COMP90024 2020S1 PROJECT2 GROUP 03 REPORT

A PREPRINT

Xulin Yang
The University of Melbourne
904904
xuliny@student.unimelb.edu.au

Xinyao Niu
The University of Melbourne
900721
xinyaon@student.unimelb.edu.au

Renjie Meng
The University of Melbourne
877396
renjiem@student.unimelb.edu.au

Mingyu Su
The University of Melbourne
912474
msu2@student.unimelb.edu.au

Lu Wang
The University of Melbourne
1054195
lu.wang4@student.unimelb.edu.au

May 29, 2020

ABSTRACT

This is the report for the COMP90024 Group 03 project. In this project we are going to develop a cloud based solution to analyse twitter data. In section 1, we are going to give a simple guide for using the system together with explanation for core features of our scripts and why we realised that particular solution. In section 2, we are going to give a documentation on the system architecture and design. In section 3, we will describe the process and approach of how the data is delivered from organization to the user. In section 4, we are going to give the descriptions, reasons and graphical results for the scenarios we have chosen for the system functionalities. In section 5, we are going to discuss the issues and challenges we faced in this project and our solutions to them. In section 6, we are going to demonstrate some creative ways that out team have adapted in this project. In section 7, we will give a link for videos of demonstration. In section 8, we will give a link to the GitHub repository of our codes. In section 9, we will illustrate team member's contribution to this project.

***K*eywords** Twitter · CouchDB · UniMelb Research Cloud · AURIN · Flask · Docker

Contents

1 User Guide

All the shell commands below are executed under *./ansible/* directory.

1.1 Deploy instances

In this step, we create and mange the cloud infrastructure such as cloud instances, security groups and volumes on the UniMelb Research Cloud in a scalable way by using ansible script and dynamically generate cloud instances' configuration. The **cities** we study are Greater Melbourne, Greater Sydney, Greater Brisbane, Greater Adelaide. Initially, we install dependencies required for running the playbook tasks such as Python-pip, openstack sdk and update pip on hosts. Then security rules for each instance is defined. We need the security rule for each instance is because we want to manage the access of the instance, for example, open port 22 for ssh access. After that, volumes on the cloud are also created in this step because we need to stored twitter in them later. We use all available cloud resources to create cloud instances as demonstrated in the figure 2 and save the created instances ip to "inventory/wm_inventory_file" for later application deployment and dynamically generation of application hosts configuration. The detail can be found in the section 2.7.2. Finally, each instance are created and allocated with resources.

At first, we need to generate the instances configuration file for instances to be deployed. The reason why we need to generate the configuration dynamically is discussed in the section 2.7.1. The generated file will be located in "host_vars/instances.yaml".

Then, we execute the shell script "deploy_instances.sh" from the localhost which run the "deploy_instances.yaml" ansible playbook to manage the creation of cloud infrastructure. The core feature of the playbook is mentioned above. The playbook is realized in the way as illustrated in the table 1.

Hosts	role	description
localhost	common	Install Python-pip, openstacksdk and update pip on hosts
	show-images	Show all available Openstack images
	create-volumes	Create volumes from vars
	create-security-groups	Create security groups and their rules
	create-instances	Create instances on NeCTAR Add hosts to Ansible in-memory inventory And create a file for saving hosts ip to inventory/wm_inventory_file

Table 1: deploy_instances.yaml

1.1.1 How to run

1. *cd host_vars*
2. *python generate-instances.py -n 4*
   where we use "-n" to specify the number of configurations to be generated for instances.
3. *./deploy_instances.sh*

The video can be found in the section 7.1.

1.2 Configure instances

In this step, we configure the instances created in the previous step by setting up the system environment and installing dependencies on the instances.

Initially, we add the NectarGroupKey to enable the ansible can access the Melbourne Research Cloud on the localhost by copying the key to " ./ssh/". Then, we add proxy for each instances to enable the instance to access the internet and can be accessed by the outside. Reboot the machine to ensure the proxy works. In the script we wait a *reboot_timeout: 3000* to ensure the reboot finished and continue the following tasks. After that, we install required dependencies for later tasks. Although, we create the instance with instance_image: 215f9775-483c-4e0a-b703-d21a5c966f2e which is "NeCTAR Ubuntu 18.04 LTS (Bionic) amd64 (with Docker)" who has docker already installed, we still update the docker version to ensure the docker is up to date. Having done the setups, we setup the GitHub proxy on the instance and clone the repository to each instances for the applications. The reasons and details can be found in the section 2.7.3. Then we mount the volume created in previous step to the instance which creates the mounting point and mounts the volume to the file system.

Hosts	role	description
localhost	add-NectarGroupKey	copy /config/NectarGroupKey to ∼/.ssh/NectarGroupKey
instances	add-proxy	Add proxy in /etc/environment Reboot the instance
	install-dependencies	sudo apt-get update; sudo apt-get install [...] pip3 install [...]
	setup-docker	Install docker as well as setting up http proxy for docker
	git-clone-source-repository	configure git ssh key clone the source code repository to the instances
	mount-volumes	mount the volumes for each instances

Table 2: Configure instances

1.2.1 How to run

1. create a file: */config/GitHubKey.pem* with your GitHub private key to enable the instances to access the repository on the GitHub.
2. *./configure_instances.sh*

The video can be found in the section 7.2.

1.3 Application host generation

In this step, we use a python script to read the created instances host ip file from "inventory/wm_inventory_file.ini". And save the allocated host for each application in a generated file "inventory/application_hosts.ini". The reasons why we want this step can be found in the section 2.7.4.

1.3.1 How to run

1. *cd inventory*
2. *python generate-instances.py -c 1 -b 1*

The video can be found in the section 7.3.

1.4 Deploy database and crawler

In this step, we deploy the cluster couchdb and the crawler to the cloud one by one.

Firstly, deploy the couchdb cluster on the cloud before crawling the tweet with crawler.

Then, use Ansible script to copy all the source code file to the server through git.After that, using the templates of Ansible to template cluster couchdb instance IP into the source code in order to allow the crawler connect to valid couchdb node. Lastly, *docker-compose up --build --rmi local* will do the rest and remove all anonymous image to optimize storage.

1.4.1 How to run

1. *./deploy_db_and_crawler.sh*

The video can be found in the section 7.5.

1.5 Deploy back-end server

In this step, we execute the ansible playbook with only one task which is to start the back-end server. In the task, at first, we shut down the previous running dockerized server. After that, we build the dockerized server with new version of back-end server code and start the server with docker.

1.5.1 How to run

1. *./deploy_backend.sh*

The video can be found in the section 7.4.

1.6 Start front-end application

The frontend application can be accessed through the following URL: `http://172.26.132.122:3456/comp90024`. If using shell commands, front end application would start by the steps as follows:

1.6.1 how to run

1. *./deploy_frontend.sh*

2 System Architecture and Design

2.1 System Architecture Design Diagram

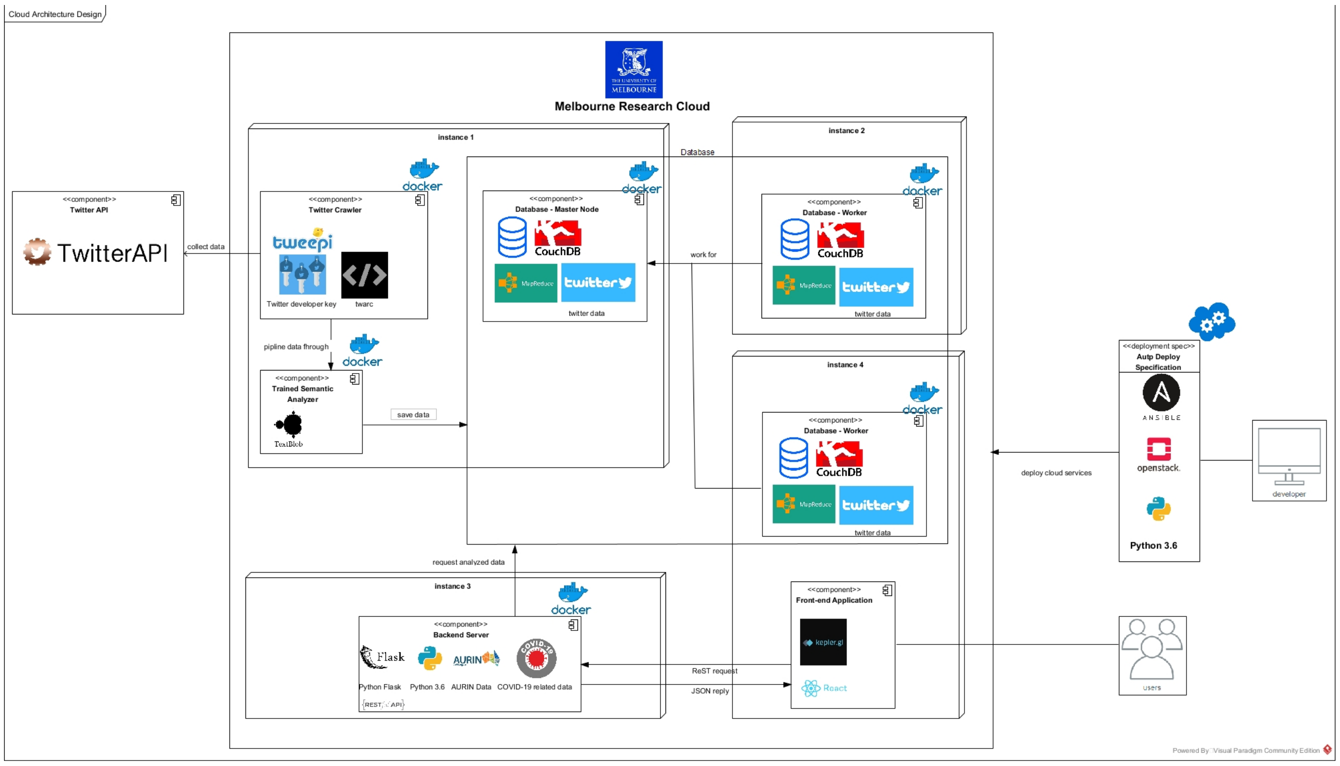


Figure 1: System Architecture Design Diagram

This is the overall diagram for our cloud system as shown in the figure 1. The whole system is deployed on the Melbourne Research Cloud (MRC) and the justification of the using of the MRC can be found in the section 2.2. As the resource allocation described in the section 2.2.1, we have 4 available instances for the our components outlined below. We deploy our instances with:

1. availability_zone: melbourne-qh2-uom
2. instance_network: qh2-uom-internal
3. instance_flavor: uom.mse.2c9g
4. instance_image: 215f9775-483c-4e0a-b703-d21a5c966f2e

Firstly, the user can browse the twitter data analytical result and visualizations by using the Front-end component and the justification of the design of this component can be found in the section 2.3. Secondly, the Front-end component retrieves the analytical result from the system by send requests to the Back-end Component through the ReSTful API.

The justification of the design of this component can be found in the section 2.4. Thirdly, the twitter data are stored in the Couch DB database. The justification of the design of this component can be found in the section 2.5. Fourthly, all the application components in our system are run in docker. The justification of the design of this choice can be found in the section 2.6. At last, we will discuss how our system can be scale out with respect to dynamical changes in the section 2.7.

