COMP90024 Australia Social Media Analytics Report

Hongxiang Yang: 674136
hongxiangy@student.unimelb.edu.au

Jiaming Wu: 815465
jiamingw@student.unimelb.edu.au

Qingyang Li: 899636
qingyangl4@student.unimelb.edu.au

Xueyao Chen: 851312
xueyaoc@student.unimelb.edu.au

May 2018

1 System Introduction

This project is aimed at building a cloud-base system that can conduct analysis on social media data and help users discover interesting stories of life in Australia. The system will start by harvesting large amount of data from Twitter, and put them into a easy to manage database. Then it conducts analysis on those data, such as semantic analysis and topic identifying, and compare its findings with related data from Australian Urban Research Infrastructure Network (AURIN) [1] and Australian Bureau of Statistics (ABS) [2] , trying to mine embedded patterns and interesting scenarios. Finally, analysis results will be demonstrated on a website with suitable visualization method, which allows visitors to navigate between different topics and see analysis results in an easy-to-understand way. It is designed that the system is an ongoing project. It will keep harvesting new tweets, and periodically repeat the analysis using the updated data pool, and refresh the new result to the website. The whole system is based on the NeCTAR Research Cloud, exploiting several high-performance virtual machines for better reliability and availability.

2 Twitter Analysis Scenarios

Analysis data used in this report are from two sources. Part of them are harvested directly from Twitter API, however, due to the API rate limit and project time limit, the collected data is not enough to support a convincing analysis. Thus an existing data pack is used as a compensation [1]. All data in the database are filtered that only tweets with coordinates information are included.

2.1 Sentiment Analysis

We implement sentiment analysis on more than 4,000,00 tweets collected from two cities: Melbourne and Sydney. All of the analytic scenarios related to sentiment analysis are in suburbs of Melbourne and Sydney. We use the percentage of positive tweets in one suburb to represent

[1] https://data.aurin.org.au/

[2] http://worldpopulationreview.com/countries/australia-population/cities/

the happiness index of that region. For comparison, we use data from AURIN and ABS. The followings are our analytic scenarios.

- Scenario 1: Happiness Index vs. liquor license number
  Venues need liquor license to sell alcohol for consumption on the premises. If a suburb has more venues that can sell alcohol, it will be easier for residents of that suburb to get access to alcohol. We assume that the convenience brought by easy access to alcohol will make people happier. Figure 1 shows the comparison of result between the happiness index and liquor license number in suburbs of the Greater Melbourne Region. As can be seen from Figure 1, people are happier in the eastern part of the Greater Melbourne Region, and the venues in the eastern part of the Greater Melbourne Region have more liquor licenses. The comparison data are collected from AURIN.

