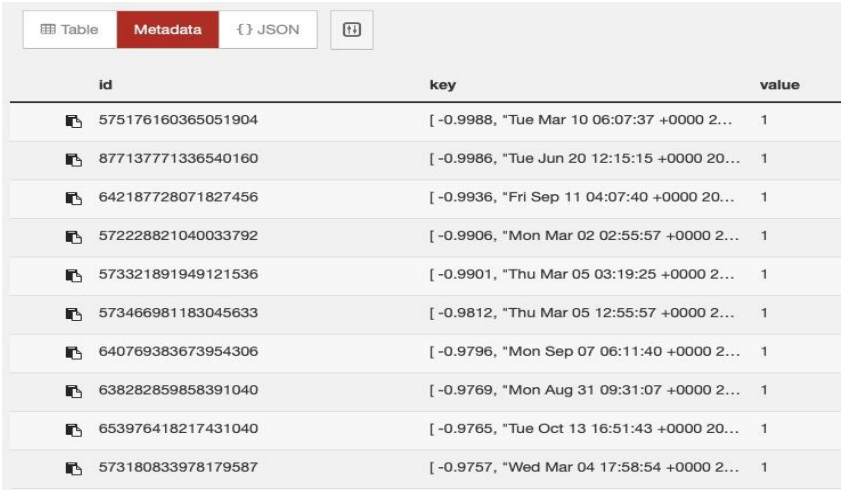
1) Map

Map operation is responsible for performing filtering and searching operations and it emits a key value pair of each document in the database whereas reduce operation produces a summary. MapReduce technique is used to parallelise when working with large datasets. We use emit function to created a key value pair in which a key value is useful to identify the document and value stores the list of specific data we want from that document.

2)Reduce

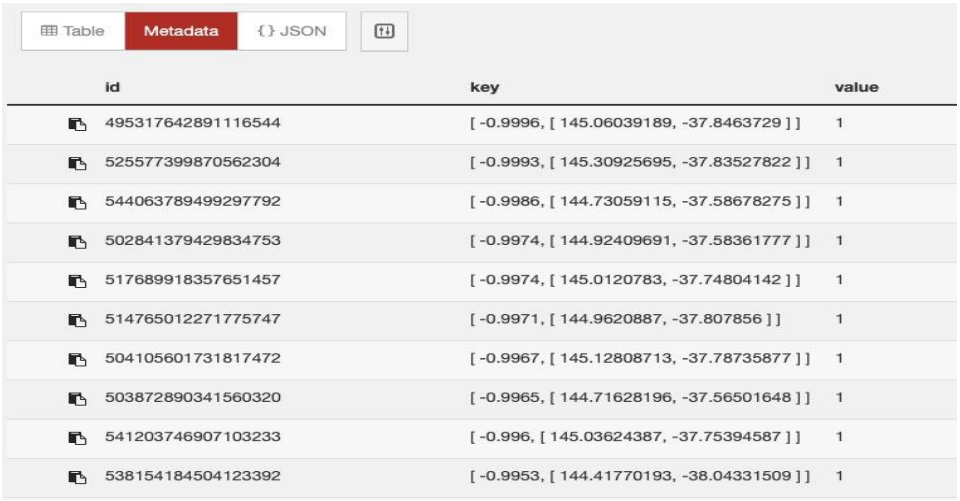
Reduce function is used to create an aggregated results from a view. The output from the map function is aggregated using the reduce function. Reduce function can take two arguments, keys and values list.

View 1: In this view, key contains of the time of the date along with the sentiment value of