2.2 UniMelb Research Cloud

The Melbourne Research Cloud (MRC) uses a private cloud deployment model. *It provides Infrastructure-as-a-Service (IaaS) cloud computing to the University of Melbourne researchers, providing access to a robust set of on-demand virtualized computing resources (such as servers and storage). The service makes it easy for researchers to quickly access scalable computational power as their research grows, without the overhead of spending precious time and money setting up their own compute environment.* [1]

2.2.1 Resource Allocation

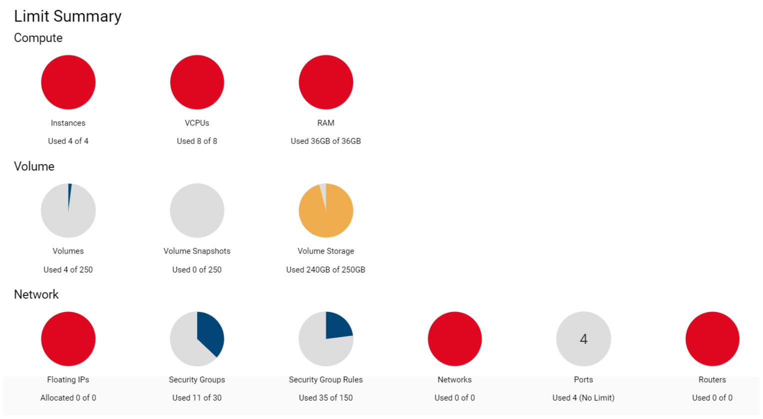


Figure 2: Melbourne Research Cloud resource usage

We are allocated with 4 servers (instances) with 8 virtual CPUs (36GB memory total) on the Melbourne Research Cloud. What's more, we are allowed to create a maximum up to 250 volumes for the instances and the total storage space for these volumes is 250GB. We will use these given cloud resource to build our system for the scenarios as shown in the figure 2.

2.2.2 Advantage

We are going to list and discuss the advantage for using the Melbourne Research Cloud for this project in this section through 9 major areas with 21 benefits and demonstrate how we can benefit from them.

2.2.2.1 Costs

1. No hardware cost
2. Reduce administrative burden
   (a) Computer power without the overheads cost and procurement of applications and infrastructure
   (b) Free cloud service use

The cost is one major advantage for using the UniMelb Research Cloud. Firstly, we have no hardware cost. We can benefit from this in our project because initial investment for hardware is very high in case of an on-premise infrastructure. As a result, we don't need to worry the hardware and infrastructure cost for the cloud system. Secondly, using the MRC can reduce our administrative burden. In one hand, we don't need to pay the electricity bill for the UniMelb Research Cloud system which can save our daily maintenance cost. In the other hand, the usage of the cloud services are free. Thus we have no service fee for the project. What's more, as shown by the figure 3, we don't need to pay for the usage of the cloud services in the MRC compared the expensive cost for the service hosted by the popular cloud provider especially the AWS and AZURE.

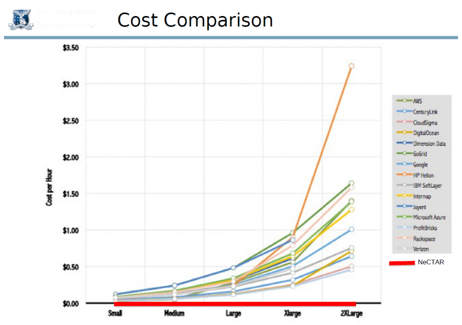


Figure 3: Cloud services comparison

2.2.2.2 Security

1. Achieve high level security
2. Trust
3. Able to set security groups, e.g., open/close ports

The security is one major advantage for using the UniMelb Research Cloud. Firstly, the high level security is achieved. As the allocated cloud resources belong to a single client which is our group. Hence, the cloud infrastructure and systems can be configured by ourselves to provide high levels of security. For example, we can login into the cloud instance by encrypted key-pair, and we can customize the data volume attachment, and etc. Secondly, we can have the trust to the cloud provider. As stated in the *Privacy Policy (MPF1104)*[1], the MRC ensure the data protection principles stated in the law for users. As a result, we can reassured to keep the data on the cloud. Thirdly, we can customize the setting of security groups. We can design the ports usage for each instance freely by adjusting the rules in the security group. Consequently, this improve the security level on each cloud instances.

2.2.2.3 Control

1. Better controls.
2. Share and Collaborate
   (a) Group work
   (b) Clone computation environment for backup or sharing

The control is one major advantage for using the UniMelb Research Cloud. Firstly, we have better controls for data, users and information assets as the MRC uses a private deployment model. Secondly, we can control the share and collaboration. In one hand, as MRC can allocate cloud resources to a group of users, it can control the who and where others can collaborate, share and access data on the cloud. We can benefit this as the collaboration on the cloud make our development more efficient. In the other hand, we can clone the computation environment for backup or sharing. For example, we can use the available OS system image to build the instances on the cloud. Furthermore, we can create snapshots for the cloud which make the cloud environment backup possible.

2.2.2.4 Efficiency

1. Superior Performance
2. Cloud server location

The efficiency is one major advantage for using the UniMelb Research Cloud. Firstly, the MRC has superior performance. As the MRC uses the private deployment model, normally private clouds are deployed inside the firewall of the organization's intranet which ensures efficiency and good network performance. Consequently, we can enjoy a more powerful cloud computation compared to the cloud uses the public cloud deployment model. Secondly, as the physical devices for cloud is in located in the Melbourne university data center, which means the instances are located in one place. As a result, single place location benefits us to transfer data between instances efficiently on the cloud.

[1] `https://about.unimelb.edu.au/strategy/governance/compliance-obligations/privacy`

2.2.2.5 Scalability

1. Create and manage your own dynamic and scalable environment on demand
2. Functionality to adjust instances with scripts
3. Applications and services can scale to meet demand for suitable applications.

The scalability is one major advantage for using the UniMelb Research Cloud. Firstly, we can create and manage your own dynamic and scalable environment on demand by using the allocated cloud resource through the way indicated by the MRS documentation. Secondly, we can have the functionality to adjust instances environment with script through ssh to scale out the management and deployment of our application. Thirdly, our applications and services can scale to meet demand for suitable. As we deploy our applications and services on the MRC, we can use the Ansible script to configure the resource allocation for each application and service to meet the requirements for them on the cloud. For example, install the dependencies for them on the cloud environment remotely.

2.2.2.6 Flexibility

1. Easy customization
2. Reallocate resources flexibility
3. No lock-in issue

The flexibility is one major advantage for using the UniMelb Research Cloud. Firstly, as the hardware and other resources can be customized easily by the development team the MRC has an easy customization characteristic. We can send request e-mails to the MRC customer team to meat our cloud customization requirement. Secondly, we can benefit the reallocate resources flexibility of the MRC. As the MRC provides a virtual machine to users. Unlike a physical machine, a virtual machine can be scaled up and down seamlessly. And when you own all the virtual machines, you can dynamically, wherever they are needed most. Thirdly, we don't need to worry the lock-in issue. Unlike using the public cloud, it is difficult to switch Azure if using AWS because the URC is a private cloud.

2.2.2.7 Availability

1. Access to all researchers and support staff
   (a) Instantly build servers
   (b) 24/7 accessibility
   (c) Detailed tutorials for all OS users
   (d) Easy access and rapid deployment of software applications
2. Snapshot volumes to enable data recoverability
3. Easy access to popular images

The availability is one major advantage for using the UniMelb Research Cloud. Firstly, the MRC is accessible to all researchers and support staff. We can instantly build servers we need them without having to plan, purchase, maintain or dispose of the cloud hardware devices. The MRC is 24 hours and 7 days per week accessible within and outside The University of Melbourne, or anywhere in the world. What's more, the MRC has detailed tutorials for all OS users to guide you how to use the cloud which make its more easy and user-friendly to use. Moreover, we can have easy access and rapid deployment of software applications on the instances by Ansible scripts without to click countless buttons and understand complex configuration steps when using UI management dashboard. Secondly, the ability to snapshot volumes on the cloud enables data recoverability. As a result, it makes the services more available with regard to the failures. Thirdly, we have easy access to popular images, for example, "NeCTAR Ubuntu 18.04 LTS (Bionic) amd64 (with Docker)" with an image id: *215f9775-483c-4e0a-b703-d21a5c966f2e* through the MRC dashboard.

2.2.2.8 Reliability

1. Trustworthy development and maintenance team
2. Reliable cloud structure

The reliability is one major advantage for using the UniMelb Research Cloud. Firstly, the Australian Research Data Commons (ARDC) team which is the team for developing the MRC is only hiring director on the LinkedIn. [2] In the other words, there is no lack of developers or maintainers in the team. As a result, the team can provide reliable support for the cloud usage for us. Secondly, as the MRC uses the openstack, it enables a rapid innovation. OpenStack's orchestration and self-service capabilities offers developers and IT staff with faster and better access to IT resources. Because developers can provision machines rapidly and on-demand, they can significantly reduce development and testing periods and have more freedom to experiment with new ideas. As a result, the ARDC team can develop new features or fix bugs for the MRC quickly which provides more reliable service. So we can benefit the reliable cloud from this.

2.2.2.9 Compliance

The compliance is one major advantage for using the UniMelb Research Cloud. As the MRC is a private cloud, and the compliance is achieved easily in private clouds. We don't need to worry about the law constraint to store on the sensitive data on the cloud. As we just follow the guidance pointed by the MRC, the law on data usage will be fine for us.

2.2.3 Disadvantages

We are going to list and discuss the disadvantage for using the Melbourne Research Cloud for this project in this section through 3 major areas with 8 problems and demonstrate how we can benefit from them.

2.2.3.1 Management cost

1. Staff/management overheads

The management cost can be one major disadvantage for using the UniMelb Research Cloud. As the MRC uses infrastructure as a service (Iaas), we need to build our own cloud infrastructure which implies we need to learn to be a DevOps.

2.2.3.2 Availability

1. Our resource is limited
2. Capacity ceiling
3. No physical access to data
4. Need to spend time to be professional for using the MRC
5. Need to use VPN to connect from outside of university network

Some availability issues can be one major disadvantage for using the UniMelb Research Cloud. Firstly, we have limit resource allocated as demonstrated by the figure 2. For example, we can't store as much data as we want on the cloud. Secondly, we have the capacity ceiling issue. Due to physical hardware limitations with the service provider, there could be a capacity ceiling to handle only certain amount of servers or storage. Thirdly, we can't have physical access to the data which makes the physical data transfer from one place to another place impossible. Fourthly, although we have detailed documentation and tutorial to use the MRC, we still need to spend time to be professional for using the MRC. What's more, in COMP90024 subject the lecture and tutors spend 3 hours lecture time to teach us how to use MRC ,including download openrc shell script, get cloud group password, and etc. Fifthly, we need to use VPN to connect from outside of university network especially during the COVID19 period[3] we need to access the cloud from our home.

2.2.3.3 Security

1. In-proper security management
2. Hardware issue

[2] https://www.linkedin.com/jobs/search/?currentJobId=1813013082&keywords=ARDC

[3] https://students.unimelb.edu.au/student-support/corona-virus

Manage security on the cloud in a wrong way be one major disadvantage for using the UniMelb Research Cloud. Firstly, it can lead to possible data loss and cause security issue. For example, leak the password can let everyone to access your cloud even delete you instances on the cloud. What's more, leak the ssh key-pair and create the service with wrong ports can also cause severe security problems. Secondly, MRC hardware devices can also be a security issue. In one hand, if hardware down, we as users can't fix it by ourselves immediately. We need to report to the maintains team and wait for them to fix which can lead to the service availability issue. In the other hand, if the hardware is stolen, our data is lost and can't recover it..

2.2.4 Summary

To sum it up, although we have some disadvantages to use the UniMelb Research Cloud. Compared to the advantages, we benefit more. As a result, using the MRC to build our cloud infrastructure and develop our cloud services is valuable.

2.3 Front-end Component

Front end is implemented by React.js and apply Kepler.gl and rsuit library to display the data analysis results. Front end allows users to input relevant parameters to fetch filtered data from backend.