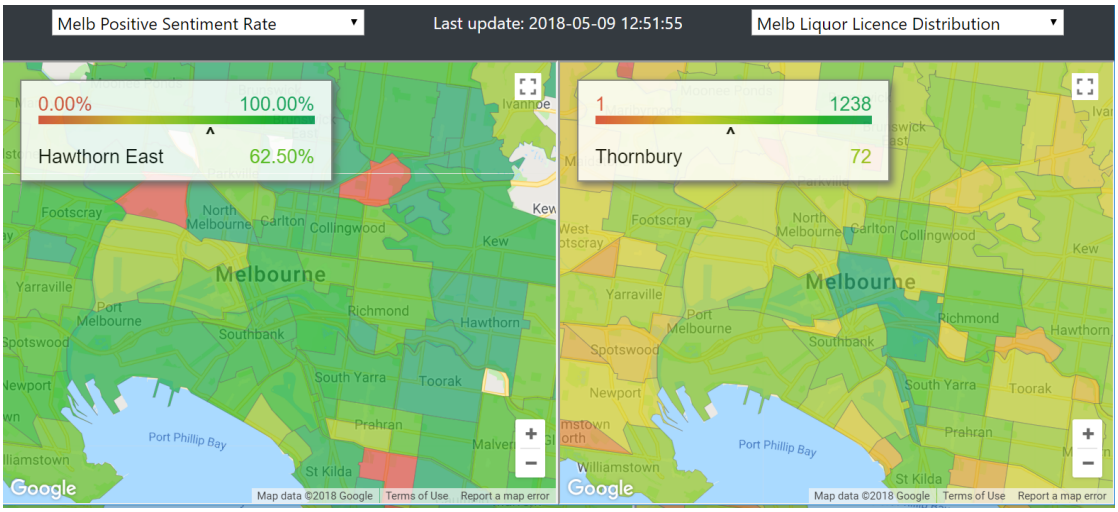


Figure 1: Happiness Index vs. liquor license number

- Scenario 2: Happiness Index vs. sport and recreation facility number
  People live in suburbs with more sport and recreation facilities could have more opportunities to participate in sports. And it is known that sports do good to people's health and can make people feel better. So we suppose that the suburbs with more sport and recreation facilities will have residents with higher happiness index. Figure 2 shows the comparison of result between the happiness index and the number of sport and recreation facilities in suburbs of the Greater Melbourne Region. As can be seen from Figure 2, people in the southeastern part of Melbourne are happier and the suburbs in the eastern part of Melbourne have more sport and recreation facilities. We can conclude that there are some relationship between the happiness level and the number of sport and recreation facilities in the Great Melbourne Region. The comparison data are collected from AURIN.

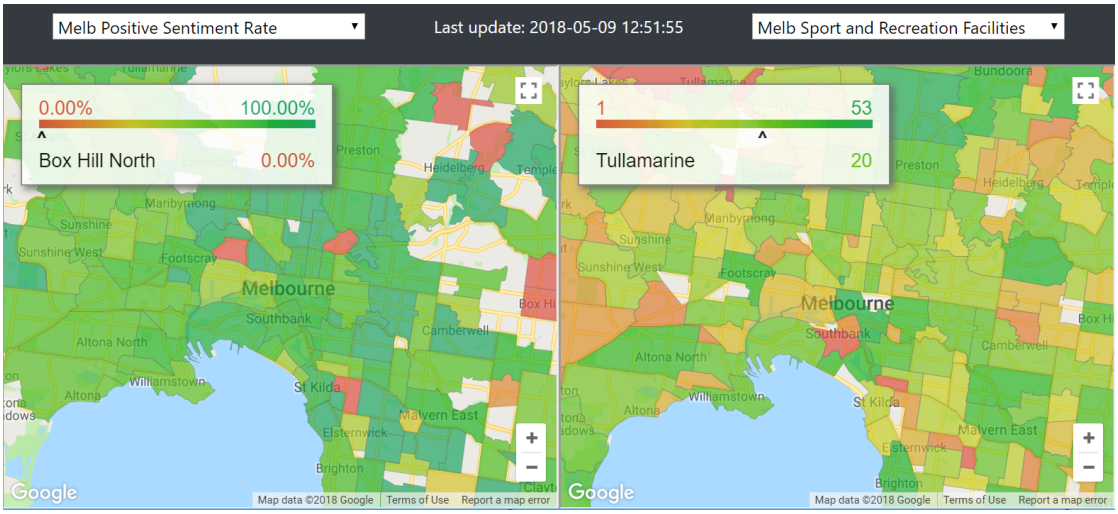


Figure 2: Happiness Index vs. sport and recreation facility number

- Scenario 3: Happiness Index vs. personal income(weekly)
  We assume that suburbs with higher average personal income will have happier residents. We compare the happiness index with average personal income(weekly) in suburbs of the Greater Melbourne Region and the Greater Sydney Region. Figure 3 below only shows the comparison of result between the happiness index and the average personal income in suburbs of the Greater Sydney Region. As can be seen from Figure 3, the coastal area of the Greater Sydney Region has residents with higher happiness index. However, the average personal income in suburbs distributed evenly in the Greater Sydney Region. Conclusion can be drawn that there is little relevance between happiness and income among the suburbs of the the Greater Sydney Region. The comparison data are collected from ABS.

