**COMP90024 Cluster and Cloud Computing**

**Assignment 2**

**City Analytics on the Cloud**

****

**Team 22**

Xinglin Qiang - 1153086

Shaohua Liu - 1150336

Wentao Hao - 1096215

Kaixin Chen - 1103908

Yichao Xu - 1045184

**Table of Content**

1. Abstract

2. Introduction & Links

3. System Architecture

4. End User Deployment

5. Backend Design

* Melbourne Research Cloud\NeCTAR Cloud Resources
  + Benefits and Issues
  + Resources allocation
* CouchDB
  + Cluster Setup and Database Sharding/Replica
  + Database Design and Error Handling
* Tweets Harvester
  + API and Developer Accounts
  + Implementation and Target Tweets
  + Harvesting Logic

6. Visualization and Scenarios Discussion

* Frontend Design
* User Guide
* Data Visualization

7. Data Analysis

* Correlation Analysis
* Sentiment Analysis
* Hashtag analysis

8. Conclusion

9. Work Sharing

10. Reference

**1. Abstract**



The extensive appliance of social media makes people's information more transparent. The data that reflects people's daily habits, opinions and behaviour characteristics is approaching us on an unprecedented scale. This also means that we can no longer rely on traditional analysis methods and tools in our research. This paper will conduct an experiment to analyse the changes of Australian tweets since 2014.



**2. Introduction**

The amount of social media data has been increasing exponentially over the last decade, there are approximately 4.6 million users just in Australia alone, and the number of new tweets can go up to hundreds of millions per month. This provides researchers an opportunity to utilize these data to conduct a series of analysis in terms of sentiment, income and age upon Australian people as each tweet contains various information such as geographical location. However, in order to support the analysis on such an enormous amount of data, it is essential to develop a cloud solution architecture that utilizes the services provided by the National eResearch Collaboration Tools and Resources (NeCTAR) research cloud that follows OpenStack protocol.

In this paper, tweets data are collected via Twitter API, they will be processed and analysed for the comparison purpose with the data from Australian Urban Intelligence Network (AURIN). Multiple virtual machines with 250GB storage in total are used to harvest, store and analysis data, as well as to deploy the entire system. We will mainly focus on four scenarios: whether the age has correlation with tweeting frequency; whether mean total income has correlation with tweeting frequency, in general; sentiment analysis on tweets. Also, we analyse the characteristics of Twitter users. The overview of system architecture along with the detailed implementation will also be documented in this paper, and we will summarize the advantages and disadvantages of NeCTAR when comparing it with HPC Spartan that we have used in Assignment 1. We will also document the user guide of our system in the very end.

**The links for our Project:**

All the files, scripts, and programs of the system are stored on GitHub repository located at<https://github.com/Taylorrrr/COMP90024-2020S1-Team22/tree/master>.

The video of demonstration:

Ansible Deployment: [https://www.youtube.com/watch?v=\_uzXLX3cHwQ](https://www.youtube.com/watch?v=_uzXLX3cHwQ%20Frontend%20Presentation:%20https://www.youtube.com/watch?v=U7CeZP5M2pg)

Frontend Presentation: <https://www.youtube.com/watch?v=U7CeZP5M2pg>

Frontend: <http://172.26.132.92/>

Backend API: <http://172.26.132.92>:5000

**3. System Architecture**

A close up of a map

Description automatically generated

Figure 1: system architecture

**4. End User Deployment**

$ cd ansible

$ ./launch-instance.sh

$ ./install-dependency.sh

$ ./deploy-software.sh

Then access the webpage through <http://172.26.132.92>

**5. Backend Design**

The rapidly growing data requires the system to support scalability, extensibility and parallel computability. Therefore, the document-based NoSQL database CouchDB was utilized. CouchDB can be easily set up as a cluster which satisfies the requirements of scalability and extensibility. Furthermore, it automatically adds a revision field ‘\_rev’ for every document stored in it to make sure the consistency between high frequency parallelized read and write operations. It also supports MapReduce (a computation model which is designed to handle big data) as a built-in function to build views of data for further retrieving or analysing.

The development of the system is separated into frontend, backend, harvester and sentiment analysis. Their communication follows the API specified bySwagger (<http://172.26.132.92>:5000) through HTTP requests. The backend is mainly responsible for instance monitoring, RESTful data storing/retrieving, view results accessing and task queue for harvesters. Harvesters will keep fetching tasks from the backend and then using the developer tokens and twitter RESTful endpoints to search relevant tweets and finally persisting into the database cluster.

**5.1 Melbourne Research Cloud (MRC)**

The Melbourne Research Cloud offers free on-demand computing resources to researchers at the University of Melbourne. It provides similar functionality to commercial cloud providers such as Amazon Web Services, Microsoft Azure and Google Cloud Platform.

**5.1.1 - Benefits and Issues**

**Benefits of MRC**

* MRC supports the openstack cloud computing platform which is deployed as infrastructure-as-a-service (IaaS). It provides virtual servers and other resources (like volumes, floating IPs) to users instantly. Users can manipulate the resources in a scripting way instead of manually which could save a lot of time during the cluster scaling.
* Extremely rich options for image selection during instance setup.
* Users in a project (team) can access the team space simultaneously with the same priority.
* Fully functional security groups managing system, which supports both IP prefix and group-wise internal communication.
* Users can create snapshots for their resources for backup regularly.

**Issues of MRC**

* The usage of resources is not real-time updated. For example, the usage of CPUs and instances are not immediately updated when we delete an instance. The delay is about less than 5 minutes.
* The waiting time for creating a snapshot can be crazy. It always took more than 3 hours in the queue for creating a single snapshot.
* Too many images have similar names which made users a little confused during the image selection.
* Potential data losses if the instances accidentally shut down. Backup is always an important issue.

**5.1.2 Resources allocation**

**(1) Instances and Volume Allocation**

The groups are allocated with 4 instances, 8 virtual CPUs, 36 GB memory and 250 GB volume storage in total.

We assign 3 instances (6 CPUs with 27 GB memory) as the hosts of the database cluster and the last instance as the host of backend and frontend. The harvesters are deployed among all the 4 instances. Each of the database instances is allocated 80 GB storage which results 240 GB in total. And the rest 10 GB is assigned to the web server instance.

**(2) Instance Image and Security Group**

We choose Ubuntu 18.04 (without docker) image, qh2-uom-internal network for all the instances and enable port 22 (ssh), 7000 (frp NAT-traversal reverse proxy) by default. And for database servers, we add them into a new security group and enable port 4369, 5984 within that specific security group so that they can communicate with each other. For the web server, we enable port 5000 (Flask backend) additionally and frontend http port 80.

**Note**: The reason why we deployed NAT-traversal is because some of our teammates are currently staying in China. And the connection via Melbourne VPN is too slow to develop.

**(3) Proxy Settings**

Due to the lack of public IPv4 addresses in MRC, choosing an internal network provider is mandatory. We need to set proxies to access the public network. The proxy settings for system, docker and git are followed by the instructions given in the discussion of LMS.