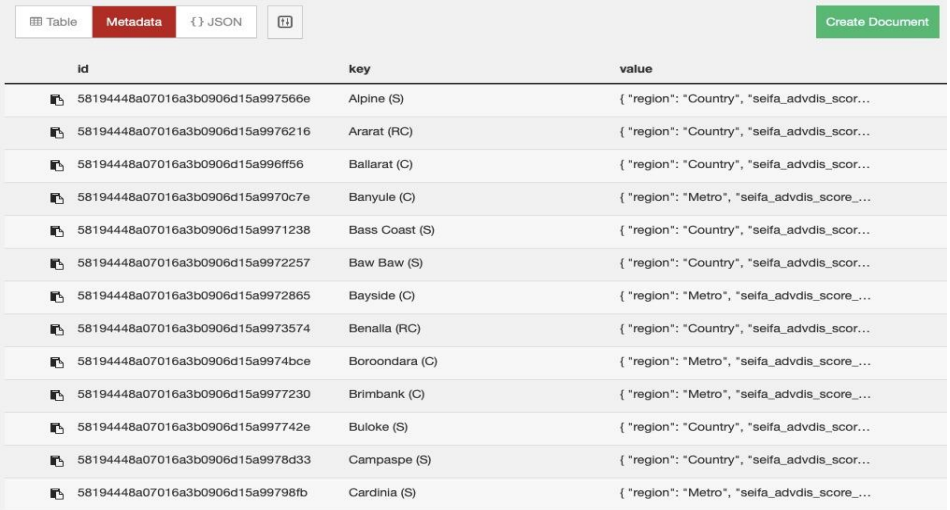
id	key	value
575176160365051904	[-0.9988, "Tue Mar 10 06:07:37 +0000 2…	1
877137771336540160	[-0.9986, "Tue Jun 20 12:15:15 +0000 20…	1
642187728071827456	[-0.9936, "Fri Sep 11 04:07:40 +0000 20…	1
572228821040033792	[-0.9906, "Mon Mar 02 02:55:57 +0000 2…	1
573321891949121536	[-0.9901, "Thu Mar 05 03:19:25 +0000 2…	1
573466981183045633	[-0.9812, "Thu Mar 05 12:55:57 +0000 2…	1
640769383673954306	[-0.9796, "Mon Sep 07 06:11:40 +0000 2…	1
638282859858391040	[-0.9769, "Mon Aug 31 09:31:07 +0000 2…	1
653976418217431040	[-0.9765, "Tue Oct 13 16:51:43 +0000 20…	1
573180833978179587	[-0.9757, "Wed Mar 04 17:58:54 +0000 2…	1

that corresponding tweet. Value column represents the number of such tweets which were tweeted on the same dates with same sentiment value.



id	key	value
495317642891116544	[-0.9996, [145.06039189, -37.8463729]]	1
525577399870562304	[-0.9993, [145.30925695, -37.83527822]]	1
544063789499297792	[-0.9986, [144.73059115, -37.58678275]]	1
502841379429834753	[-0.9974, [144.92409691, -37.58361777]]	1
517689918357651457	[-0.9974, [145.0120783, -37.74804142]]	1
514765012271775747	[-0.9971, [144.9620887, -37.807856]]	1
504105601731817472	[-0.9967, [145.12808713, -37.78735877]]	1
503872890341560320	[-0.9965, [144.71628196, -37.56501648]]	1
541203746907103233	[-0.996, [145.03624387, -37.75394587]]	1
538154184504123392	[-0.9953, [144.41770193, -38.04331509]]	1

View 2: In this view, key contains of the location coordinates of each tweet and the sentiment value of that tweet. Values gives the total values of such tweets which were tweeted from same location coordinates with same sentiment value.



id	key	value
58194448a07016a3b0906d15a997566e	Alpine (S)	{ "region": "Country", "seifa_advdis_scor…
58194448a07016a3b0906d15a9976216	Ararat (RC)	{ "region": "Country", "seifa_advdis_scor…
58194448a07016a3b0906d15a996ff56	Ballarat (C)	{ "region": "Country", "seifa_advdis_scor…
58194448a07016a3b0906d15a9970c7e	Banyule (C)	{ "region": "Metro", "seifa_advdis_score_…
58194448a07016a3b0906d15a9971238	Bass Coast (S)	{ "region": "Country", "seifa_advdis_scor…
58194448a07016a3b0906d15a9972257	Baw Baw (S)	{ "region": "Country", "seifa_advdis_scor…
58194448a07016a3b0906d15a9972865	Bayside (C)	{ "region": "Metro", "seifa_advdis_score_…
58194448a07016a3b0906d15a9973574	Benalla (RC)	{ "region": "Country", "seifa_advdis_scor…
58194448a07016a3b0906d15a9974bce	Boroondara (C)	{ "region": "Metro", "seifa_advdis_score_…
58194448a07016a3b0906d15a9977230	Brimbank (C)	{ "region": "Metro", "seifa_advdis_score_…
58194448a07016a3b0906d15a997742e	Buloke (S)	{ "region": "Country", "seifa_advdis_scor…
58194448a07016a3b0906d15a9978d33	Campaspe (S)	{ "region": "Country", "seifa_advdis_scor…
58194448a07016a3b0906d15a99798fb	Cardinia (S)	{ "region": "Metro", "seifa_advdis_score_…

View 3: In this view, each key gives the region and value gives the information about that region such as its region type, unemployment rate etc.