2.3.1 Kepler.gl

Kepler.js is a open source project that developed by Uber and dedicated for visualising spatial data. The reason that we choose it as a tool to visualise our twitter data since it can fit our need the best. It is such a powerful tool that allows visualisation of geo-data on the map, which can help us easily find cluster-level interesting feature, but also allow us to perform time series analysis and animation of geographical data. This project contains scenarios 1 and 4 which have time related data changes, each scenario will be attached a time-serial animation to illustrate the interesting found. However, some obsolete modules involved in kepler.gl would cause incapability warnings and even errors, and it costs too much computing resources for running this module.

2.3.2 React.js

This system uses React.js to implement the front end website that show the analysis results with diagrams and animations. By using this, we could easily separate the job of front-end and backend From the website, users can choose the scenario animations with kepler.gl by clicking the scenario category on the side panel, and choosing scenario with parameters in the comparison component, then the results of data fetched from back end would show in the different types of diagrams with library 'rsuit', including pie chart, bar chart, multi-bar chart, and multi-line chart. 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 using, and it is friendly for fresh users to implement web applications. However, the updates of React.js bring many incapable warnings and errors during implementation, and sometimes not clear warning descriptions making the bug fixed periods becomes harder.

2.4 Back-end Component

The back-end server is deployed on the instance 3 as shown in the figure 1. The server is developed by python language with Flask web development framework which deliver the service by accessing it through ReSTful API. The backend server is responsible for receiving ReSTful API resource request from Front-end and return required data to be visualized which is stored in the CouchDB database in JSON format as the response.

2.4.1 Flask

By using Flask to develop the back-end server, we can benefit from following advantages. Firstly, developing a system with Flask is simple. If you understand Python well, then you'll be able to move around and contribute to a Flask application pretty easily. It's less opinionated so fewer standards to learn. Secondly, we can flexibly make modification to existing system. There are very few parts of Flask that cannot be easily and safely altered because of its simplicity and minimality. Thus we can make changes flexibility. Thirdly, compared to Django, Flask has better performance. Flask can be thought as a micro framework being slightly more "low-level" than Django. There are fewer levels of abstraction between you and the database, the requests, the cache, etc. So performance is inherently better from the start. Thus a better performance can be achieved. Fourthly, it is free.

However, there can be several disadvantages for using Flask. Firstly, Flask is a mini web serving framework, it doesn't provide enough scalability for larger projects and is comparatively slower than others. You don't get enough flexibility like in Django. However, it is not a problem for our project. As Flask is a mini web serving framework, it is best suited for small route. We just use URL to get required resource across the web. The Flask community is not as large as nor as good organized as the Django's which means a smaller community is inherently bad. However, this won't influence us too much. Because we can use the COMP90024 Canvas forum to ask questions if any which will be answered by the lecturer or tutors. We are well supported.

To sum it up, we can benefit from using Flask to build the back-end server compared to using Django. Although there can be some drawbacks for using the Flask, they can be overcame and won't affect our develop progress too much.

2.4.2 ReSTful API

We adopt ReST based way to mange user's access to remote resources. By using ReST, we can benefit from advantages as mentioned below. Firstly, compared to SOAP, ReST is easier to use. In one hand, ReST allows a greater variety of data formats, whereas SOAP only allows XML. In the other hand, as ReST is coupled with JSON (which typically works better with data and offers faster parsing), it can offer better support for browser clients, which makes the system is very easy to be debugged by using browsers. Secondly, ReST has superior performance. As we can cache the information that's not altered and not dynamic. In our case, the data from AURIN is cached in the back-end server which can be quickly accessed by ReST with no latency. Thirdly, ReST has good scalability. We can add additional URL for resources access without too much difficulty. Fourthly, as the client only uses Http GET message to request resource from the system, the request call to the system is safe. In other words, do not change repeating a call is equivalent to not making a call at all.

To sum it up, we can benefit from using ReST to build the back-end server compared to using SOAP.

2.5 Database Component

2.5.1 Couchdb

Our system uses a 3-node Couchdb cluster as our database. As shown in the figure 1, they are deployed on the instance 1, 2 and 3 respectively. The cluster database has two major responsibility. Firstly, it stores data received from the crawler and removes the duplicated tweets. Secondly, it is responsible for handling the MapReduce View requests from the backend server.

2.5.2 CouchDB - Pros and cons

Firstly, **the removal of duplicated tweets** are based on the document-id feature of the Couchdb. Couchdb forces each document has an unique id and also allows developer to set the document id with provided value. With these two functionalities, in our database, each tweet is stored as one document and the document id is the corresponding tweet id. Then, the crawler would ask the Couchdb to check if the tweet exits by passing the tweet id, if the tweet does exits, it would be discarded. Thus, there would be no tweet duplication in the cluster database.

Secondly, Couchdb provides convenient HTTP APIs for both creating and querying MapReduce Views. Based on this functionality, the backend server could perform the data computation jobs easily by calling appropriate HTTP calls to create and query specific MapReduce Views.

Thirdly, Couchdb is really convenient to use, particularly setting up a cluster and the Fauxton GUI. With CouchDB, we **only** needs to publish the appropriate ports and make several HTTP calls with correct node IP address, then the cluster is working properly, which is quite convenient. Then, during designing the MapReduce Views, with the help of the Fauxton GUI, we could design as well as test the View on the GUI and see the output directly, which makes development easier.

Lastly, Couchdb gives the developer the full control over the file conflicts resolution. It would sends all the conflicts file to the application, the developer could solve it in any appropriate way.

However, no thing is perfect, so as Couchdb. Refer to the CAP Theorem, Couchdb focuses on availability as well as partition. So, it would loses some consistency over all the nodes. But, in our application, the inconsistency means that the number of tweets on each node would be slightly different and because our front-end data visualization is based on millions of tweets, the small difference of the number of tweets between different nodes would not make a big difference.

In summary, by opting the Couchdb cluster as our database, our system could remove the tweets duplication gracefully as well as design and query MapReduce View easily. However, the Couchdb cluster might be inconsistent, but the inconsistency is acceptable for our application. Thus, it is reasonable to adopt the Couchdb in our application.

2.6 Containerization

This section will mainly discuss two topics related to containerization, which try to explain the reason why we need containerization and why we choose docker as the tool eventually combining with the real world scenario we encountered when developing the whole system in this project.

2.6.1 Why containerization?

This is a fairly easy question to answer. The most critical benefit that we earn from containerization is that we can keep the consistency between production and deployment environment. As our team members working in fair distinct OS, including OSX, Windows, Ubuntu and Deepin, it is very hard to guarantee that one piece of code runs perfectly fun on my laptop will still run on others' computer and servers without strange errors. The reduce the time we need to spend on debugging dramatically, as once the environment is set, the program can run anywhere with the container.

From a more practical perspective, it is much easier to setup the environment for development, migration and deployment. Using docker as an example here, when migrate the code from one host to another, simply pack the source code and unpack on new host only. Then, type docker-compose up, everything will be set without any extra effort. When deploying the service to server, there are two options. It can be either pull the built image from docker hub (or any other place that we can hold our image) or use Ansible script. Both way are relatively easy and straight forward.

Last but not least, it has better performance in terms of security. Different container can communicate through virtual network. The good things about this is that if we deploy a database and load balance backend on a single server, only the backend could access the database. Furthermore, due to this network isolation, we could let 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 request of the container. In this way, we have more flexibility of which one or more port we want to accept request without changing the source code, but only the container settings.

2.6.2 Why Docker?

We use docker as a containerization solution for our project. Even though there are other tools can achieve the similar goal, for instance, Tectonic, docker has its own advantages. There are mainly two reasons for us to choose docker. Firstly, it is much easier to learn. Considering the time constrain of this project, we need to choose a tool that easier to learn, so that we can build something instead of spend most of time just getting familiar with the tool. Also, there are many tutorials and resources out there that cover most of the problem we entered during building up the system using docker. Furthermore, It has a better community. Through Docker Hub, we could get used of many community extended version of docker image but still retain the ability to customize it.

Docker has some other very useful proprieties than can make us life easier. First of all, building by stage. In this project, multiple part may share a similar base environment. In this case, we could first built the similar part first and then each of them build separately on the basis of that, so that we don't need to write the same thing multiple times and still keep the container light-weight. we could also use this properties on merging multiple images and the size of the image would be small than the sum of their sizes. The other one is caching, where we making changes to the docker container, we do not need to rebuild the whole image from the very beginning. This good features boost our developing speed a lot. The last one should be mentioned is docker-compose, where we could start and shutdown services easily. When building our crawler system, as it is a IO intensive task, multi-processing (due to python GIL does not support real multi-threading) is used in this case. When we need to shut down the crawler, we need to use search for the process ID in the system and shut them down one by one, which is not very convenient especially during debugging. With docker-compose, we could easily start and shut the crawler with one single command.

2.7 Dynamically scale out

2.7.1 Dynamically cloud instances creation configure

As shown in the figure 2, we only have maximum of 4 cloud instances and 250GB volume storage available. In the case that there is a change in the cloud resource allocation, our system need to be able to have the ability to dynamically scale up or down with respect to the changes. So we have a python script in "host_vars/generate-instances.py" and we can use "-n <n instances you would like to be created>" to scalably generate n instances with the volume and security group rules for each instance respectively. For example, we want 5 instances than 4 currently, we just specify "-n 5".

2.7.2 Auto cloud host ip saving to inventory

As we need created instances' ip address after the creation instances playbook, we need to store them on disk. After creation complete, the ansible task will use "ansible/roles/create-instances/templates/ansible-hosts.j2" jinja file to store all instances' ip address to "ansible/inventory/wm_inventory_file.ini". As a result, no matter how the instances creation requirements change, the ip addresses saving is not affected. We can scalably handle the ip settings.

2.7.3 Auto GitHub key proxy management

Each time when we want to clone or pull from the GitHub repository, we need to use our own GitHub private key. As a result, the key should be copied from local to instances and deleted after the task to ensure the key is not leaked. And copy our predefined GitHub proxy configuration to remote to grant the access to GitHub. The benefit of have this management is that no matter how we change the instances, the GitHub management won't be affected and security is ensured.

2.7.4 Dynamically cloud application host ip allocation

As mentioned in the section 1.3, we have the python script to dynamically generate hosts ip for each application such as twitter crawler, database, back-end server. When we decided to scale up/down the cloud resource allocation for each application, we just modify the python script and the result inventory will be changed. For example. we add more instances for database or we deploy more crawlers in the system. When the application deployment's Ansible playbook executed, they are able to know the hosts to be deployed accordingly. We can scalably manage the deployment of services.

3 Data delivery

In this section, we will describe the process and approach of how the data is delivered from organization to the user.

3.1 AURIN Data Collection

Initially, to have the data to be compared with twitter data, we need to collect AURIN data from AURIN portal [4]. The challenge for finding appropriate data from AURIN can be found in the section 5.6.1. Having found desired data from AURIN, we download it as JSON format because JSON format is required in the front-end and back-end communication. After that, we prepossess the downloaded data and store them in the back-end server waiting to be sent to the user as mentioned in the section 3.5.1. Finally, we have the prepossessed education-level, foreigner percentage, health service accessibility, income, population demo-graphical data and psychological distress data from AURIN ready to be used in our scenarios.

3.2 Twitter Data Collection

Use the access Token to ask twitter for new data which satisfy our condition. Once the new data is received, do the pre-processing and send the processed data to database for permanent storage and indexing. Notice that as this is mainly a IO intensive work, multi-processing are used. That is, multiple access token are used (for now, we only have 4) and assigned by a dedicate thread to do fulfill the work.