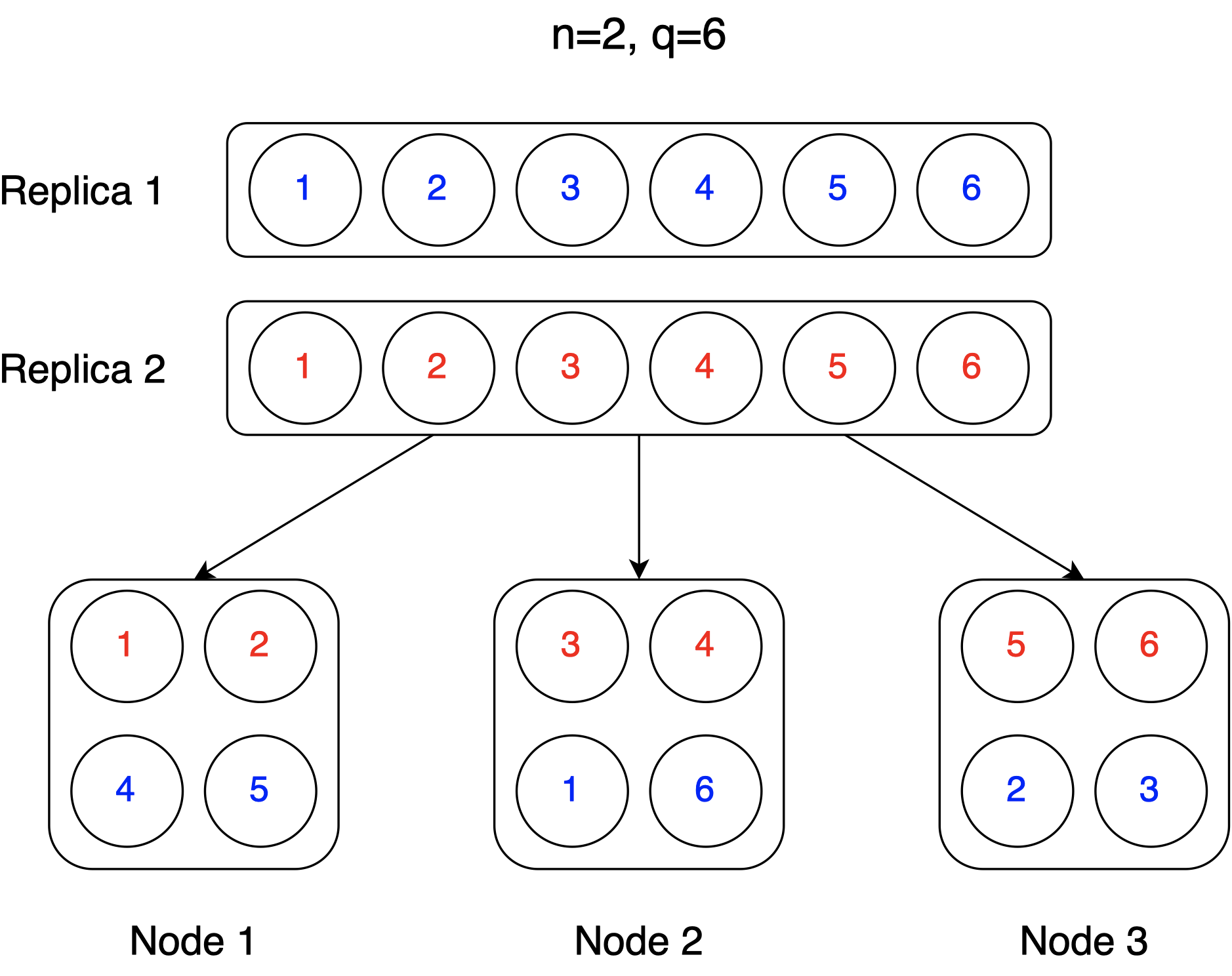
**5.2 CouchDB**

CouchDB is an open-source document-oriented NoSQL database. It provides ACID semantics, multi-version concurrency control, distributed architecture, JSON-based document storage, MapReduce views and rich HTTP API.

**5.2.1 Cluster Setup and Database Sharding/Replica**

We use container technology docker to deploy the database cluster and bind mounted volume to persist the data. Official Apache image is chosen rather than the version of IBM since the full-text search plugin is not necessary for our system. The cluster setup script can be found in the templates folder of roles ‘couchdb’ in ansible. During the steps of cluster setup, we encounter #2797 [bug](https://github.com/apache/couchdb/issues/2797) (Janl, 2020) which cannot successfully finish the entire process without invoking a HTTP GET request to the database index page first.

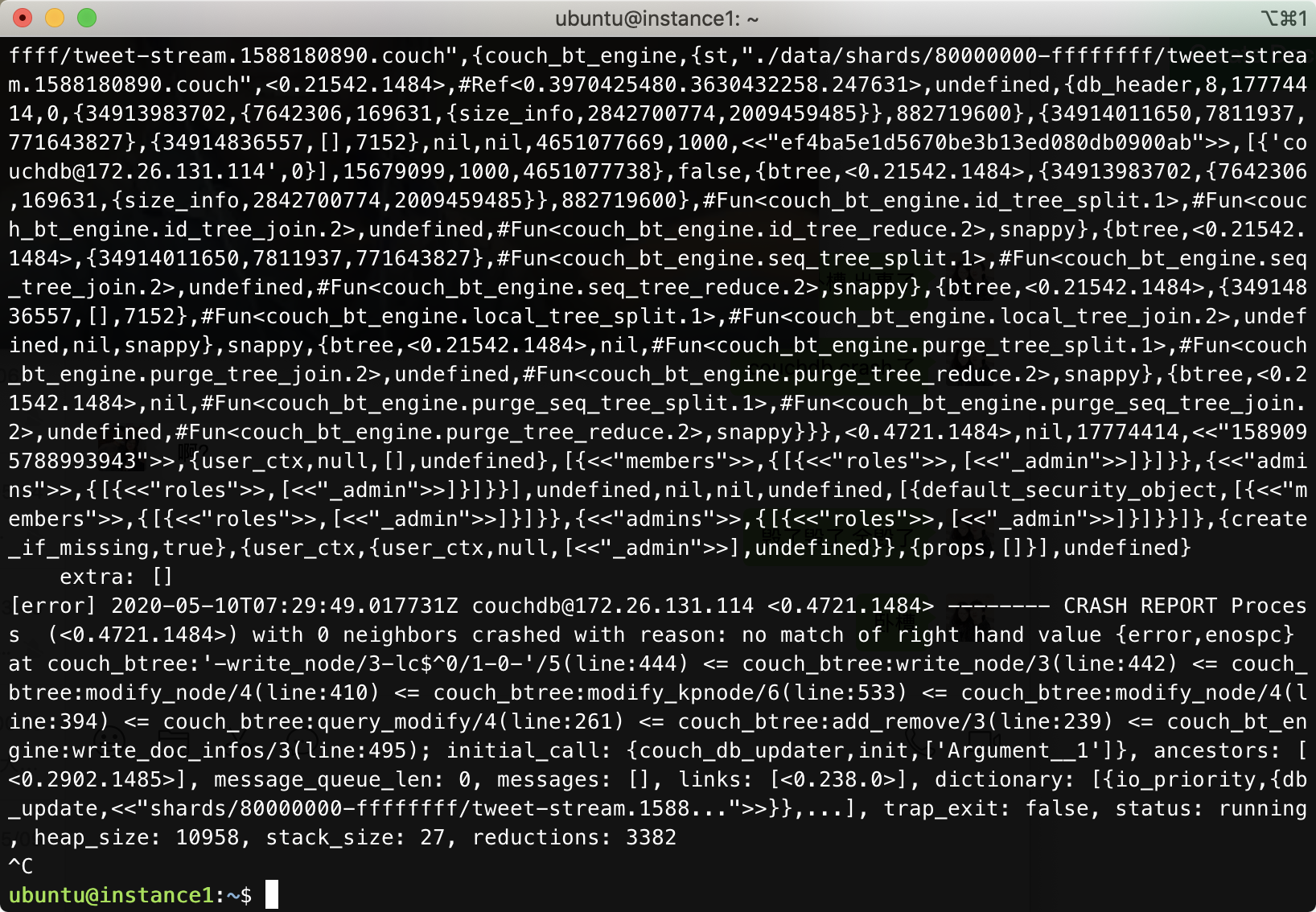
The sharding strategy for the databases in the cluster are q=6 shards and n=2 replicas which means totally 12 shards evenly distributed among the three instances. Splitting the databases to shards can efficiently boost the MapReduce procedure due to the parallelization.