In the project, the tweets data and the unemployment data from Aurin was stored in CouchDB. The tweets data was taken from the Twitter API which comprised of columns such as location coordinates of the tweets, text of the tweet, date and time of when the tweet was posted, unique id which was used to prevent duplicate tweets from entering the database and sentiment value of each tweet was calculated. The Aurin unemployment data contained the unemployment status in different parts of Australia. Both these datasets were used to relate the unemployment status of the region with the sentiment score of the tweets tweeted from that region with the hypothesis that high unemployment in particular region should give more negative sentiment tweet from that region. The location coordinates from the tweets were used to map the name of the city from where the tweet was posted. Also, this data was used to find the trend of the difference of positive sentiment tweets and the negative sentiment tweets in the Victoria region. CouchDB helped us to index large volume of twitter data efficiently.

VIII. Web application

The web application component was built in python primarily using the dash libraries. Dash is a framework for developing interactive dashboards with dynamic and reactive visualizations. Dash implements flask as well which allows for online deployment of the dashboards.

The dashboard contains four main components that are separated each onto separate tabs. Each component contains a visualization that demonstrates an aspect of the inferences that were made after the aggregation of data.

Data retrieval from the database is controlled by a reactive event that is triggered by the clicking of the button. Deployment of the app is aided by the package gunicorn, a web server gateway interface for python on unix machines.

The choices made by the team during the construction of the application were driven by the necessity for familiarity and straightforwardness. Python was the language that the group was most comfortable with and dash allowed for an efficient implementation without needing to worry about the applications on a finer granularity. Likewise, gunicorn allowed for deployment to be enacted with little issue.

Invocation:

Running the web app from a local machine:

To run the app locally, the following steps need to be followed:

1. If python (3.6) isn't installed on the machine, then the following is required:

```
$ sudo apt-get install python3.6
$ sudo apt install python3-pip
```

2. Install Gunicorn for python 3 onto the machine, the standard gunicorn will not work for files using versions of python 3.0 or greater:

```
 $sudo  apt-get gunicorn3
```

3. Ensure that the required dependencies are present in the system, these are expressed in the requirements file:

```
$ pip3 install -r Cluster-and-Cloud-Project/UI/requirements.txt
```

4. To deploy with gunicorn, move to the dash_files folder and invoke gunicorn with the apps server

```
$ cd Cluster-and-Cloud-Project/UI/dash_files
$ gunicorn3 app:server  -b 127.0.0.1: 8050
```

The resulting web application can be viewed at http://127.0.0.1:8050/app/.

IX. Tweet Scenario Analysis

The primary goal was to investigate the sentiment of local twitter users in particular respect to unemployment that may have arisen due to the lockdown protocols of covid-19. The twitter data was mined to focus on the tweets the mentioned a selection of key phrases or were related to those phrases.