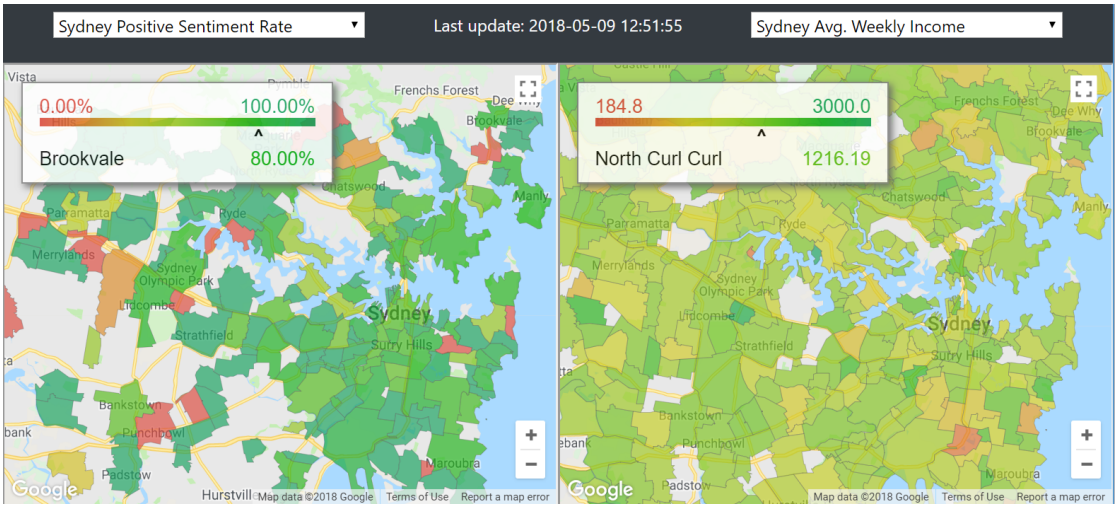


Figure 3: Happiness Index vs. average personal income(weekly)

- Scenario 4: Happiness Index vs. voluntary work participation rate
  It's a common sense that helping others will make people happy. So we assume that suburbs with higher voluntary participation rate will have residents with higher happiness index. Figure 4 shows the comparison of result between the happiness index and the voluntary work participation rate in suburbs of the Greater Sydney Region. As can be seen from Figure 4, the coastal area of the Greater Sydney Region has residents with higher happiness index. However, the voluntary work participation rate in suburbs distributed evenly in the Greater Sydney Region. Conclusion can be drawn that there is little relevance between happiness and the voluntary participation rate among the suburbs of the the Greater Sydney Region. The comparison data are collected from ABS.