Figure 2: cluster setup

**5.2.2 Database Design and Error Handling**

During the development, we stored all of the data including tweets, tweet users and relevant MapReduce views in the same database at first. Then after a few days, our database crashed due to the rapidly growing data and the bad design. Meanwhile, the entire 240 GB volume was flooded by the crash reports shown in the figure.

Then, we realized that although CouchDB can store any JSON document in the same database, the views are still updated during irrelevant documents added, removed or updated. Therefore, the documents should be grouped by their data types to different databases (just like ‘tables’ in RDBMS) for better performance. In this project, we construct two databases named ‘tweets’ and ‘users’. Details will be discussed in the following section.

Figure 3: crash reports for CouchDB

**5.3 Tweets Harvester**

**5.3.1 API and Developer Accounts**

Twitter provides two types of API for gathering tweets — RESTful with rate limits and Streaming without rate limits. Twitter APIs are only available to authorized developers now. Unfortunately, only two applications of developer accounts in our group were approved. Therefore, the efficiency of the harvester might be slightly slower than other groups.

**5.3.2 Implementation and Target Tweets**

To access the APIs, we create python programs using ‘tweepy’ library.

As mentioned above, the database crashed once due to the enormous data harvested. We decided to partition the ‘tweets’ database by user id and shrink the search scope to the tweets with geo-location enabled so that it could easily classify the Statistical Area Code (SA3) to make further analysis with AURIN spatial aggregated data.

**Caveats**:

* Since only the tweets with geo enabled are persisted, they are relatively correlated with topics about travels and scenic spots. Furthermore, we cannot distinguish whether the tweets were posted by travellers which might corrupt the analysed result against AURIN.
* Some of the geo coordinates are distributed along the coastline, which cannot be classified the SA Code accurately.

**5.3.3 Harvesting Logic**

**(1) Streaming API**

Streaming API utilized publisher-subscriber pattern which can provide developers a persistent HTTP connection to the Twitter server, which is a real-time delivery based on the custom constraints. For example, we can apply the location filter with the country bounding box to harvest all fresh tweets in Australia. But it’s also worth noting that the logical relationship between constraints is logical ‘or’ rather than logical ‘and’, which means it’s better not setting another constraint for solely geo-enabled tweets harvesting. After getting the raw tweet, with the help of tweet id we can efficiently filter out duplicate tweets and then cut off some useless fields and finally pass the tweet JSON object to the backend for data persistence. For those tweets with ‘retweets’ and ‘quotes’, we need to recursively apply the persisting logic mentioned above.

**(2) User Model and RESTful API**

Although the streaming API does not have rate limitation, only few of the tweets would meet the requirement (with geo-enabled). Therefore, we must explore more relevant history tweets based on these small fractions of active users. These users are persisted in another database named ‘users’ with additional fields ‘Searched’, ’Expanded’ and ‘Level’.

* Searched (boolean): Whether the user has been searched by GET statuses/user\_timeline endpoint. (Rate Limitation: 900 requests / 15 minutes / user auth token, return no more than 200 tweets per request)
* Expanded (boolean): Whether the user has been explored by GET friends/ids and GET followers/ids endpoints. (Both with Rate Limitations: 15 requests / 15 minutes / user auth token, return no more than 5000 user ids per request)
* Level (integer): The priority of the user in the BFS search tree.

The theoretical maximum throughput is 360,000 tweets and 300,000 users per 15 minutes.

We could use field ‘level’ as the emitted key for the ‘Map’ function. Then the view can be used as a priority queue due to CouchDB sorting all the entries in the view by key ascending order.

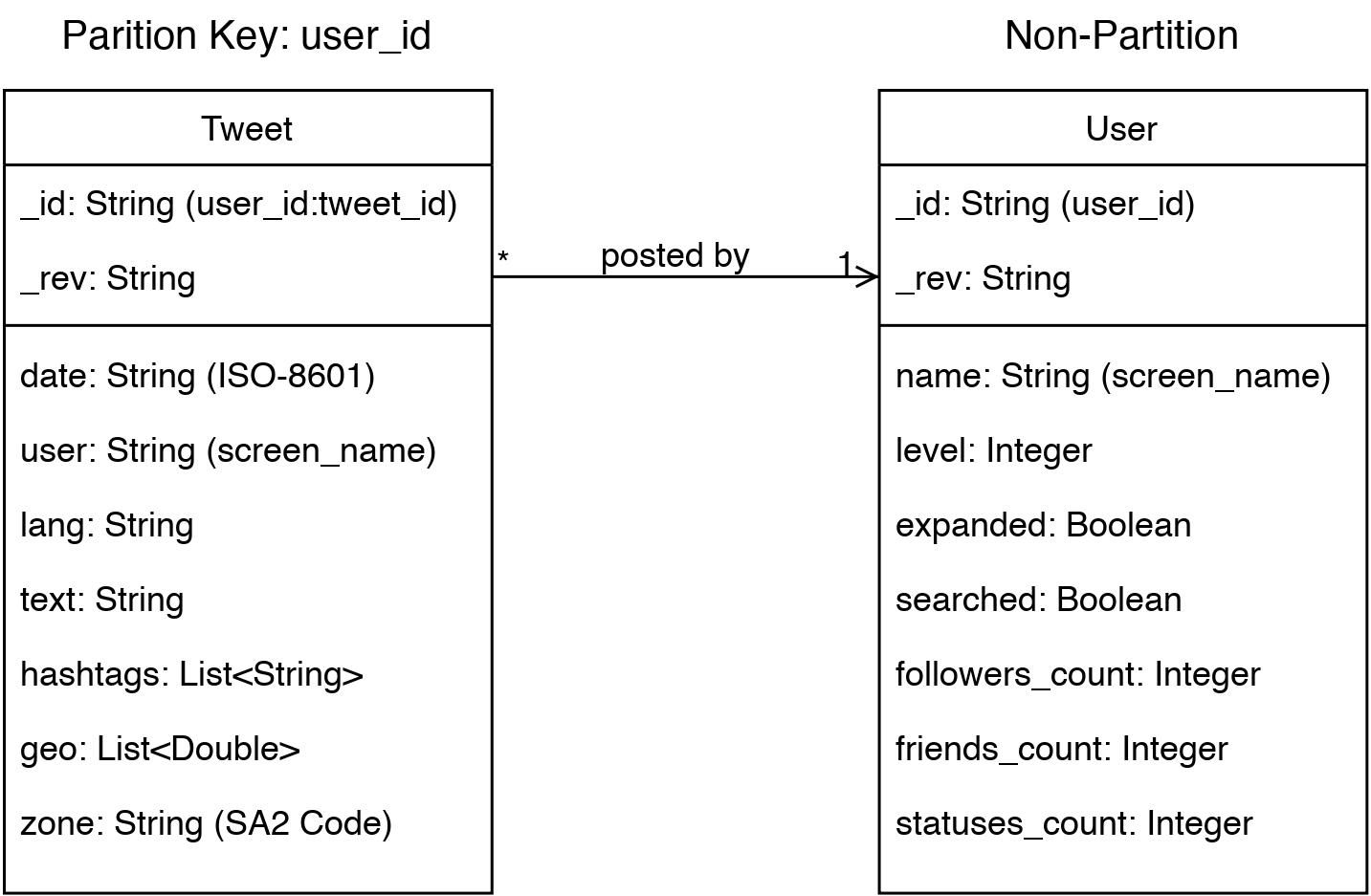
**UML**

Figure 4: tweet & user table

**(3) BFS Searching Algorithm and Task Queue**

The harvester modules are designed to be scalable which means the system can be easily scaled up when we have more developer tokens. Therefore, two simple message task queues implemented with Redis were deployed in the backend server for centralizing and caching the user priority queue. One for searching historical tweets and another for exploring friends and followers’ relationships. Both of the task queues will keep fetching un-processed users from the database if the volume is less than the predefined threshold and then mark the states of fetched users as in ‘queue’. The harvester workers distributed among the instances will get the task user from queue as input and then perform following behaviours:

* Exploring Task: Using RESTful API to retrieve ids of friends and followers of the user. Mark the level of these new ids to the level of the source user + 1 to denote lower priority in the BFS searching.
* Searching Task: Using RESTful API together with Python generator to progressively search the historical tweets of the user. Persist the tweets with geo-location enabled and keep calculating tweets geo rate, abort the searching task if necessary (rate < 1% for first 400 tweets) or promote the user to high priority (maximum rate > 5%) by marking its level to 0.

**6. Visualization and Scenario Discussion**

**6.1 Front-end Design**

The front-end web application powered by Vue.js is implemented based on the template provided by “Vue Now UI Kit”, it consists of four pages – Home, Login, Team and Maps, whereas the Maps page contains all the functionalities.

1. Team page

This page shows the team that contributed in this system, also indicating each member’s role, as shown in Figure 5.

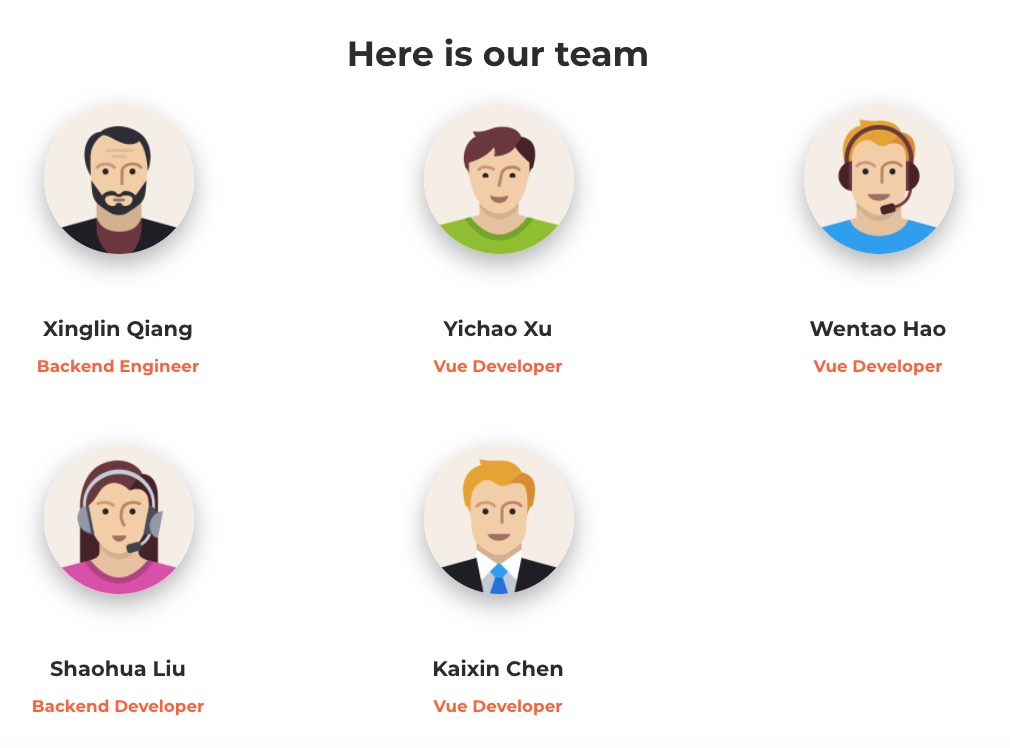


Figure 5: team page

A close up of a map

Description automatically generated

Figure 6: map page

2. Maps page

Since this project is dedicated to tweets in Australia, therefore the world map is limited to Australia only, as shown in Figure 6. This page lists all the core functions of the system, including AURIN data previewing, AURIN data detailed chart showcasing. First time opening this page might take a while, as the system is fetching a large amount of data from the CouchDB database.

The elements that appear on this page can be summarized as – Google Maps API, pie chart, line chart, bar chart, radar chart, and mixed chart, button, and pop up card. The display of the Google Map is set up according to the Google Developer documentation by acquiring the API key from Google Cloud Platform Console; a series of charts are used to visualize all the data fetched from the back-end; the button on the left side of the page will expand a menu containing AURIN data selection, region detailed visualization by SA3 code upon on click (see Figure 7); mouse action includes left click and hover over, when hover over on an area, a card will pop up showing the SA code, area name, average income, number of Twitter users, and the age distribution of the selected area (see Figure 8); when left click, it will add a marker on the map.

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

Figure 7: function menu Figure 8: pop up card

When dealing with displaying data on the map, the Google Maps Data Layer is used as a container for arbitrary geospatial data to display GeoJSON data. GeoJSON is a standard for geospatial data on the internet. The Data class follows the structure of GeoJSON in its data representation and makes it easy to display GeoJSON data. Use the *loadGeoJson()* method to easily import GeoJSON data and display points, line-strings and polygons.

The Display Options menu on the left is responsible for showcasing the detailed data below the map element. It supports search by SA3 code function and is able to input up to 10 SA3 code at once, and provides four different types of charts to show the result (see Figure 9, 19, 11, 12).

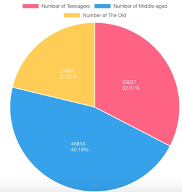
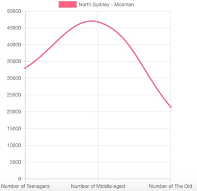
 

Figure 9: pie chart Figure 10: line chart

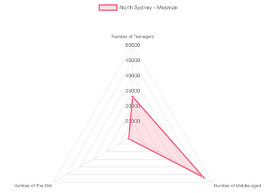
 

Figure 11: bar chart Figure 12: radar chart

Meanwhile, the system supports 6 types of AURIN data display, which are age, salary, other languages, income, population, and tweets density respectively.

**6.2 User Guide**

In this section, a brief overview and basic guide are given to manipulate the system.

In order to start using the system after accessing the webpage, click *All Menus*, then select *Maps*, the user will be welcomed with the Google Maps specified in Australia.

Users can add a marker on the map by left clicking the mouse and view the summary information by hovering over a certain area (Melbourne and Sydney only).

The *Display Options* button located on the left side of the page gives users the ability to view different data distribution such as salary, age, languages by changing the options listed in *AURIN Data* dropdown button.

If the user wants to have a different view of data of a certain area, simply type in the area’s SA code, a sample SA code table is shown in Figure 13. After hitting *Go*, a new section will appear below the map, showing the data representation based on which types of charts the user selects to display.



Figure 13: SA table

**6.3 Data Visualization**

Data visualization plays a vital role in the system, in this process, we fetch the data encoded in JSON format from CouchDB via HTTP GET request, different colour scheme is used to render different objects (age, income, population).