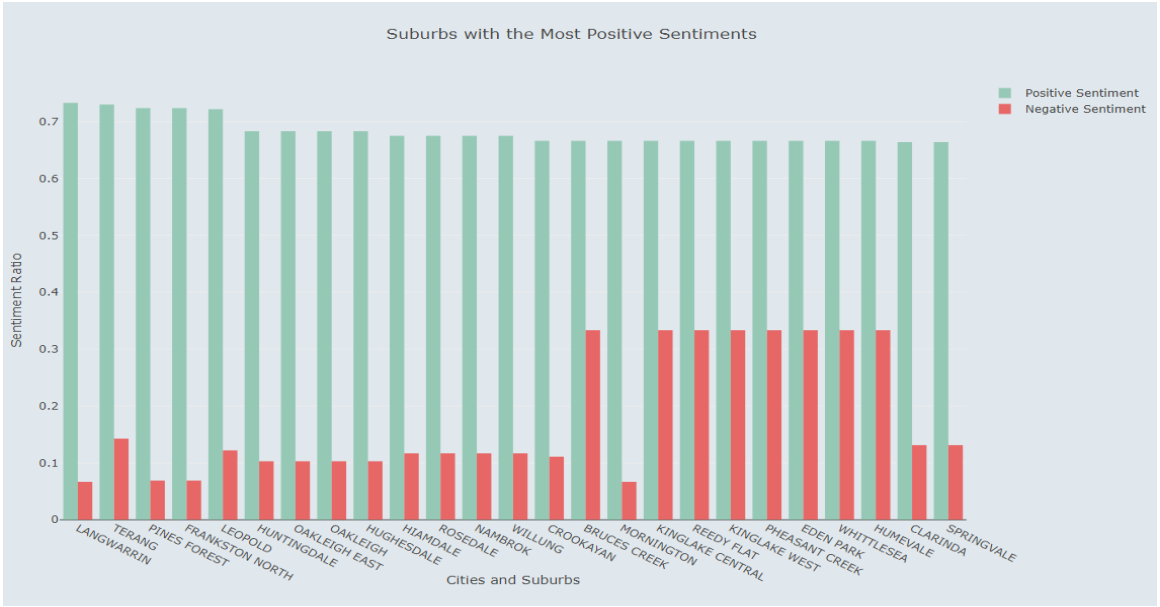


Figure 5

In figure 5, the positive and negative sentiment ratios of the 25 administrative areas with the greatest positive sentiments are shown. All areas are overwhelmingly positive with minimal negativity. The majority of these regions are within the metropolitan area of Melbourne and there seems to be some regional clustering. For example, Hughesdale, Oakliegh, Oakliegh East and Huntingdale are all suburbs that are immediately adjacent.

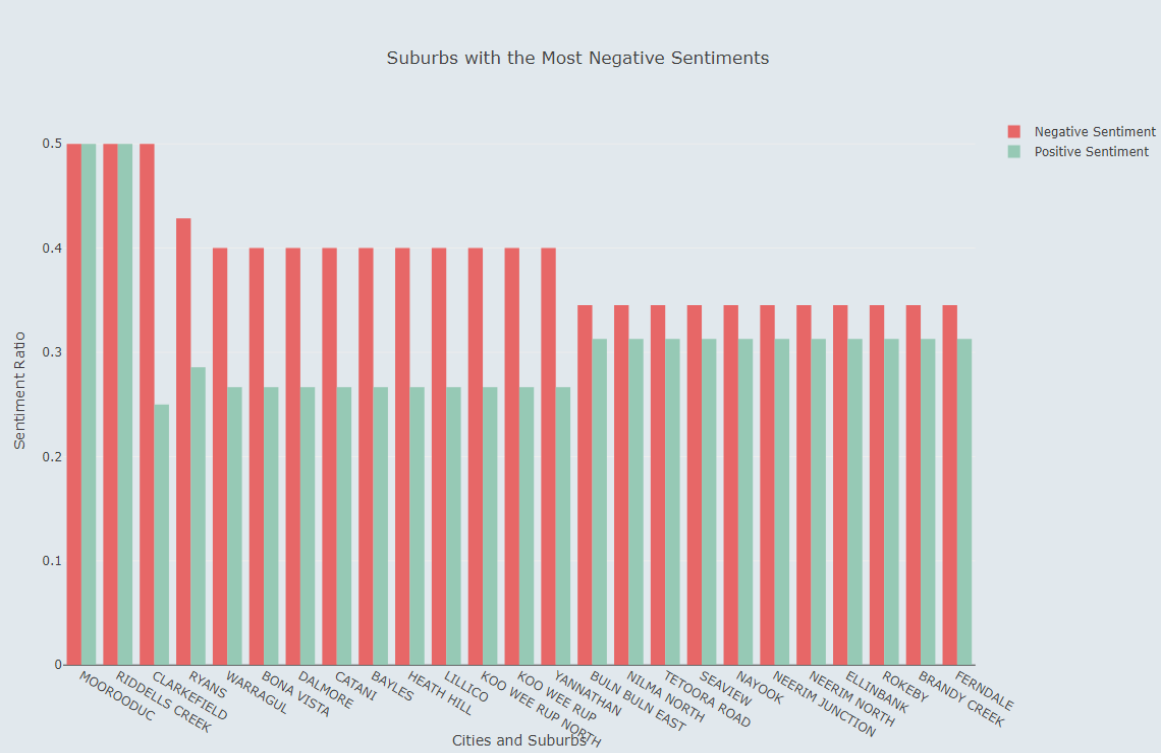


Figure 6

In figure 6, the reverse of the previous is shown and the sentiment ratios of the administrative areas with the greatest negative sentiments are shown. The negativity is not as great as in the prior graph, as no value exceeds 0.5. Interestingly, two of the three areas with the greatest negativity have equal positivity. Another contrast to the prior chart is that there are a larger proportion of regional administrative areas.