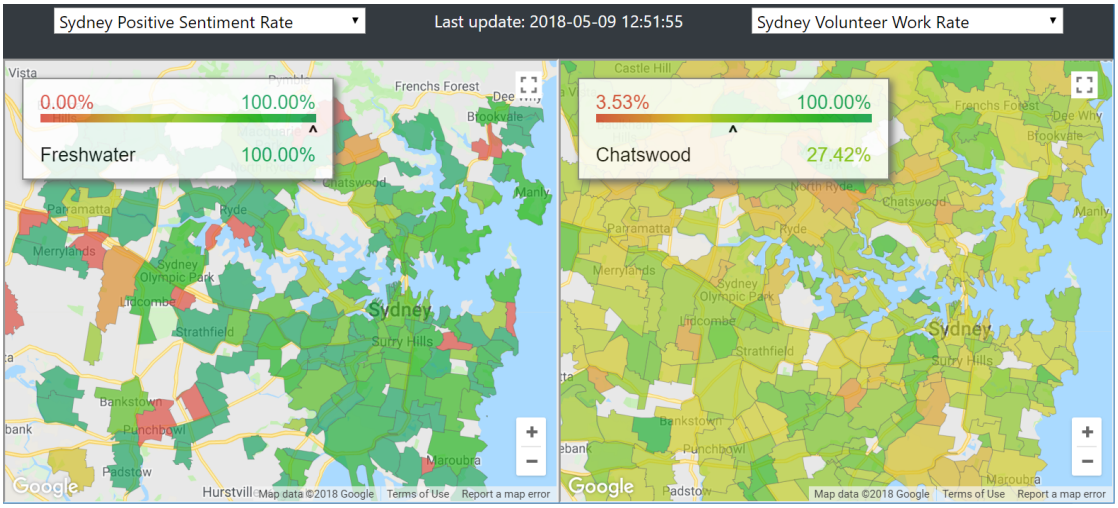


Figure 4: Happiness Index vs. voluntary work participation rate

- Scenario 5: Happiness Index vs. crime number
  Crime such as assaults and robberies have bad influences on people's daily life. We assume that suburbs with more crime cases will have residents with lower happiness index. Figure 5 shows the comparison of result between the happiness index and crime cases number in center suburbs of the Greater Sydney Region. We can see from Figure 5 that suburbs with more crime cases prone to have lower happiness index. For example, suburb Sydney city has the most crime cases among all the suburbs investigated, which is one of the "most unhappy suburbs". So we can draw conclusion that suburbs with more crime prone to have less happy residents. The comparison data are collected from AURIN.

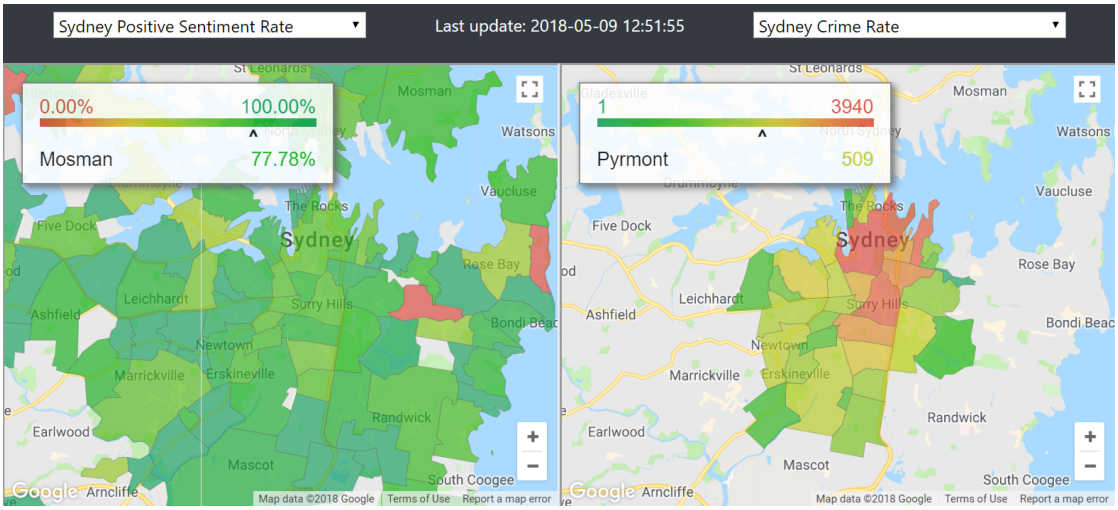


Figure 5: Happiness Index vs. crime number

2.2 Language Usage

We also analyse the non-English language use condition between real world and Twitter. We compare the top 10 Non-English languages spoken at home and the top 10 Non-English languages used on Twitter in 5 cities. Figure 6 only show our analysis on the data of the city Sydney. As we can see from Figure 6, there are many overlaps between the language types spoken at home and used on Twitter. Chinese, Arabic, Taglog rank within top 10 in both language use conditions. However, it seems that Chinese people are more willing to speak their own language at home instead of using it on Twitter. And people from Southeastern Asia use their own languages more frequently than speak them at home. The possible reason for this interesting observation is that Chinese people have their own social network apps such as Wechat and Weibo, so if a Chinese person want to communicate with others or share his/her life on social network via Chinese, Twitter won't be his/her first choice. Another interesting observation is that Japanese is not among the top 10 Non-English languages spoken at home, which indicates that the population who speak Japanese as native language is not that large in Sydney. However, Japanese is a frequently used language on Twitter. One possible reason for this observation is that Japanese is a popular language among the residents in Sydney, many people choose to learn Japanese as their second language. The other possible reason for this observation is that many young people in Sydney love Japanese culture and usually share videos containing Japanese on Twitter. The comparison data are collected from AURIN.