**7 Data Analysis**

In this section, we will do some statistical analysis on the data collected from Twitter harvester and Australian Urban Research Infrastructure Network (AURIN) including correlation and sentiment analysis.

**Dataset**

The dataset consists of two parts: Twitter data and AURIN data. The former dataset, which is collected in the way described in the Twitter Deep Digging section by the Twitter harvesters built on 4 instances of the UniMelb Research Cloud, contains more than 4 million tweets posted in Australia. The Twitter data which involves amounts of emotional messages reflecting the senders’ current feelings is a good sample data for sentiment analysis and can represent the emotions of local residents. While the latter dataset, which is exported from AURIN portal, comprises the estimated annual personal income data and the population information by age and sex in Sydney and Melbourne published by Australian Bureau of Statistics (ABS) within the scope of Statistical Area Level 3 (SA3). Although both data are not collected in 2020 and may not be the latest dataset (the population data is counted in 2017 and the income data is counted in 2016), they can in some ways demonstrate the current situation of regional income and population due to this kind of “big” data does not change much in a few years. Also, we introduce Tourism Expenditure (Tourism Regions) published by Tourism Research Australia (TRA) in 2015. The AURIN dataset provides a big picture of realistic conditions of the regions whereas the Twitter dataset reveals the true feeling of each person, and we can do some practically meaningful analysis by combining both datasets.

**7.1 Tweeting Frequency Analysis**

Social media has completely changed the way that people social, communicate and work in the past decade, it is reported that only 5% of adults in the United States used social media platforms in 2005, whereas in contrast the 70% in 2019 (Allen, 2019). When it comes to teens, university researchers indicated that a large portion of high school seniors spend less an hour a day in face-to-face social interaction, instead, more students prefer to stay online and live online, many of them have been found using Facebook more than 8 hours a day (Allen, 2019). Therefore, according to this phenomenon, we’d like to find out whether tweeting frequency has correlation with age, here we only take Melbourne into consideration.

**7.1.1 Analysis**

To find the correlation between teen ratio and tweeting frequency, we need to take three steps:

1. Retrieve the data that contains the number of teens and total tweets number of a specific region.

2. Divide teen number by total population of a specific region to calculate the teen ratio.

3. Plot teen ratio and total tweets number to observe the final result.

However, the row data that is used to plot does not give us a distinct observation (see Figure 14), so we process the data further by removing the 3 outliers and reducing the magnitude of the number of tweets by 10,000.

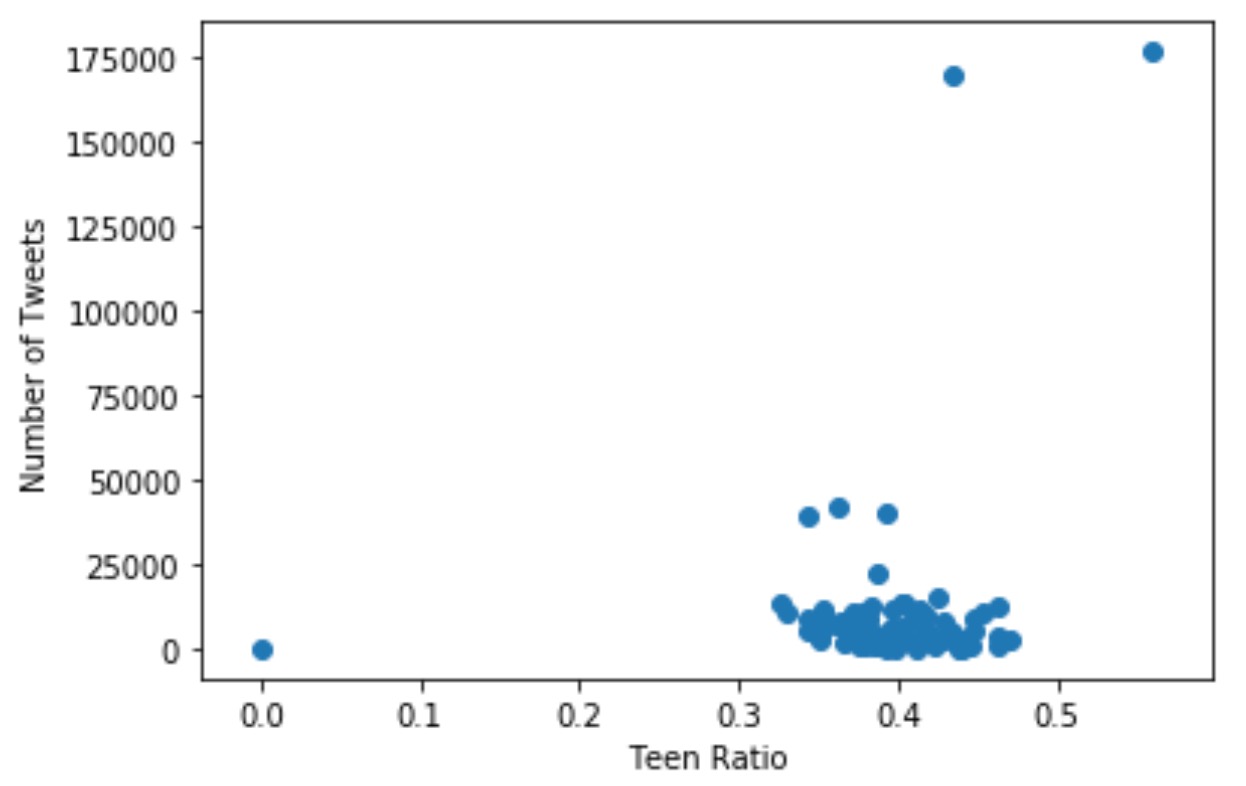


Figure 14: Teen Ratio and TweetNum

**7.1.2 Processing AURIN Data**

As aforementioned above, we downloaded the population by age and sex data and Income data of Greater Melbourne and Greater Sydney within SA3 scope from AURIN portal. For population data, we summed up the population into different groups including the teen (aged 0~30 years), the middle-aged (aged 30~60 years) and the old (aged above 60 years) and calculated the teen ratio which is the percentage of the teens taken up in the total population.

**7.1.3 Results**

After cleaning the data, we draw a scatter diagram between the teen ratio (x axis) and the number of tweets (y axis), the final result is shown in Figure 15. As we can observe, the majority of the tweets number is lied between 1000 to 12,000, 3 very high tweets number data all appear on the left side of the middle point of teen ratio, and the increase of teen ratio does not have much effect on number of tweets in every region in Melbourne, it even comes with a -0.26 correlation coefficient.

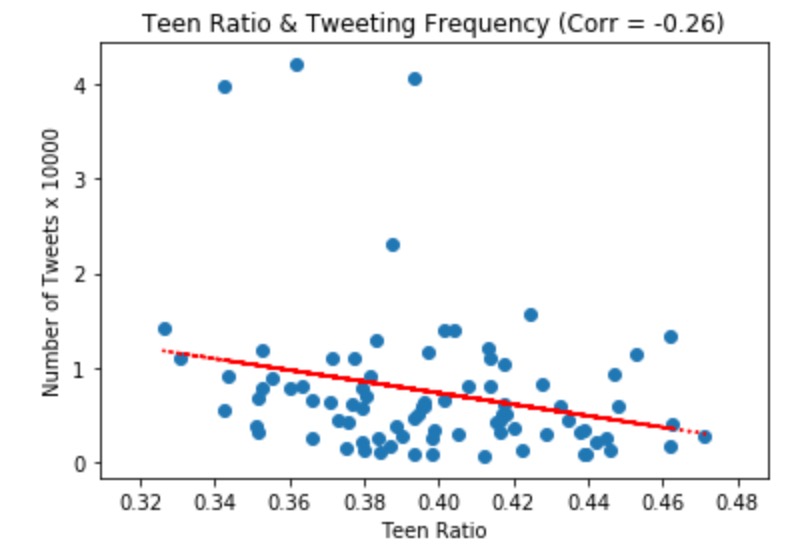


Figure 15: Teen Ratio and TweetNum (modified)

**7.1.4 Summary**

From the above analysis, we can conclude that the teens might not be the major group for Twitter, however, since the data is for Melbourne only, this result will possibly be different if we extend our analysis to the whole Australia.