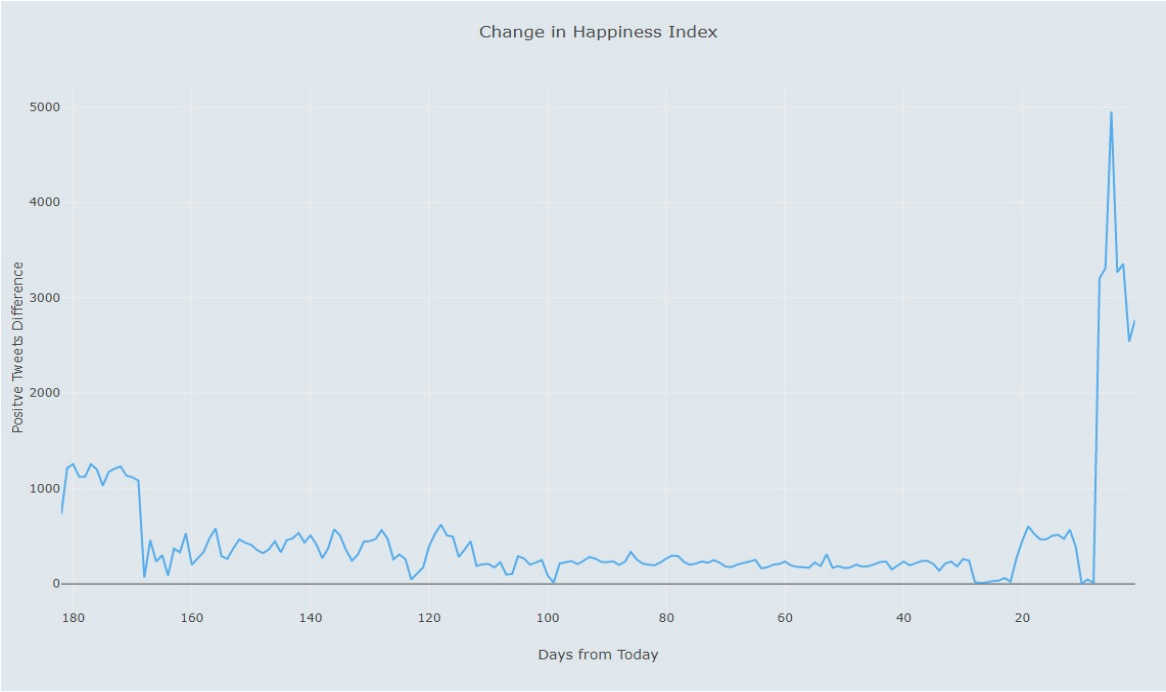


Figure 7

Figure 7 shows the overall net difference of positive and negative tweets each day or a "happiness index". Positive values indicate that there is stronger positive sentiment and negative values indicate stronger negative sentiment. Based on the current results, it can be seen that there has been consistent positive sentiment online. Moreover, this net difference appears to be increasing every day.