3.3 Twitter Data Pre-processing and Analysis

After data has been retrieved from twitter, we modify and generate several fields from the original twitter object. As we are aiming for the data from only the four great area, including *great Melbourne, great Sydney, great Adelaide or great Brisbane*, we usually have four crawler in charge of each so that we could perform the task in parallel and easily to gain some performance benefits. As the geographical data that we earn from each twitter object is a polygon, we replace it with its centroid to assume this is the real place of the person who send this tweet. By doing so, we could easily plot the person on map to find some geo-related interesting phenomenal. On top of that, we classify each tweet to one of the four great area for later sentiment analysis purpose. Another thing needs to be processed is the text field of each twitter. The text might be truncated sometimes which may affect the result of our sentiment analysis. Therefore, we modify the text filed of tweet, to ensure it contains the full text by replacing truncated one with extended text. Then, the sentiment analysis could be performed on top of the processed tweets. We calculate polarity and subjectivity information from each tweet message by using the TextBolb[5].The higher polarity means the more positive the tweet is and the lower subjectivity indicates the more objective the tweet is. After that, emotional words analysis has been performed. We count emotional words in the tweets message based on the lexicon from this website[6]. Last but not least, we also count the word usage of different length in the tweet message. Words of length 0-4, 5-7 and larger than 7 are considered to be small, medium and long word respectively.

3.4 Data Storage in CouchDB

In this section, we'll describe how data are being processed on CouchDB and how backend retrieved these processed data. In section 3.4.1, we'll describe how we use MapReduce to process the tweets inside CouchDB. Then, in section 3.4.2, we'll describe how the load of CouchDB are being balanced across all nodes. A demo video describing the CouchDB can also be found in section 7.7.

3.4.1 Database MapReduce

In CouchDB, it provides a default MapReduce functionality, which is called View. By creating and using views, we can filter the tweet data, and perform calculations on these data. In this project, we've created multiple views, where each view will extract certain data fields we're interested in from the tweets. Then the backend application will use the corresponding RESTFul APIs to access these views and retrieve the data from CouchDB.

[4] `https://aurin.org.au/`

[5] `https://textblob.readthedocs.io/en/dev/`

[6] `https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm`

3.4.2 CouchDB Load Balance

When we use CouchDB in our project, we've deployed the database to multiple nodes, in which together acts as a cluster. We can access any of the node, and still get the consistent data. However, if all the requests are being sent to one single node, we haven't utilise the processing power of the rest of the nodes. Therefore, to balance the load of the CouchDB, we've distributed the requests across all the nodes. In details, when backend initiates a request to retrieve tweet-related data from CouchDB, it will use the next host address in the list of host addresses for CouchDB nodes. By iterating over this list, the requests will be evenly distributed to each node.

3.5 How data is retrieved from database and sent for visualization in the user end?

In this section, we will describe how data is delivered from database to the front-end to be visualized through the back-end server. In section 3.4.1, we discussed how twitter data to be visualized is collected from the database. Then, in the section 3.5.1, we will discuss how the collected twitter and AURIN data is sent to the front-end through the back-end server.

3.5.1 ReSTful API GET request implementation for visualization data

As mentioned in the section 2.4, we use ReSTful API for data transfer between the user and the remote resource. Having retrieved the data as mentioned in the section 3.4.1, we just wrap the result data into JSON formatted response message to the front-end. Briefly, for each diagram to be visualized in the front-end, we store the data for the diagram as the value and the diagram name as the key in the response JSON message for each scenario through one GET request response. Similarly, AURIN is also delivered in the same way.

3.6 How data from back-end server is visualized to the user?

The backend return the data in JSON format for each GET request. And the data for each diagram is matched with a key in the response. After fetching the data from the backend, the diagrams will be visualized to the user through the screen.

The figure 4 - figure **??** gives a glance at how our front-end application visualize graphical result to the user.

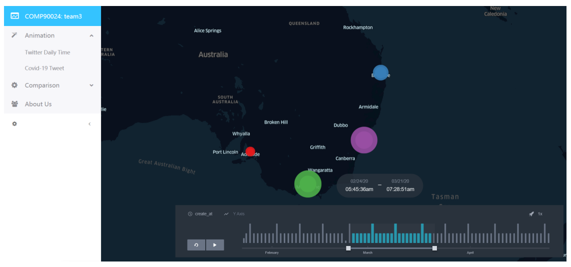


Figure 4: Animation tab

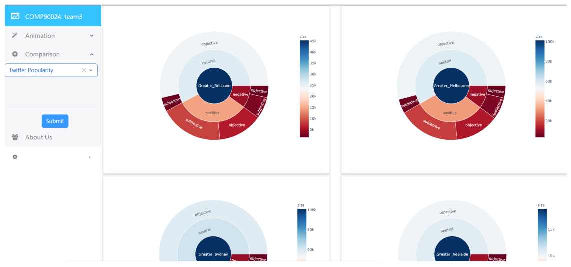


Figure 5: Scenario Comparison tab

The figure 4 shows that the user can starts and pause the animation. What's more, the user can customize the period in between.

The figure 5 shows that the user can browse the graphical result for each scenarios.

4 System Functionality

In this section, the system functionality contains functional achievements and 5 scenarios descriptions,and scenario section includes describing information of 5 scenarios with reasons for choosing this scenario, as well as diagrams and comparisons results.

4.1 Functionalities achieved

1. Using the MapReduce capabilities offered by CouchDB for social media analytic and comparing the data with official data from AURIN.
2. Build the cloud infrastructure to support the social media analytic.
3. Auto harvest twitter data for social media analytic.
4. Auto deploy the CouchDB on cloud for data storage.
5. Clean and analysis the collected data.
6. Visualize analysis result and display it.

4.2 Cities to be studied

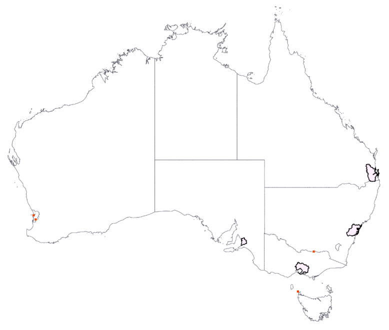
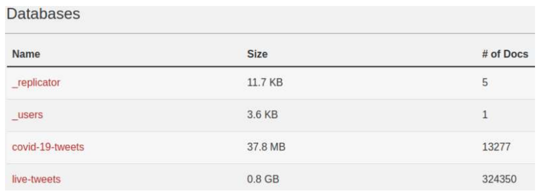


Figure 6: Greater cities

As you can see from the figure 6, these are the greater cities we are going to be studied in this project, which are Greater Melbourne, Greater Sydney, Greater Adelaide, Greater Brisbane. We choose these four greater cities because the majority of the Australia population live here. And study these areas can be representative enough to give valid conclusions.

4.3 Collected data census



Databases

Name	Size	# of Docs
_replicator	11.7 KB	5
_users	3.6 KB	1
covid-19-tweets	37.8 MB	13277
live-tweets	0.8 GB	324350

Figure 7: Collected twitter data census

As you can see from the figure 7, we have collected 324k tweets in total.

4.4 Scenarios

Following are the scenarios we have chosen for the system.

1. Community social media user behavior and characteristic analysis
2. Twitter sentiment analysis for each greater city

3. Correlation between tweets language and local cultures, education level
4. Relation between community development level and people's attention to COVID-19
5. The changes of People's emotionally feel over time during the COVID-19

4.4.1 Community social media user behavior and characteristic analysis.

4.4.1.1 Scenario description

This scenario focuses on analysing the correlation between the daily Twitter usage trend with the portion of different age groups in the four big cities in Australia, including Adelaide, Brisbane, Melbourne and Sydney.

4.4.1.2 Why we choose this?

The combinations of age groups in each city are different, and each age group has different habits and patterns on using Twitter. Therefore, it is valuable to analyse this pattern and then compare with the daily Twitter usage trend to see whether there is any relationship between the age groups and certain Twitter usage pattern.

4.4.1.3 Graphical result

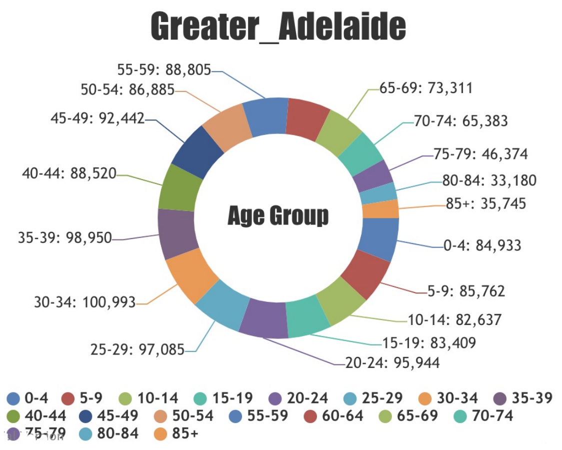


Figure 8: Population of Greater Adelaide

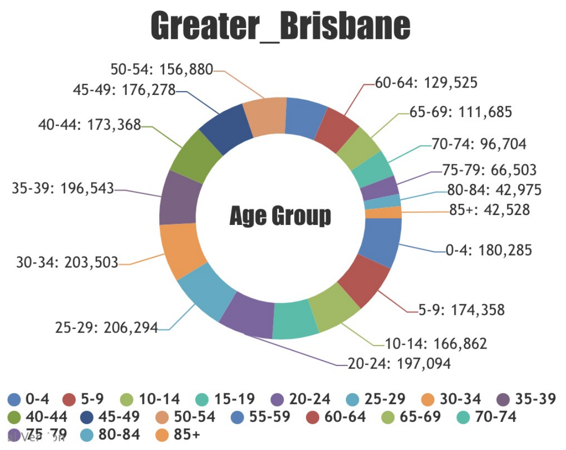


Figure 9: Population of Greater Brisbane

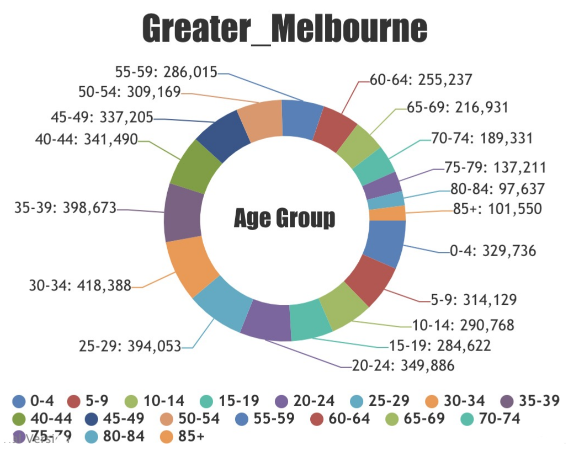


Figure 10: Population of Greater Melbourne

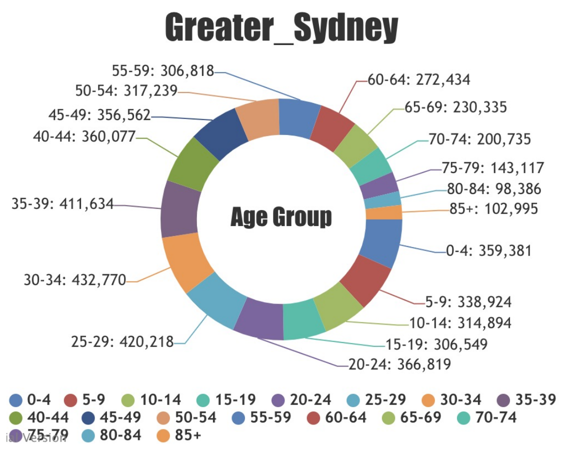


Figure 11: Population of Greater Sydney

These pie charts demonstrate the composition of population in each of the four big cities. By observing these charts just by our eyes, the difference between each of the charts with the other ones is minor.