**7.2 Sentiment Analysis**

Sentiment Analysis, or Opinion Mining, which tries to identify emotions and opinions within a given text, is a sub-field of Natural Language Processing (NLP). The goal of sentiment analysis is to measure sentiments, evaluations, attitudes and emotions of the message sender based on the computational treatment of subjectivity in a text. Here we do the sentiment analysis on each tweet text to figure out the sender’s emotions hidden in the tweet and use the results to build the sender’s satisfaction degree about the region he/she was currently located. We have two hypotheses here:

1) Tweet senders’ satisfaction degree may be affected by the mean income of the region because people in rich regions may feel happier than those in relatively poor regions.

2) Tweet senders’ satisfaction degree could be influenced by the age distribution of the region due to that regions with more young people may be more living and fuller of vigor.

We will further examine both hypotheses in our sentiment analysis.

**7.2.1 Analysis Method**

**(1) Limitation and Error in Lexicon Methods**

There are plenty of sentiment analysis methods in NLP and the mainstream methods are based on lexicons. However, most analysis based on lexicons need to spend much time in pre-processing the text including tokenizing the text into a few chunks or words, converting the words to their canonical forms and removing the noise (e.g. stop words and punctuations) in the text. It is very time consuming when you apply the cleaning process to handle thousands of tweets. Apart from that, typical social media text will contain quite a lot of emojis, such as “:)” and “:(”, and capitalized words strengthening the senders’ emotions which will be cleaned by the traditional methods in the cleaning process. Also, sometimes some negative words can overwhelm the positive words, and if we treat both with the same weight in traditional methods, we will misunderstand the true meaning of the text.

**(2) VADER and Error Handling**

Due to the reasons described above, after comparison we decided to use Valence Aware Dictionary and sEntiment Reasoner (VADER) as our main sentiment analysis method. VADER is a famous lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. Like other methods based on lexicons of sentiment-related words, VADER identifies each word of the text as to whether it is positive or negative according to the lexicon. But the difference is that each word in VADER lexicon has an actual rating score, for example “good” is 1.9 and “nice” is 1.8, and that means each word is treated with different weight in VADER. VADER will capture each word’s contribution in the text and produce a final metric - the compound score, which has been standardized to range between -1 and 1, to measure the sentiment in the text. Besides, another advantage of VADER is that emoticons and capitalized words, which are commonly used in tweets, in the text are taken into account. And last but not least, VADER has extremely fast processing speed which best fits for our big tweet data.

**7.2.2 Analysis Steps**

Our analysis on sentiment is mainly composed of three parts: processing AURIN data, using VADER to extract emotions from tweets and finally comparing AURIN data with the emotion results.

**(1) Processing AURIN Data**

As we mentioned above, we downloaded the population by age and sex data and Income data of Greater Melbourne and Greater Sydney within SA3 scope from AURIN portal. As for income data, we extracted “Estimates Of Personal Income Mean Total Income (Excl. Government Pensions) $” to represent regional annual mean income.

**(2) Doing Sentiment Analysis on Tweets**

Firstly, we used the VADER method to identify each tweet’s emotion and label it as “Positive” if the compound score is larger than 0.2, “Neutral” if the score falls between 0 and 0.2 and “Negative” if the score is less than 0. Here we only do the analysis on tweets in English for high accuracy. Then, we counted the number of each category of tweets in each SA3 region in Greater Sydney and Greater Melbourne, which is corresponding to the geographical scope in AURIN data. And in the end, we compute the satisfactory degree or positive index of each region by making the number of positive tweets divided by the total number of tweets.

**(3) Comparison between Sentiment Results and AURIN Data**

Finally, we combined the sentiment results obtained from section 2.2.2 with the processed AURIN data in section 2.2.1 to make a complete table of the whole data and plotted scatter diagrams and trend lines to do correlation analysis between variables.

**7.2.3 Results**

**(1) Income and Satisfactory Degree**

We drew a scatter diagram between the annual mean income of each region (x axis) to the satisfactory degree of the region (y axis) and obtained the Figure 16. As we can see, there are four clusters in annual income: less than $70,000, $70,000~90,000, $90,000~120,000 and above 12,0000, and the number of regions with the satisfactory degree below 0.4 in each income slot are respectively 9, 2, 1 and 0. Approximately, the satisfactory degree tends to be higher in the region with higher annual income (correlation coefficient = 0.1379) and that verifies our hypothesis one. Since people living in regions with high income may not have to worry about lots of things, most of tweets in these regions appear to be positive.

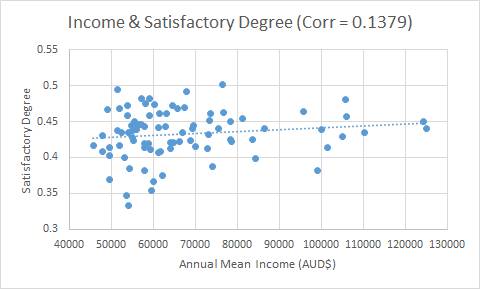


Figure 16. Annual mean income and satisfactory degree

**(2) Teen Ratio and Satisfactory Degree**

Then we drew a scatter diagram between the percentage of the teenagers taking up in the total population in each region (x axis) to the satisfactory degree of the region (y axis) and obtained the Figure 17. Out of our expectation, apparently the satisfactory degree tends to decrease with the increase in the teen ratio in each region (correlation coefficient = -0.1732). A few regions with the satisfactory degree lower than 0.4 appear with their teen ratios exceeding 0.38. We reckoned the reason why this happened is that many teenagers are currently fighting for their lives, they are facing pressure from their families, their work and the society and they are right in the stage to strive for success. Therefore, it is common for the teenagers to make some complaints on tweets. In addition to above, we also plotted a scatter diagram (Figure 18.) between the annual mean income of each region (x axis) to the number of the teenagers in each region (y axis) and found that the income is negatively related to the number of teens. That means most teenagers actually are living in regions with relatively low income and they are striving for better lives.

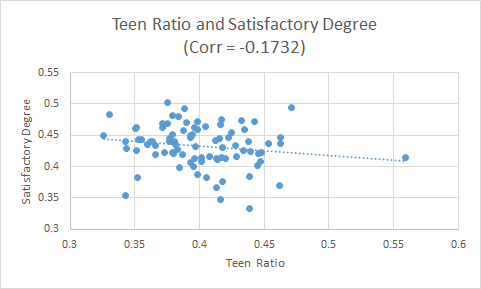


Figure 17. Teen ratio and satisfactory degree

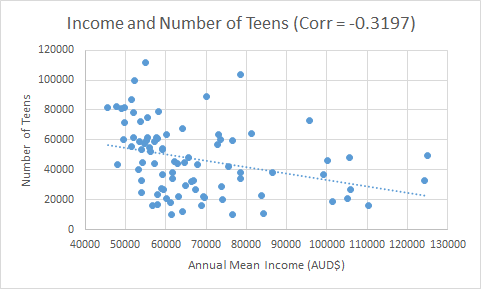


Figure 18. Annual mean income and number of teens

**7.2.4 Summary**

Through the sentiment analysis and comparison above, we found that the income of the region can slightly and positively affect the sentiments of the tweets in each region and the teen ratio in each region tends to negatively influence the sentiments of the tweets. And via further research on the relation between the number of teens and annual mean income in each region, we discovered that most teenagers are living in the regions with relatively low income and considered that they are still struggling for their lives. Therefore, we believe that in order to improve the satisfactory degree in each region, the local government should not only concentrate on boosting the regional economic development, but also focus on improving the treatment of the teenagers and try to provide them with opportunities for social engagement and personal development.