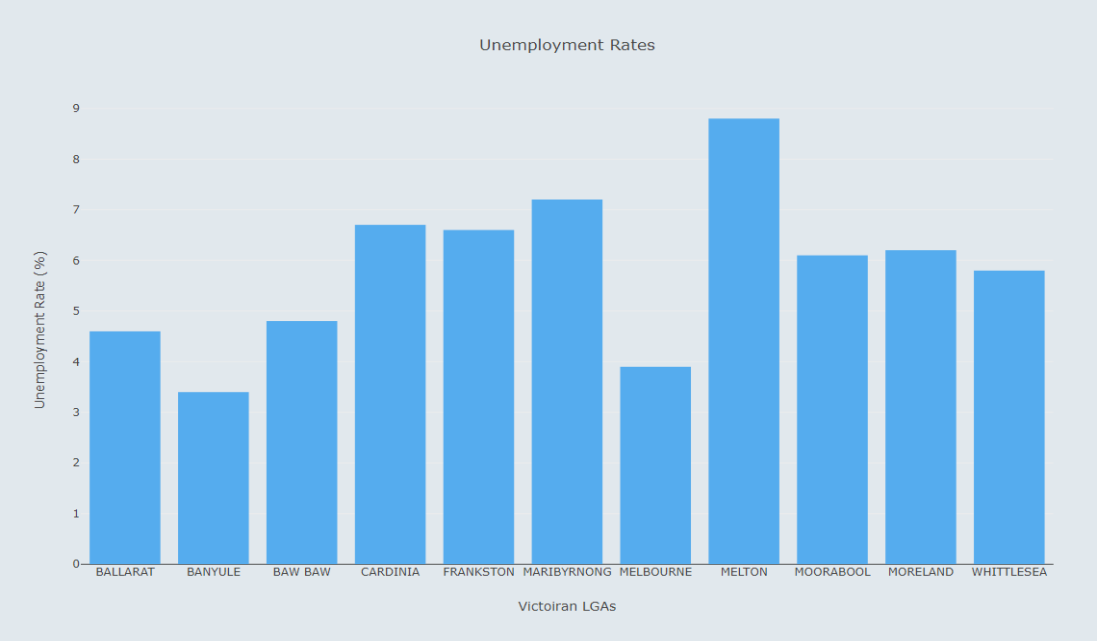


Figure 8

The Final Figure 8 visualizes the most recent unemployment data from various regions in

Victoria. From this it can be gathered that the rates of unemployment have traditionally been higher in more regional areas of Victoria, as the city of Melbourne has by far the lowest unemployment rate of those present.

X. Error Analysis

One of the major challenges in the *Melbourne University Research Cloud* was the computation power of the instances, which was quite low. Hence, there wasn't any way one could increase it even if one wanted to. Services like AWS, Azure allows its users to increase the number of IOP's if one pays more for it. But this feature was not available to use in this project and hence sometimes the code was quite slow and it impacted the performance of the software application. The retrieval of data from CouchDB was one of the areas which was impacted heavily. Moreover, database performance was heavily impacted by the limitations in the number of IOPs assigned to each team.

The other major issue which impacted the project was the way Melbourne Research cloud was hosted. In broader terms if the number of logging users were quite a few the creation

of instances took way longer than normal, this was generally true in the final stages of the assignment submission with more people logging into the network and hence causing a general delay across the line.

Mining of twitter data was quite slow and was troublesome due to the quota of tweets which every individual in the team could mine. This handling caused a delay in the execution of the code and somehow were able to navigate through it using scheduling. One of the team members scheduled the jobs in the VM's to run every 30 minutes without the interference of manual interaction. This was done using CRON jobs in Linux. The other issue was the issue of retweet and duplicated tweets which may be mined across 5 id's of the team members. The issue was dealt by mining the tweets and storing them in the temporary file while the harvester moved on to the next person's id. This way before any tweet was stored in the system it was doubled check against all the other tweets harvested and was only stored if it was unique. This also had major performance implications on the code and hence it was also scheduled in Linux.

Another major issue that the team came across was that inferences could have been staled by the time there was a call to the UI if the respective team member was dealing with static data. To overcome this issue, the team developed the code to always retrieve the latest data from the CouchDB and run the inferences on top of it. This gave way to a new problem in the form of duplication. Since already inferred data would be queried again and would then multiply every time a request is made via the UI. To overcome this, the team managed to write a dynamic inference code and integrate it in the system which would refresh the inferred data every 5 minutes. Also, to make sure that the data was not being unduly deleted and that there is a backup if in case of any mishaps in dealing with the data, the team had created a back-up of the inferred data in CouchDB which would be replicated at regular intervals.

One of the major drawbacks of the system architecture was the non-use of CI/CD pipeline. Due to the time constraint the team was not able to deliver on this front. The pipeline would have been useful to push the code and have an automation testing in place to deliver more stable software/update

XI. Conclusion

It is quite evident from the inferences that have been drawn in the project that how the study that has been conducted for this project has managed to relate it with the respective AURIN database. The visualisations showcased in the FrontEnd/WebApp section bear testimony to the hypothesis that indeed there are a good amount of people who have been negatively affected by unemployment in and around Melbourne. The inferences also show that when the *Corona Virus* hit Australia earlier this year, there was a sudden spike in the number of people being unemployed, and this can be clearly observed in the analysis of the tweets, showcasing a sudden rise in the in the tweets with negative sentiments.