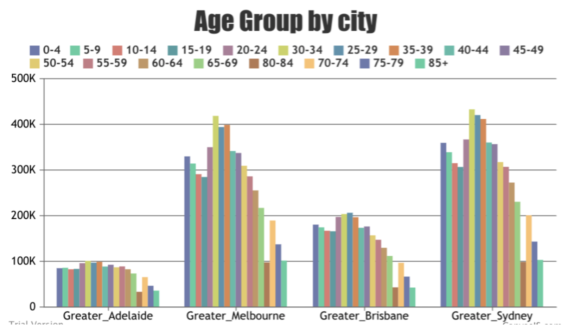


Figure 12: Age group by city

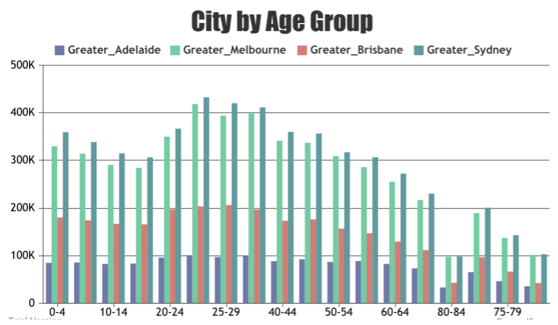


Figure 13: City by age group

In Figure 12 and Figure 13, these bar charts illustrate the population of each age group.

- In Adelaide, the age group with the highest number of population is from 30 to 34 which is 100,993 and the lowest is the age group of 80 to 84 which is 33,180.
- In Brisbane, the age group with the highest number of population is from 25 to 29 which is 206,294 and the lowest is the age group of 85+ which is 42,528.
- In Melbourne, the age group with the highest number of population is from 30 to 34 which is 418,388 and the lowest is the age group of 80 to 84 which is 97,637.
- In Sydney, the age group with the highest number of population is from 30 to 34 which is 432,770 and the lowest is the age group of 80 to 84 which is 98,386.

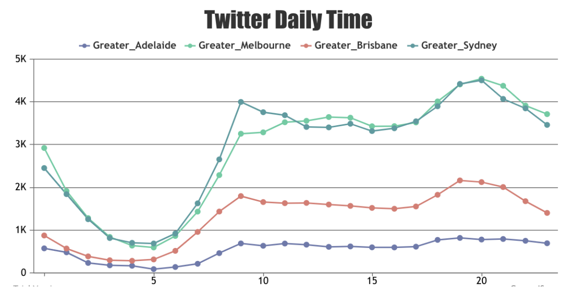


Figure 14: Twitter usage from Monday-Friday in 24 hours

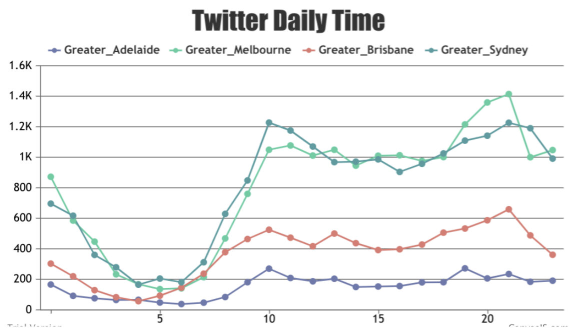


Figure 15: Twitter usage from Saturday-Sunday in 24 hours

The above two line graphs have shown the trend of Twitter usage during weekdays, as shown in Figure 14, and during weekends, as shown in 15. In Figure 14, there is a clear peak hour of usage at 9:00 a.m. in the morning across all four cities. Then the usage trend stays stable in the afternoon until 5:00 p.m. After 5:00 p.m., there will be another small peak period of 4 to 5 hours. After 12:00 a.m., the Twitter usage of the 4 cities simultaneously decrease sharply. During weekend, the peak in the morning will be 1 hour late compare to weekdays as shown in Figure 15. This is an indication of people will generally wake up later during weekends. Overall, these two line graphs have clear indicated how the Twitter usage pattern aligns with people's daily routine. People will start tweeting in the morning when they wake up and stay stable during the day. Then after work, people will enjoy their after work hours and sending more tweets. Eventually, when people start to fall asleep, the Twitter usage will gradually decrease.

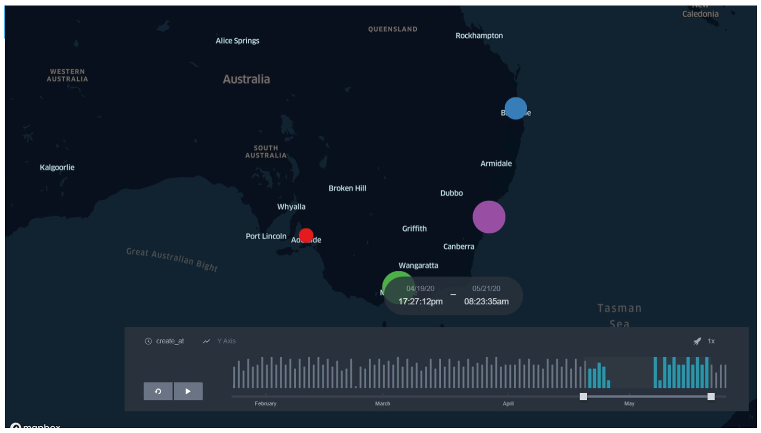


Figure 16: Twitter daily time animation

Figure 16 is the animation for the twitter user's using time in each hour in a period of time. The animation can be seen from the video as listed in the section 7. It provide an alternative way for user to view time serialized data for the figure 14 and the figure 15.

4.4.2 Twitter Sentiment Analysis For Greater City

4.4.2.1 Scenario description

In this scenario, we are going to analyse the text of the twitter from two dimension including the subjectivity and polarity, and trying to find the inner connection between how people raise their opinions and other features that we get from AURIN , for instance, the mental distraction rate of each city, for each of the greater city area. As this is a special time, we can much easier target a user/cluster as people are moving less during quarantine.

4.4.2.2 Why we choose this?

It is worth noticing that sentiment analysis is extremely useful in social media monitoring especially for short text platform, like twitter. It allows us to gain an overview knowledge of the wider public opinion behind certain topics. We can also profile a cluster of people using the public statistics, to help us gain some interesting knowledge between the characteristic of a cluster of people and their opinion.

4.4.2.3 Graphical result

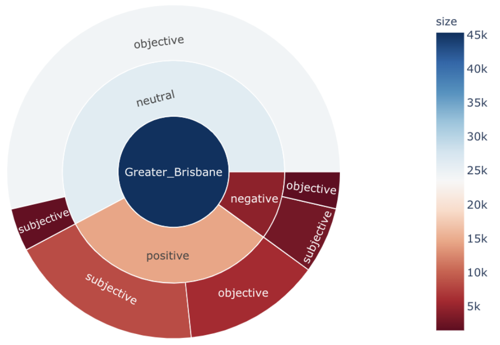


Figure 17: The polarity graph of sentiment for Greater Brisbane.

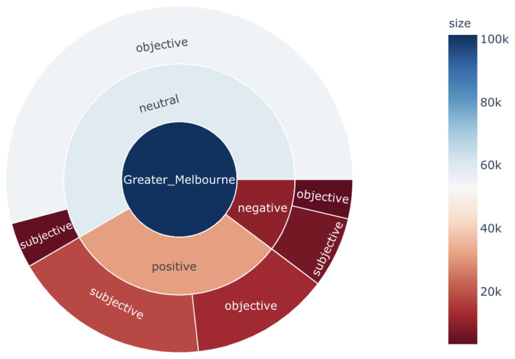


Figure 18: The polarity graph of sentiment for Greater Melbourne

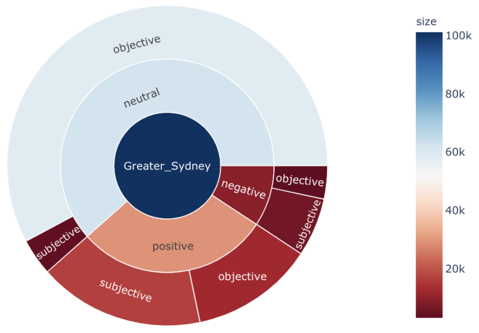


Figure 19: The polarity graph of sentiment for greater Sydney

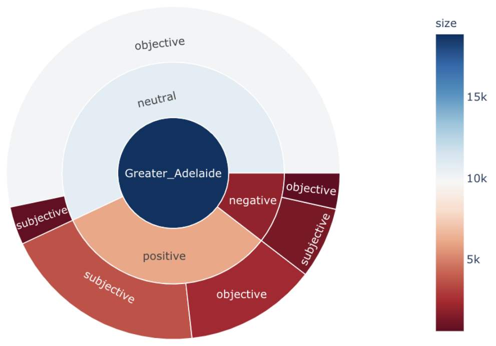


Figure 20: The polarity graph of sentiment for greater Adelaide

There are four pie chart that illustrate the sentiment analysis result for each greater area. The reason to use pie chart is that there are much more twitter in greater Melbourne area and greater Sydney area as there are significantly more people live in these two area[12]. The center of the graph represent the greater area of the data. The second layer represent the polarity of the twitter in that area. The out-most layer represent the subjectivity of each twitter and been classified in two three sections based on its polarity. It is quite obvious that over 90% tweets which have a neutral polarity tends to be objective not matter which great area we are in. Considering the data are collected during corona-virus period and some tweets are collected from corona-virus public data set, only around 10% data are express negative feeling. We may conclude that this is represent most of people are to some extent satisfied with what the government do in terms of this global pandemic. However, by applying a ANOVA test within these four area, we found that the significant level is lower than 5%, which indicate that the difference between these four greater area is insignificant. However, we still could find some slight difference between those area. It turns out that there are about 5% higher neutral polarity rate in greater Sydney area. Combining this with the education data about each greater area[32], we could find that greater Sydney has noticeable lower psychological level stress which may correlate to this discovery.

4.4.3 Correlation between tweets language in greater city and local cultures, education level

4.4.3.1 Scenario description

The main focus of this scenario is on the relationship between the language usage of the tweets and the city's culture and education level. Firstly, we could compare the percentage of English tweets with the percentage of foreigners in each city. Then, we would investigate the relationship between education level of each city between the tweet word length. Words of 0-4, 5-7 and larger than 7 are considered to be short, medium and long respectively.

4.4.3.2 Why we choose this?

The motivation here is that our system could provides the data as well as the computation ability for doing this, because our system could harvest tweet data from different city in real time and carry out Map-Reduce on the collected data. More importantly, Australia is a multi-cultural country. So, it is worth some investigations.

4.4.3.3 Graphical result

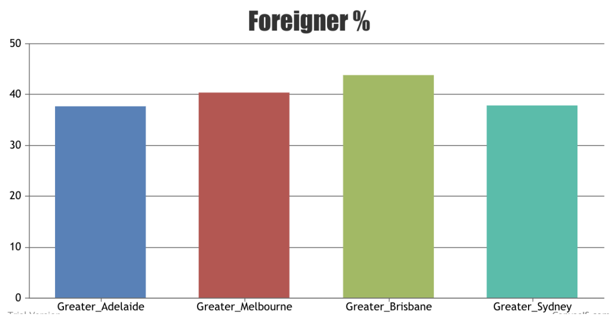


Figure 21: Percentages of foreigners in cities

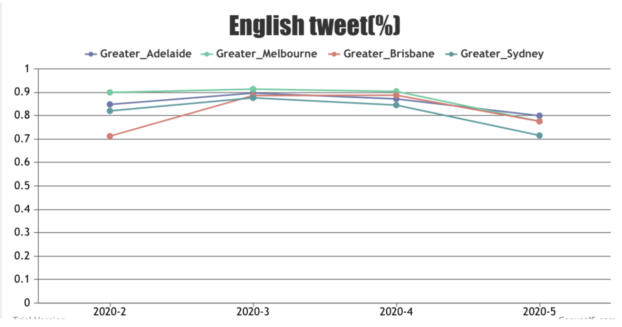


Figure 22: Percentages of English usage in tweets

Figure 21 shows the percentage of foreigners in four cities separately. Foreigner means the resident who is born outside of Australia. The Greater Brisbane has more than 40% foreigners, and the Greater Melbourne has around 40% while both the greater Adelaide and the greater Sydney have around 38% foreigner of the total population.