**7.3 Hashtag Analysis**

As we aim to figure out the correlation between regions and their topics, we only work on tweets with exact geolocation. The region is divided by Statistical Area 3 (SA3) and all tweets with exact geolocation will be distributed into their corresponding areas. Due to the independent tweets content and large scale of tweet dataset, we decide to take advantage of high-performance computing (HPC) techniques. HPC can increase computation efficiency by deploying one task to several multi-core servers. Instead of a single thread computation, a task will be divided into several independent workloads and can be aggregated into a single final result.

**7.3.1 Trending Hashtag Analysis**

One simple strategy to quickly understand the main idea of tweets is to analyse their hashtags. Most people only put hashtags when it is necessary and worth emphasizing. Therefore, we can obtain trending topics by simply counting frequent hashtags. It is important to note that counting hashtags can also be done by applying map-reduce feature in CouchDB, where we store all harvested tweets; however, it is more efficient to utilize the University of Melbourne HPC facility SPARTAN instead of CouchDB servers, which contribute the majority computation power to harvesting and making views for front end.

**7.3.1.1 Pre-process**

To identify the users who follows trending topics, we need two passes of searching:

1. Traverse all tweets to identify trending topics
2. Identify users who used hashtags relating to trending topics

Due to the large scale of the dataset, we decide to process tweet data line by line to avoid buffer overflow. Each tweet will first be parsed into a json object and only hashtags will be extracted from tweet json. 30 most frequently used hashtags will be recorded to help further analysis. The application is deployed on SPARTAN with eight cores on two nodes. Each node only works on some proportion of the whole work, which is achieved by a line counting trick. As each node has a unique rank r, we can ask each core only to process the i-th tweets where so that the dataset is divided into eight equal workloads.

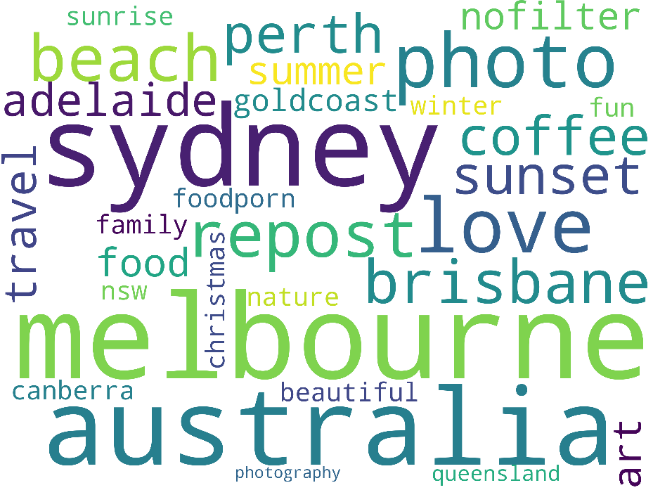
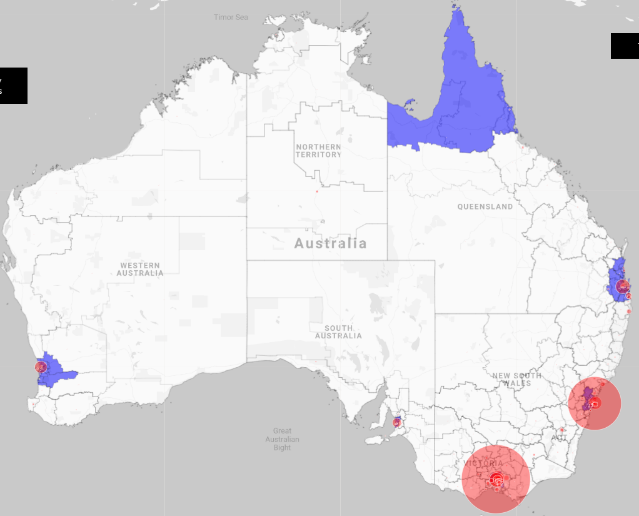
 

Figure 19: dataset text Figure 20: data distribution

**7.3.1.2 Result**

As can be seen in the following word cloud picture (Figure 1), most hashtags are relevant to travel and cities, such as Melbourne, beach, photograph etc. Also, among 116963 users, we identified 30983 users (about 26%) who like to share their feelings about travel and cities. The percentage could be significantly higher if we apply more advanced techniques on the whole text of tweets. It is reasonable to have such a high percentage of travel and photo tweets in our dataset since people want to let friends know when they arrive at new locations and see new things.

After introducing data from Tourism Expenditure (Tourism Regions) published by Tourism Research Australia (TRA) in 2015, we believe the twitter data we harvested is a small social media society of travel lovers. Those Top 30 hashtags can cover all Top 10 tourism expenditure regions (blue regions on map) sorted by tourism expenditure from high to low. Only North Queensland has not shown a great number of tweets, but we expect there will be rapid increase in the following years. I will refer ‘travellers’ to users who like sharing travel experience later in this report and ‘non-traveller’ to those who do not. Also, ‘travel-related tweets’ is used to describe tweets containing at least one of Top 30 hashtags.



Figure 21: tourism expenditure

**7.3.1.3 Further analysis**

To further prove our statement, we want to eliminate the factors from income and population. One common heuristic is that the number of tweets mainly depends on the population and the income of the region. We introduce population and income data of every SA3 area in AURIN. Also, the number of travel-related tweets in that area, which contains Top 30 hashtags, will be considered as well. Therefore, every SA3 area compares its total number of tweets (harvested online) with population, income (AURIN data) and travel-related tweets (subset of total number of tweets), The correlation r estimates the strength of the linear relationship between two variables.

When correlation r (x, y) = 1, x has direct positive influence on y. A correlation of 0 means no linear relationship between two variables. Given the table below, we can tell there is little correlation between income and the number of tweets in the region and moderate correlation between population and the number of tweets. For example, both Dandenong and Playford have mean annual income around 45,000; whereas the number of tweets in Dandenong (21,102) is significantly higher than those in Playford (4,133). However, when it comes to travel tweets, there is almost a perfect linear relationship.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | total tweets | Travel-related tweets | population | income |
| total tweets | 1 |  |  |  |
| Travel-related tweets | 0.986 | 1 |  |  |
| population | 0.467 | 0.456 | 1 |  |
| income | 0.224 | 0.225 | 0.502 | 1 |

**7.3.2 Sentiment Analysis between travellers and non-travellers**

It is interesting to conduct sentiment analysis on both traveller and non-traveller to observe the difference between two groups of people. Experiments are conducted to answer two different scenarios.