Overall, this assignment was quite wholesome which gave every member in the team a good insight into how the applications/softwares are developed in the industry and how various stages come together to deliver the final product. This assignment showed the use of devops and its vital role in the era of cloud and how it can be leveraged to deliver high

performance applications to the user. The load balancer, replications, clustering, IOPS taught us the value of infrastructure and how progressing without either one of them can have catastrophic implications to the final application.

Appendix A: User Guide for the Software Application

Ansible Folder

1. Under nectar folder run the run-nectar.sh sudo ./run-nectar.sh to create instances, volume, security groups.
2. Under set-env folder run sudo ./run-container.sh to create docker image, container, mount volume, install dependencies, install CouchDB.
3. Under web-app run sudo ./run-web-app.sh to host the web app on the instance.

(Using Windows, the command dos2unix is used for the conversion of sh file before running the sudo ./run-nectar.sh.)

Harvester Folder

The code for the harvester and web app is cloned from the git repository[1]. The scheduler in set-env will run the harvester.py after 30 minutes. historic_tweets.py is used to fetch the historic tweets.

web_app Folder

To run web app locally on your machine the following lines should be all that's needed.

pip install -r Cluster-and-Cloud-Project/UI/requirements.txt

python Cluster-and-Cloud-Project/UI/dash_files/app.py

the app will be accessible on http://172.26.134.13:3000/app/

A quick rundown of the files and their roles:

app - creates and deploys the server+app (main file)

layout - contains the overall structure and design of the app

tabs - contains the structure and elements of each tab

call-backs - functions that allow for interactive updating of elements in the app (optional)

database - database class for handling couchDB

Aurin Folder

This contains the data set used from Aurin website by downloading in JSON format.

CouchDB Views Folder

All the views made using map reduce feature of CouchDB are mentioned under the folder. Charts, plots shown in front end are dynamically loaded by calling these views from the front-end application.

Appendix B: Work Breakdown

This project was assigned to a team of 5 members, all of whom have different academic backgrounds in context to the project specifications. While some of the team members came from a Statistics background, some of them had a sound knowledge about computer applications and softwares. Due to the current global health crisis, there did arise issues in communicating initially, but the gaps were somehow bridged with mutual understanding and the team members never let those gaps come in between the progress of the project.

No team member had any solid background in any of the project requirements, so, though the work was divided initially between the team members, there was always a mutual and mature understanding between the team mates with a readiness to accept challenges and giving a helping hand to each other whenever and wherever needed. There isn't any shying away from the fact that the current situations in the world did at times produce hindrances, but the project was indeed completed by a combined team effort. The work breakdown for every team member is listed below (in no particular order):

Sania Khan	Deployment using Ansible, Twitter Harvester, System Architecture, Story Building
Udit Goel	Deployment using Ansible, Twitter Harvester using Python, Analysis for the project.
Gaurang Sharma	Twitter Harvester (Python implementation), CouchDB (MapReduce functionality), Analysis for the Project
Kanav Sood	CouchDB(MapReduce functionality), Analysis for the project, Inferential Visualisations
Jack Crellin	Front-End User Interface, Inferential Visualisations using various software applications and tools.

The report writing was a collective effort, where every team member contributed by writing and explaining about the work and tasks performed by them for the project.

Appendix C: Project Demonstration Links

GitHub Repository

https://github.com/gaurang96/Cluster-and-Cloud-Project

Deployment Demo

https://www.youtube.com/watch?v=gOhJ_GV73HE

Appendix D: References and Citations

1. Australian Urban Research Infrastructure Network(AURIN):
   https://aurin.org.au/
2. CouchDB Documentation (2017) *What is a View?*
   https://docs.couchdb.org/en/stable/ddocs/views/intro.html#what-is-a-view
3. Apache CouchDB (6th Aug, 2019) by IBM Cloud Education
   https://www.ibm.com/cloud/learn/couchdb
4. Chan Mike, (Mar 5, 2019) *SQL vs NoSQL*
   https://www.thorntech.com/2019/03/sql-vs-nosql/
5. Docker Package Documentation
   https://www.synology.com/en-ca/dsm/packages/Docker
6. Ireland Jordan, (Oct 11, 2019), *Python3 and Flask Installation on Go Daddy*
   https://towardsdatascience.com/installing-python-3-and-flask-on-godaddy-1635fe6f24bc
7. Scott Robin, (Oct 25, 2012), *Twitter Header Image Size/Dimensions*
   https://silicondales.com/tutorials/twitter-header-image-size-dimensions/