Then, figure 22 illustrates the percentage of English tweets from February to May in 2020 for four cities respectively. For all the four cities, English is the dominant language of tweeting from February to May.

So, we could see that although around 40% of the residents in each city are born outside of Australia, the dominant tweet's language is still English.

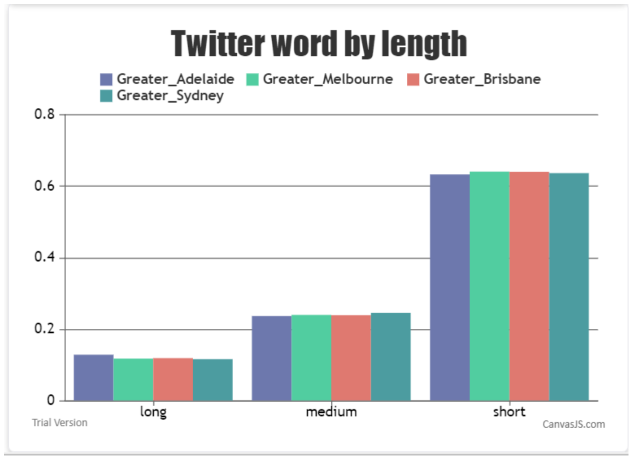


Figure 23: Tweets counts by length

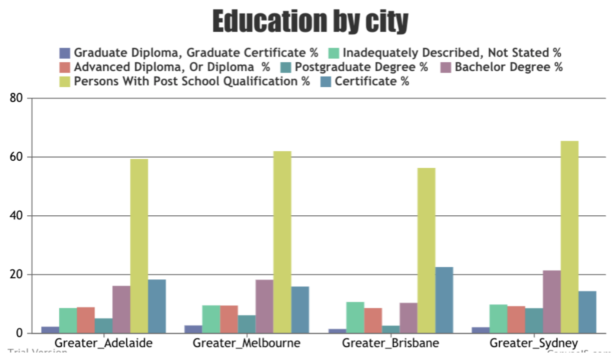


Figure 24: City education level

From Figure 23, we could see that as the length of words increases, its frequency of usage is decreasing among all the four cities. Furthermore, the usage of short, medium and long words are quite similar across all the four cities.

Figure 24 shows 7 education level for each city. From this figure, we could see that in each city, most citizen has education level of post school qualification. Furthermore, the distribution of education for all cities are quite similar, except for the postgraduate education level, the great Sydney has the most people in this education level.

In conclusion, there is no clear evidence to show there is a correlation between education level and the word length used by the tweets.

4.4.4 How the community development level relates to people's attention to COVID-19.

4.4.4.1 Scenario description

In this scenario, we study thing like the correlation between the health service access, income and number of tweets related to COVID-19 for greater cities in Australia. How hotness of COVID-19 is on twitter during the COVID-19 period (public data of COVID-19 cases in AU).

4.4.4.2 Why we choose this?

We choose this scenario because we think how people's attention to COVID-19 related with the public service development. How can the system functionality support this scenario? Firstly, we can use the collected twitter data as illustrated in the system functionality 1. And we will display the graphical result in this scenario for the system functionality 7.

4.4.4.3 Graphical result

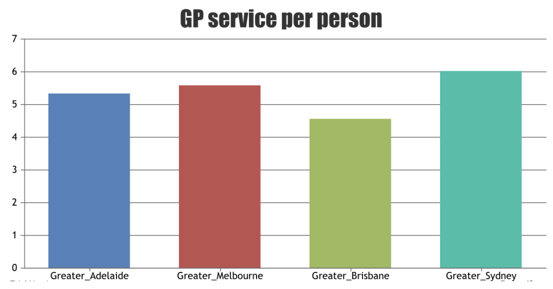


Figure 25: GP service availability

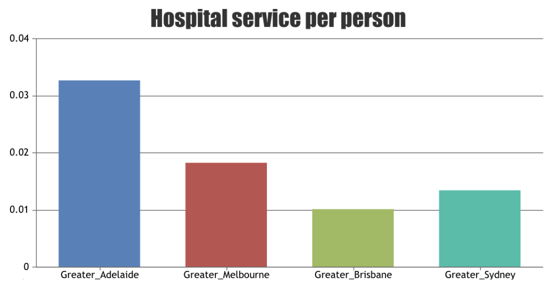


Figure 26: Hospital service availability

As you can see from the figure 25, this is the data from AURIN about people's access to GP service. People in Greater Sydney have highest access with 6 per person. People in Greater Brisbane have lowest access with 4.5 per person.

As you can see from the figure 26, this is the data from AURIN about people's access to hospital service. People in Greater Adelaide have highest access with 0.03 per person. People in Greater Brisbane have lowest access with 0.01 per person. However, there is not much difference between cities in hospital service availability.

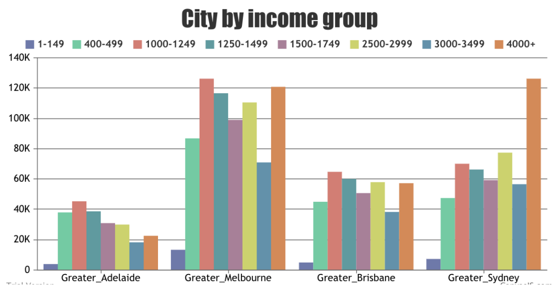


Figure 27: City's income level

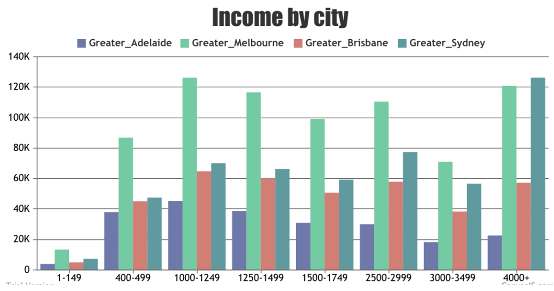


Figure 28: Income level by city

As you can see from the figure 27 and figure 28, this is the data from AURIN about people's income in various greater city. People in Greater Sydney have majority group of people with income $4000+. People in Greater Brisbane are distribute uniformly among all income groups except $1-$149 group.

As you can see from the figure 29, this is the twitter data for people's attention to COVID-19 during the COVID-19 period. People in Greater Sydney and Greater Melbourne has a huge increase in the attention at the start of March and the attention among all cities continuous to the April.

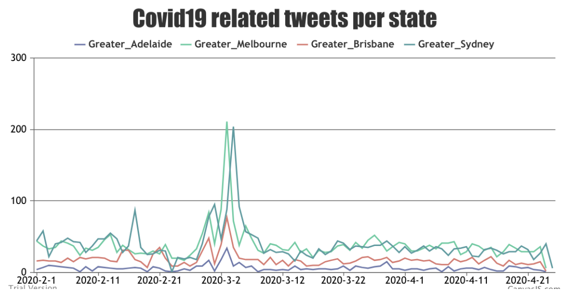


Figure 29: COVID-19 hotness on Twitter

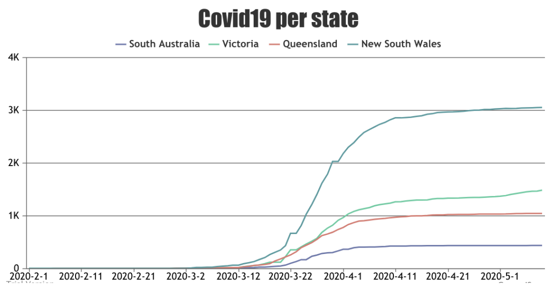


Figure 30: COVID-19 detected cases in states

As you can see from the figure 30, this is the public data we found online for the detected COVID-19 cases in Australia states during the COVID-19 period.

From the figures listed above, Greater Melbourne and Greater Sydney these two cities have better health service accessibility than Greater Adelaide and Greater Brisbane, people have higher attention to COVID-19 on twitter when the COVID-19 has breakout in the Australia. In other words, people's income in Greater Melbourne and Greater Sydney have larger percentage of rich income group compared to the Greater Adelaide and Greater Brisbane.

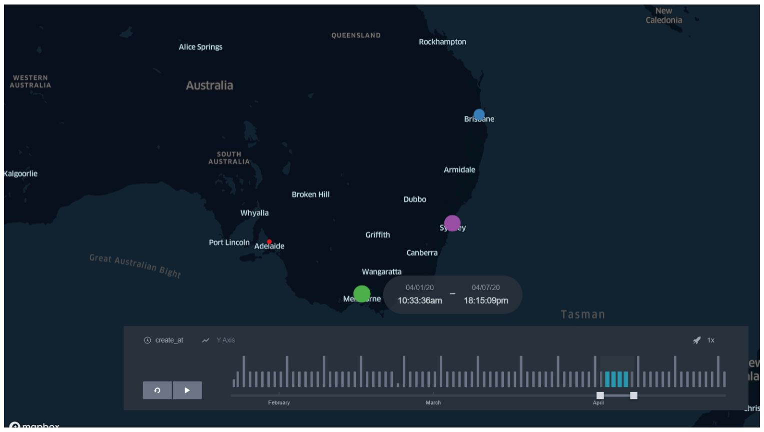


Figure 31: COVID-19 related tweets animation

Figure 31 is the animation for the number of tweets posted related to COVID-19 during the COVID-19 period. The animation can be seen from the video as listed in the section 7. The animation gives another way to represent the figure 29 in a time serialised way for the user.

In conclusion, the better GP service accessibility the city have, the higher awareness of COVID-19 people have. The richer the people in city are, the higher attention to COVID-19 people have on twitter.

4.4.5 The changes of People's emotionally feel over time during the COVID-19

4.4.5.1 Scenario description

In this scenario, we measure the user's emotion when using twitter from the tweet text. The motion learned can be seen in the figure 33. We compare the user's emotion during COVID-19 period in Greater Melbourne, Greater Sydney, Greater Adelaide and Greater Brisbane with AURIN data for these greater cities. The AURIN data we are going to be compared with is cities' high psychological level stress percentage. And we want to study whether people emotionally feel during the COVID-19 period differ geographically?

4.4.5.2 Why we choose this?

We choose this scenario because we think study this topic related to people's emotion can help we to understand the community in Australia better. How can the system functionality support this scenario? Firstly, we can use the collected twitter data as illustrated in the system functionality 1. And we will display the graphical result in this scenario for the system functionality 7.

4.4.5.3 Graphical result

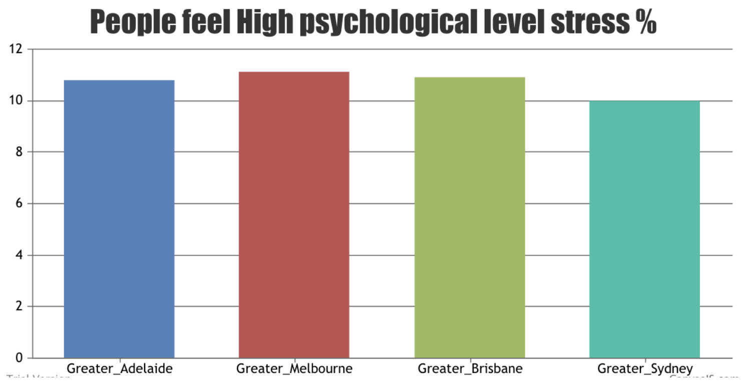


Figure 32: City psychological stress level

As you can see from the figure 32, this is the data from AURIN about how much people feel high psychological level stress among 100 people in each greater city. People in Greater Melbourne are most stressed with about 11.5%. People in Greater Sydney are least stressed with about 10%.