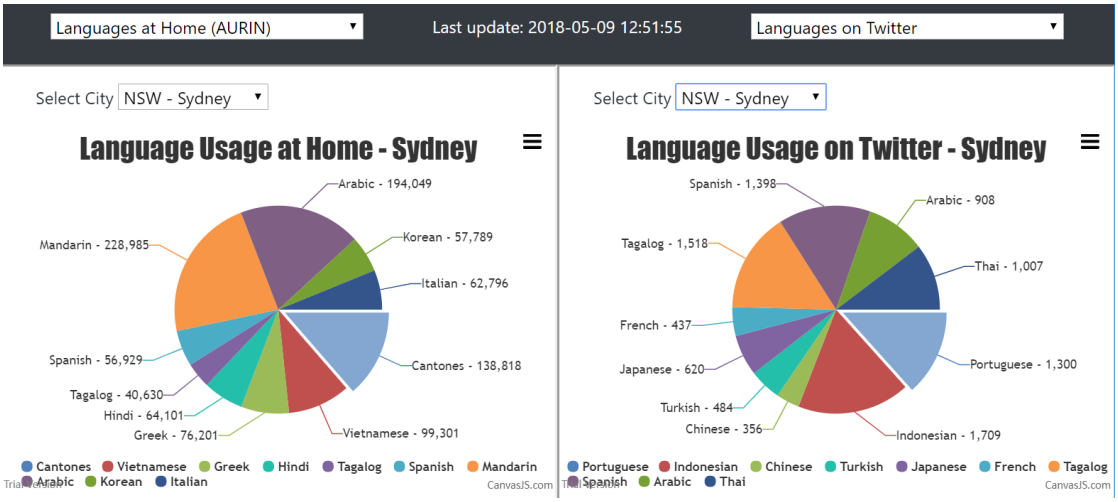


Figure 6: Language Usage

2.3 Topic Modeling

In this part, we implemented a python application to identify and visualize 10 topics presented in the tweets we've obtained[2]. There are many approaches are available in the field: Term Frequency and Inverse Document Frequency (TF-IDF), NonNegative Matrix Factorization techniques, Latent Dirichlet Allocation(LDA), etc. LDA is the most popular technique and we implemented a naive version in the following steps:

1. Read tweets from a predefined view from couchDB
2. Text cleaning and preprocessing
   - Extract words, hashtags, urls, @-mentions
   - Normalization and lemmatization
   - Prepare Document-Term Matrix
3. Build the LDA Model (gensim, Log 1)
4. Visualize the result (pyLDAvis, Figure 7) and generate a html file

```
<Database 'tweets'>
number of tweets: 487165
(0, '0.014*"budget" + 0.014*"great" + 0.013*"thank" + 0.012*"always" + ..."')
(1, '0.011*"look" + 0.009*"#auspol" + 0.008*"would" + 0.006*"enjoy" + ..."')
(2, '0.013*"#sydney" + 0.005*"weekend" + 0.004*"favourite" + ..."')
(3, '0.018*"melbourne" + 0.009*"australia" + 0.006*"might" + ..."')
(4, '0.016*"night" + 0.009*"better" + 0.008*"amaze" + ..."')
(5, '0.017*"#budget2018" + 0.015*"happy" + 0.008*"agree" + ..."')
(6, '0.017*"#hiring" + 0.016*"thanks" + 0.013*"#careerarc" + ..."')
(7, '0.015*"people" + 0.011*"morning" + 0.010*"please" + ..."')
(8, '0.031*"today" + 0.017*"sydney" + 0.009*"humidity" + ..."')
(9, '0.012*"start" + 0.009*"friend" + 0.008*"close" + 0.007*"beautiful" + ..."')
```