1. Whether travelers are more satisfied with their lives than non-travelers
2. Whether travel-related tweets improve satisfaction than other tweets

**7.3.2.1 Satisfaction analysis**

Satisfaction of travellers (ST) can be easily computed by feeding all tweets published by travellers into Vader analyser. Similarly, the satisfaction of travel-related tweets (STF) is measured by feeding all travel-related tweets into the analyser. Both satisfaction scores will be normalized by their number of tweets.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Non-traveller satisfaction | Traveller satisfaction (including TF) | Travel-related tweets satisfaction (TF) |
| Satisfaction | 0.178 | 0.214 | 0.163 |
| Number of tweets | 1139986 | 2621175 | 360486 |
| % total number of tweets | 30.30% | 69.70% | 9.58% |

As shown in the above table, it is not surprising that there is a considerable improvement (20.2%) in traveller satisfaction (0.214) than non-traveller satisfaction (0.178) since people enjoy travelling and sharing new experiences. Recall that travellers are only 26% of the whole twitter users in our dataset, whereas they contribute to almost 70% tweets of the total number of tweets. It may also indicate travellers are more productive than non-travellers. However, when it comes to the satisfaction of travel-related tweets, it is surprisingly lower than travellers’ general tweets, which means that travel is not the key aspect to improve their satisfaction. The satisfaction of travel-related tweets is even lower than the satisfaction of non-travellers’ tweets. One possible reason for this may duo to the fact that travellers are not actually “travel lovers” and travelling is just a routine of their lives like drinking coffee.

**7.3.2.2 Challenges and limitations**

One issue when collecting hashtags to know the current trending topics is that some hashtags are only used by few Twitter users, whereas those hashtags become trending topics due to extremely productive users. To minimize the impact of productive users, we build user records for counting the number of hashtags they used. When one user shows a significant hashtag usage, only some proportion of hashtags will be counted in trending topics and every user will at most contribute 50 hashtags. Also, even though we have already kept the geolocation of tweets, the cities on tweets do not match our AURIN data standard (SA3), which makes it difficult to make comparison between harvested data and AURIN data. we build a classifier to label tweets with SA3 depending on their geolocations.

Sarcasm has become a hot topic in natural language processing (NLP). It is even a difficult task for humans to determine. Our hashtag analysis does not take account for sarcasm since if we assume sarcasm has uniform distribution among all tweets, even though the satisfaction is higher than actual, the relationship between travellers and non-travellers will remain the same. However, it is still a major source that makes inaccurate results.

**7.3.3 Analyse satisfaction gap between travellers and non-travellers**

The aim of this section is to detect what are major factors that produce a satisfaction gap between travellers and non-travellers. It is important to note that travel-related tweets are only approximately 14% (9.58/69.7) of travellers’ whole tweets. The major factors should be hidden in the hashtags of the other part of the non-travel-related tweets (about 86%). Therefore, we invent new criteria: hashtag satisfaction. Every tweet satisfaction will contribute to all its hashtags. For example, if , all hashtag satisfaction of hashtags of will increase by 0.3. The Hashtag satisfaction of hashtag h will be finally normalized by the number of occurrences of h and it measures the mean satisfaction value whenever twitter users use this hashtag h. In order to maintain generalization, we only consider hashtags when they occur more than 1000 times. In this section, we compute hashtag satisfaction of both travellers and non-travellers.

A picture containing text, newspaper

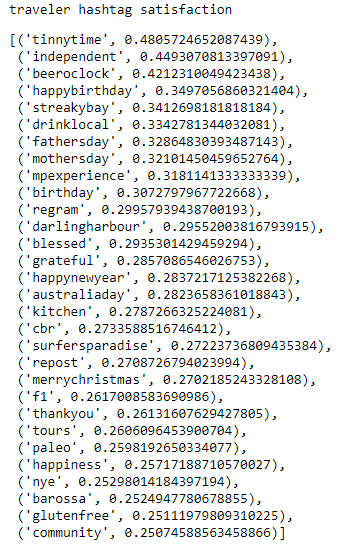
Description automatically generated

Figure 22: traveller VS non-travller satisfaction

By analysing every hashtag in different groups, we can tell that non-travellers have a great passion for sports, music and news, such as “cwc15” (Cricket World Cup 15), “tbt” (The basketball tournament), “nowplaying” (music platform) and QandA (Q&A news).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Top hashtag satisfaction of non-travelers | | | | | |
| Sport | ausvind | cwc15 | bbl04 | ausopen | endomondo | iamcarlton |
| Music | nowplaying | navyblues | riseandshine | australianmusic | Vote5sos | fans1d |
| News | QandA | auspol | victraffic |  |  |  |

In contrast, travelers prefer to drink beer and celebrate festivals, such as “beeroclock”, “happybirthday”, “fathersday” and “merrychristmas”.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Top hashtag satisfaction of travelers | | | | |
| Festival | happybirthday | fathersday | mothersday | merrychristmas | australiaday |
| Drink | beeroclock | drinklocal |  |  |  |
| Tour | mpexperience | streakbay | darlingbarhour | sufersparadise |  |

**7.3.3.1 Discussion**

Travelers are generally happier and productive than non-travelers. The major difference between them is their living habits. Non-travelers tend to watch sports and news at home; whereas travelers pay more attention to celebrating their commemoration days. In fact, celebration may be the major factor that increases travelers’ satisfaction. Our results indicate that people who like to share geolocations are more likely to celebrate on festivals. It is possible to help twitter identify target customers when there is a coming festival. For example, Twitter can recommend carnations to users who have a high frequency of sharing geolocation on Mother’s Day.

**Conclusion**

The backend module is designed with scalability and extensibility by applying automated deployment and traditional producer-consumer pattern. We can easily scale up the system by deploying more consumers (harvesting workers) as we acquire more developer tokens and increasing the volume of the task queue at the mean time. CouchDB is natively scalable since user can add nodes into the database cluster at any time to rebalance the data storage in order to face the challenge of future data deluge. Sharding and replica technologies are the icing on the cake which significantly boost the MapReduce progress by parallelized computation.

The frontend module is based on Now-Ui-Kit, a powerful and beatufitul Bootstrap 4 UI kit created by Tim, it provides various different components that we can use directly in this project. In the sentiment analysis, we found that the satisfactory degree is slightly positively related to the annual mean income and negatively related to the teen ratio in each region. In order to improve the satisfactory degree, the local government should boost regional economics and provide teenagers with more development opportunities. Also, we found that the satisfaction of travelers are generally higher than non-travelers. One possible reason is that travelers pay more attention to celebrating outside the hourse, while non-travelers spend most time on watching sports on TV and listening to Music at home. Twitter can recommend goods to target users (travelers) when some festival is coming. Additionally, we discovered just in Melbourne alone, teens are not the major user among the all, as they do not tweet as often as the teen ratio increases, one possible reason is due to the culture difference between Melbourne and other cities, we believe this result will be more reasonable when we expand our analysis to the whole Australia.

**Work Sharing**

1. Xinglin Qiang

1. Backend Architecture Design
2. Data Harvesting
3. CouchDB Setup and Views
4. Backend Implementation (Flask)
5. Ansible Automated Deployment

2. Shaohua Liu

1. Ansible Automated Deployment
2. Documentation -- Swagger
3. System Containerized
4. Data Harvesting

3. Wentao Hao

1. Frontend - Framework setup
2. Data Analysis - Teen ratio and tweeting frequency

4. Kaixin Chen

* 1. Frontend – Map functionality
  2. Data Analysis – Hashtag Analysis

5. Yichao Xu

1. Frontend - Statistics charts, time search and popup cards.
2. Data - AURIN data download and process.
3. Data Analysis - Sentiment analysis
4. Video making.

**Reference**

[1] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[2] Allen, S. (2019). Social Media’s Growing Impact on Our Lives. American Psychological Association. Retrieved from:<https://www.apa.org/members/content/social-media-research>

[3] “Cluster set up with API fails with unknown\_error #2797” github.com/apache/couchdb/issues. N.p., 21April, <https://github.com/apache/couchdb/issues/2797>