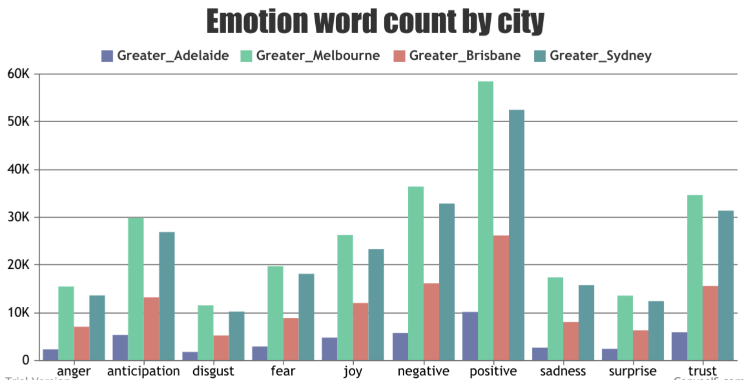


Figure 33: Tweets emotional words count

As you can see from the figure 33, this is the data we categorized from twitter. We use the blob library as shown in the figure 44 to learn the emotion among the tweet text. The top 10 emotion we learned are: anger, anticipation, disgust, fear, joy, negative, positive, sadness, surprise and trust with counts legend by cities.

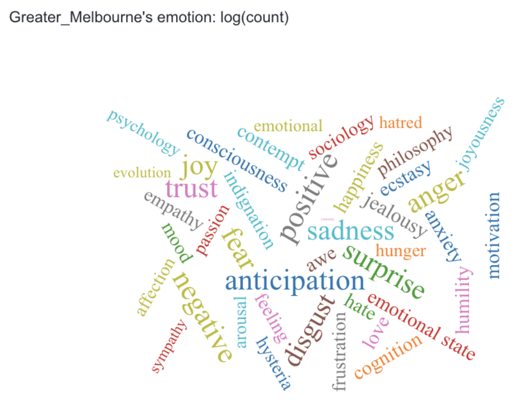


Figure 34: Greater Melbourne emotion word's word cloud

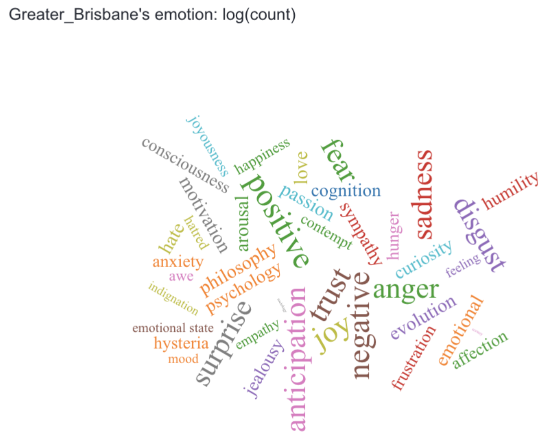


Figure 35: Greater Brisbane emotion word's word cloud

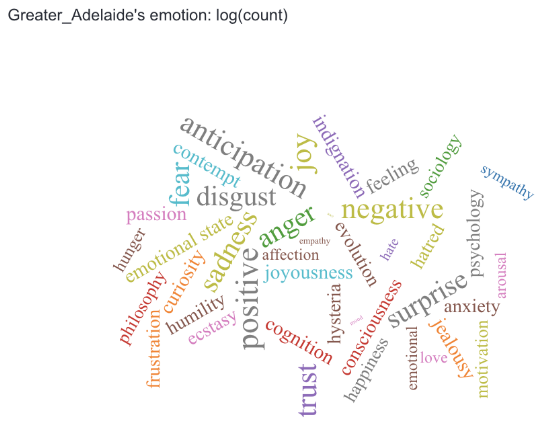


Figure 36: Greater Adelaide emotion word's word cloud

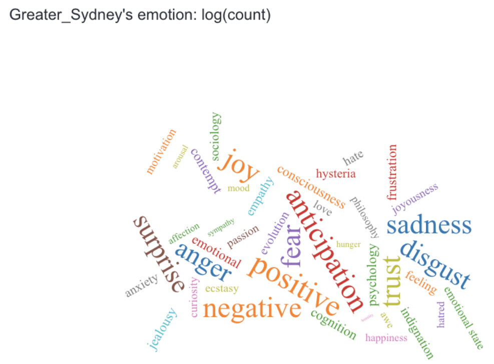


Figure 37: Greater Sydney emotion word's word cloud

The figure 34 - the figure 37 are the word cloud for all emotion words learned for various greater cities. But we have *log(count)* in this case to prevent the word's size differ too much as the count may.

From figure 32, although people in Greater Melbourne are more stressed than people in Greater Sydney, from figure 33, Greater Melbourne have more positive tweets then all other cities. From figure 34, the major emotions people have in the Greater Melbourne are positive, negative, trust. From figure 35, the major emotions people have in the Greater Brisbane are positive, negative, anticipation. From figure 36, the major emotions people have in the Greater Adelaide are positive, trust, negative. From figure 37, the major emotions people have in the Greater Sydney are positive, negative, trust.

In conclusion, although some people feel negative in all greater cities, the majority of people still feel positive on twitter during the COVID-19 period.

5 Issues and challenges

5.1 Error handling

5.1.1 Back-end server error handling

Firstly, as we use Flask with ReSTful API, the GET request parameters are required respectively for various scenarios. So in the python code, we set the parameters in the request URL required. So the system won't be crashed by incomplete parameters request. Secondly, the back-end server won't process undefined GET request and a 404 NOT FOUND error will be returned. Thirdly, if the request contains unknown parameters, the back-end server still works without crashing as it uses Python exception to handle such scenarios and ignore the parameter's value if any.

5.1.2 Front-end application error handling

Firstly, we have restrict the user input by letting user to choose the parameter for the scenario instead of letting user to type them. As a result, we prevent the front-end application from the parameter errors. The figure 38 gives an example.

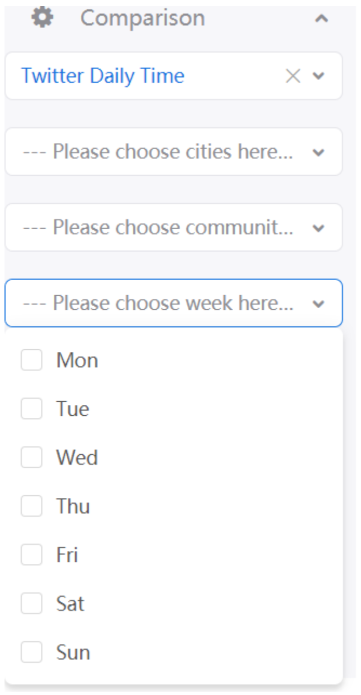

Comparison
Twitter Daily Time
--- Please choose cities here...
--- Please choose communit...
--- Please choose week here...
Mon
Tue
Wed
Thu
Fri
Sat
Sun

Figure 38: User input restriction

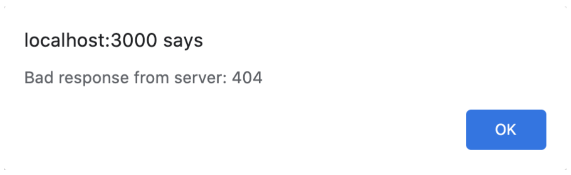
Secondly, we popup error message for users. In figure 39, this an example error notification. In figure 40, the user has missing one parameter input but still submit the request. Then our application will notify the user with the popup error prompt.

5.1.3 Couchdb error handling

The crash of server is inevitable in the real world and it may happen at any time. Thus, our team decide to use the Couchdb cluster as our database instead of just one standalone Couchdb database. With the cluster database, our system could provide more reliable service compared to the single node database.

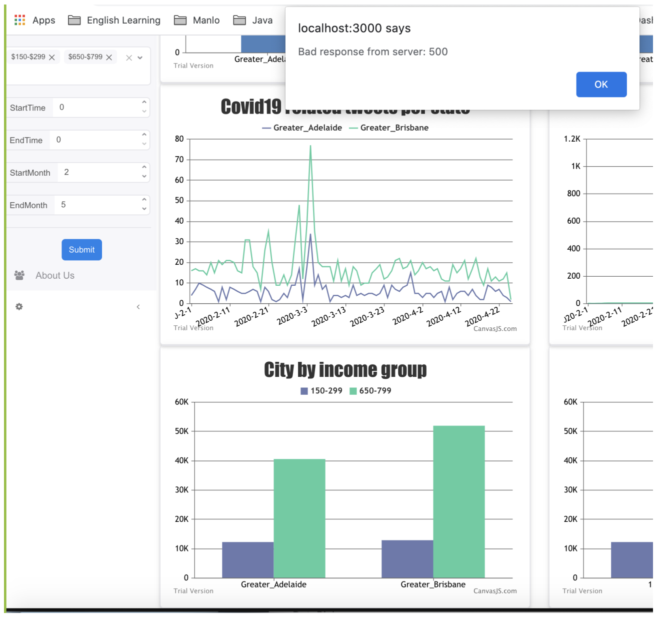
5.2 UniMelb Research Cloud using difficulty

During using the Unimelb Research Cloud, our team find that if we delete the instance and then create a new instance immediately, there is some probability the new instance have the same IP as the deleted old one. Then, when we cannot ssh onto the newly created instance, the error message would say the identification of instance is changed.


localhost:3000 says

Bad response from server: 404

Figure 39: Example user frontend error notification 1


localhost:3000 says

Bad response from server: 500

Figure 40: Example user frontend error notification 2

5.3 Twitter

5.3.1 Mining twitter content limitation

To collect the data from twitter, there are mainly two different ways. The first one is using a spider the query the data from twitter. The pros of this approach is that we could skip the restriction of twitter to get twitter data beyond 7 days boundary and rate limit (quota imposed by the Twitter APIs), however, we need to frequently change the proxy to prevent twitter from banning our IP address. Reducing the query rate and combine it with proxy pool can overcome this. But more importantly, the other disadvantage by using the spider is that we are missing some of the meta-data of the tweets, including the geographically information about the person send the twitter which is critical for our analysis. Therefore, we decided to use some twitter data from public data set and the official twitter access token as the sources of our data. In this case, we could get both old tweets and new tweets.

In order to retrieve old data from public data set, we need to *hydrate* the twitter data from simple twitter id to twitter object. A python package call Twarc are used here to get twitter object from id, and automatically handle twitter rate limit by sleep the specific process. The same package are used for streaming new data to database as well.

5.3.2 Language processing limitation

It is known that tweet is a classical example of short text corpus. The language analysis of short text is challenging, since the text contains limited information and this does not provide as much as information like articles do. Thus, our system collects more than 32k tweets to make sure the dataset is **representative** for each city, and by this the information shortage of tweet could be solved.

5.3.3 Sentimental analysis

The extraction of sentiment from the text is a challenging task, because the raw data contains noise information. Thus, our system use the nltk package to perform data clean, sentence segmentation as well as the word tokenization, in order to remove as much as possible noise from the raw data. Then, as mentioned in section 3.3, our system uses the TextBolb library to extract the polarity and objectivity from the tweet message.