Code Block 1: Topic Modeling Log

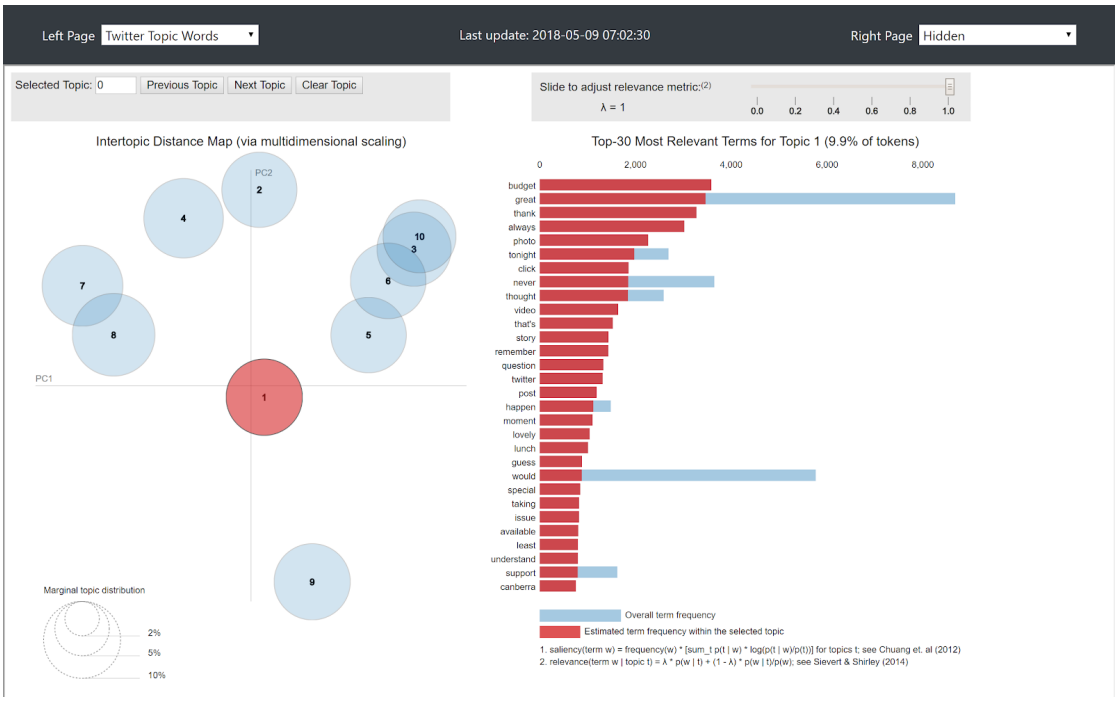


Figure 7: Topic Visualization

2.3.1 Side note: Boosting Numpy Performance

Noticing that running the above application is very time consuming, i.e., its takes approximate 3 hours to extract topics from 480,000 tweets. In gensim, the most extensive computation is done inside the low-level routines of linear algebra, namely, inside NumPy package, which is independent of any gensim code. Installing a fast BLAS (Basic Linear Algebra) library for NumPy can improve performance up to 15 times [3]. In our application, we applied the OpenBLAS library. A benchmarking program is used to test its performance on a single machine (Log 2).

```
Dotted two 4096x4096 matrices in 4.80 s.
Dotted two vectors of length 524288 in 0.57 ms.
SVD of a 2048x1024 matrix in 2.89 s.
Cholesky decomposition of a 2048x2048 matrix in 0.52 s.
Eigendecomposition of a 2048x2048 matrix in 42.25 s.
```