5.3.4 Different Tweet Standard

During storing tweets into the couchdb cluster, we found that tweets might follow different tweet object standards. For example, one tweet we extracted from March-2020 names its tweet message filed as *"full_text"* while another tweet from May-2020 names the same filed as *"text"*. This would make later MapReduce View more difficult, since same information may have different name. So, before send it to the cluster, we name the tweet message filed of all tweets as *"text"*. Furthermore, some tweet's message filed are truncated, thus we replaced it with the full text in the tweet object.

5.4 Containerization

We discuss the difficulties we faced during containerizing our applications in this section.

5.4.1 CouchDB

During using Docker container to deploy Couchdb cluster, it is important to export **all** the correct ports for the container in order for the couchdb nodes to communicate with each other in order to set up a cluster.

5.4.2 Flask-restful

It is quite challenging that to setup the environment for Flask backend as it require may dependencies that could only found in different package manager. It took us a while to figure out how to use *wget* to quietly install the Anaconda from the server. Furthermore, some software dependency would not be installed by *npm* (nodejs package manager). It assume some shared system level package installed but it is actually not there due to docker version Ubuntu only contains minimal libraries that required by Ubuntu. We could only add one dependency from the stack trace report of the program each time. Thankfully, the docker cache made the rebuilding fairly fast so that we could figure the whole set of dependency quickly. Apart from that, the environment inside container (Linux kernel) is different from our developer system which is Windows, the imports that works on his laptop does not work inside docker container. This also took us sometime to figure out the real reason. After flatten the file structure, everything is ready to go.

5.5 Database

5.5.1 Duplicate tweets

As motioned earlier in section 2.5.2, **the removal of duplicated tweets** are based on the document-id feature of the Couchdb. Our database stores each tweet as a single document and use its tweet id as the document id. Then, the crawler would ask the Couchdb node to check if the tweet exits by passing the tweet id, if the tweet does exits, it would be discarded. Thus, there would be no tweet duplication in the cluster database.

5.5.2 The bug of cluster Couchdb itself

During setting up the Couchdb cluster on the Melbourne research cloud, our team failed a lot at the start, although we **strictly follow** the Couchdb setup documentation. It always returns the {"error":"unknown_error","reason":"undef","ref":1124911208}. And after researching, we found the solution on this Github issue[7]. It says "if the / path is not accessed before finish_cluster is called,{"error":"unknown_error","reason":"undef","ref":1124911208} is given." Then, we add commands to access the / path before finish_cluster. After adding this one more extra step, the setup of cluster Couchdb is successful. However, this extra step is **not mentioned** in the official Couchdb documentation.

5.6 AURIN data collection

5.6.1 data searching

AURIN is crucial infrastructure for researchers, government and industry, accelerating research into our towns, cities and communities. It has 5504 data sets from 139 organization categorized in various spatial level [8], for example, statistical area, local government area. As a result, looking for related data to be used to couple with twitter data analysis is like fishing for a needle in the ocean. We spend a lot on looking for appropriate AURIN data sets.

5.6.2 data prepossessing

Firstly, the data set we found on AURIN are categorized in local government level which is not what we want, i.e. greater city level for Melbourne, Sydney, Brisbane and Adelaide. As a result, we write prepossessing python script for each data to merge several local government level data entries to one greater city level entry to suit our analysis

[7]https://github.com/apache/couchdb/issues/2797
[8]https://data.aurin.org.au/

scenario. The difficulty is that each greater city geographical level have more that twenty local government area. As a result, we manually set local government area code for each greater city as a list and merge the data accordingly as shown in the appendix 10.1. Secondly, we need to deal with missing data. As some data sets, there can be some missing entries. We replace them by 0 for no data.

6 Creativity

6.1 Continuous Integration

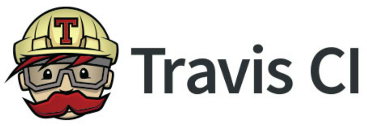
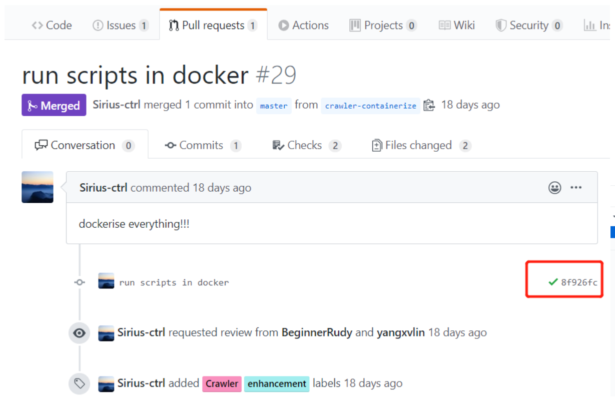


Figure 41: Travis.ci



run scripts in docker #29

Merged Sirius-ctrl merged 1 commit into `master` from `crawler-containerize` 18 days ago

Sirius-ctrl commented 18 days ago

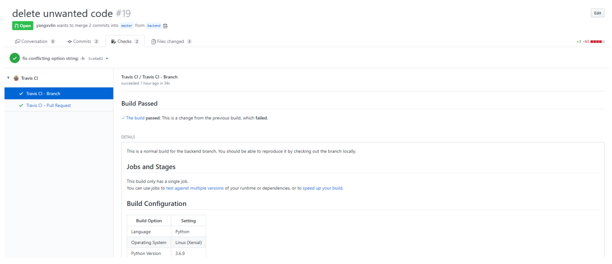
dockerise everything!!!

`run scripts in docker` ✓ `8f926fc`

Sirius-ctrl requested review from BeginnerRudy and yangxvlin 18 days ago

Sirius-ctrl added Crawler enhancement labels 18 days ago

Figure 42: Travis.ci for team collaboration



delete unwanted code #19

Open yangxvlin wants to merge 2 commits into `master` from `backend`

fix conflicting option string: -h `5cd4a02`

Travis CI / Travis CI - Branch
succeeded 1 hour ago in 34s

Build Passed

✓ The build passed. This is a change from the previous build, which failed.

DETAILS

This is a normal build for the backend branch. You should be able to reproduce it by checking out the branch locally.

Jobs and Stages

This build only has a single job.
You can use jobs to test against multiple versions of your runtime or dependencies, or to speed up your build.

Build Configuration

Build Option	Setting
Language	Python
Operating System	Linux (Xenial)
Python Version	3.6.9

Figure 43: Travis.ci for auto testing

We use the Travis.ci as demonstrated in the figure 41 for the continuous integration during the system development in the GitHub repository. We have several reasons to make this decision. Firstly, continuous integration can automatically run tests for the system which can let us benefit from the automation analogs to the benefits for the dynamical deployment. As you can see in the figure 43, the Travis.ci auto testing service is running for the pull request. Secondly, it can decrease code review time for better team collaboration. As you can see in the figure 42, all team members can directly see the testing result on the GitHub page.Thirdly, it is easy to use and it has student account premium.

6.2 Diagram Collaboration

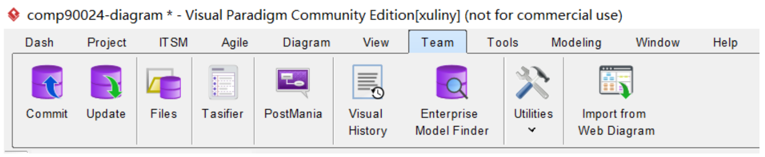


Figure 44: Visual Paradigm Diagram

Diagram designing for the cloud system structure requires collaboration. The visual paradigm diagram has community version which is free and has online repository for diagrams which is analog to GitHub repository but it is for diagrams not codes. As shown in the figure 44, one team member can commit changes on the diagrams and others can update and see the changes which makes the collaboration easy.

7 Video Link

Here is the youtube video list for all the videos listed https://www.youtube.com/playlist?list=PL3rzNLhlw3lEjaleheqtzTwldK0Ih0WBx

7.1 dynamic instance creation & ansible instance creation

https://youtu.be/Vra5LKE59jA

7.2 ansible instance config

https://youtu.be/1rew1qxwqFE

7.3 dynamic application instance allocation

https://youtu.be/S7UQqnWT3jM

7.4 ansible backend deployment

https://youtu.be/fJEajx3Y-tc

7.5 deploy couchdb and crawler

https://youtu.be/NQDD-EJcJIo

7.6 CouchDB utilization

https://youtu.be/My_GMzdd81E

7.7 Crawler demo

https://youtu.be/PE1zy-Hl9Vo

7.8 Crawler utilization

https://youtu.be/Q1oeL8ShoKY

7.9 Live demo

https://youtu.be/noAKgXqnqUw

8 GitHub Link

For the implementation requirements for the version-control system for sharing source code. We have selected the GitHub as we are all familiar with this tool. Link: https://github.com/yangxvlin/comp90024-project2.

9 Individual contribution

Name	Contribution
Xulin Yang	Report Latex repository administrator, GitHub repository administrator, Visual Paradigm repository administrator, Setup Travis.ci service, Create Slack channel for group communication, Writing the ansible scripts & dynamically scale out python script, Design & implement ReSTful back-end system API, Document back-end system API in the swagger, Collect, prepossess and analysis data from AURIN, Write the report, Make demonstration videos on cloud deployment and configureation, Apply Twitter developer account
Xinyao Niu	Whole system containerization and deployments, Total crawler developments, Data pre-processing, Report writing and perform data analysis, Make demonstration videos and prepare for live demo, Apply Twitter developer account
Renjie Meng	Writing the ansible scripts for frontend, backend and crawler Settup the Cluster Couchdb Settup the HTTP API call from the backend to the Couchdb cluster Data-preprocessing and tweet duplication removal Report writing Make demo about ansible, Apply Twitter developer account
Mingyu Su	Writing the report Make the demonstration video of Couchdb utilization Handle the communication between backend and Couchdb, write design docs as well as MapReduce Views, Apply Twitter developer account
Lu Wang	Write the frontend application, Write the report, Schedule zoom meeting

Table 3: Team member contribution

10 Appendix

10.1 Lga code to Greater city conversion

```
GREATER_ADELAIDE_LGA_CODES = [40150, 43650, 40310, 42030, 45680, 40120, 44550,
                                          45340, 44340, 44060, 42600, 40700, 47980,
                                           48410, 40070, 41060, 45290, 40910, 48260
                                          , 46510, 47700, 47140, 45890]
GREATER_MELBOURNE_LGA_CODES = [25340, 21450, 21610, 22170, 22670, 23430, 20910,
                                          22310, 24970, 25900, 26350, 23670, 27450,
                                           27350, 21110, 26980, 24410, 21890, 20660
                                          , 24210, 25710, 27070, 24850, 25620,
                                          24600, 25250, 23270, 24130, 25150, 27260,
                                           24650, 23110, 21180, 23270, 25060, 24330
                                          ]
GREATER_BRISBANE_LGA_CODES = [36580, 35010, 31000, 36250, 34590, 36510, 34580,
                                          33960]
GREATER_SYDNEY_LGA_CODES = [18550, 13100, 14000, 16370, 18000, 14500, 17420, 13800
                                          , 10750, 16350, 10900, 14900, 18400,
                                          16100, 11500, 11450, 17150, 10750, 12850,
                                             13950 , 16250 , 10350 , 16700 , 14100 , 14700
                                            , 18250 , 15950 , 15350 , 15150 , 10200 ,
                                            11520 , 14800 , 17200 , 18500 , 18050 , 16550 ,
                                             11100 , 16650 , 14450 , 14150 , 17100 , 11550
                                            , 11300 , 10150 , 15200]
```

References

[1] UniMelb Research Cloud Documentation infrastructure, R., Services, R., services, O., & Cloud, R. (2020). Research Cloud. Retrieved 28 April 2020, from *https://research.unimelb.edu.au/infrastructure/research-computing-services/services/research-cloud*