Code Block 2: OpenBLAS benchmark

2.4 Popularity Analysis

The popularity analysis is to find out where are those popular tweets sent in Australia. Basic idea is that the higher influence the tweet, the darker color it will show. In a national view of the map, the color can represent the general difference of Twitter influence between areas. As can be seen from the heatmap, popular tweets are mostly located at big cities, ranking from highest are Sydney, Melbourne, Brisbane, Gold Coast, Perth, Adelaide. This result is obviously related with area population. But one thing can be noticed is that, although the population of Sydney and Melbourne is quite close [3], the tweet influence of Sydney is apparently higher than Melbourne. The limitation of this analysis is that, it only shows tweets that post with coordinates, which are only small parts of the total tweets, and thus is not representative of the overall tweet popularity among Australia.

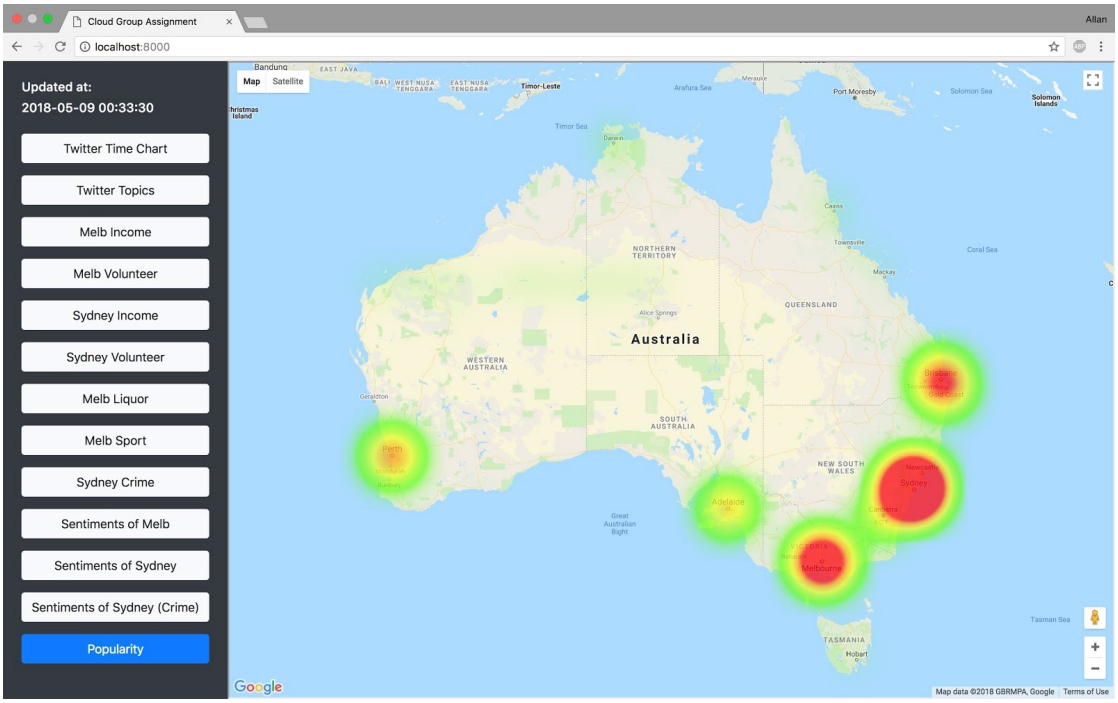


Figure 8: Popularity Heat Map

[3] http://worldpopulationreview.com/countries/australia-population/cities/

```
ec2_conn.run_instances('ami-00003837',
                       max_count=number_of_instances,
                       key_name='id_khris',
                       instance_type='m1.medium',
                       security_groups=['ssh'],
                       placement='melbourne-qh2')
```

Code Block 3: Launch Instance

```
with open('/etc/ansible/hosts', 'a') as f1, \
        open('ips', 'w') as f2, \
        open('/etc/ansible/group_vars/{}'.format(group_name), 'w') as f3:
    f1.write("[{}]\n".format(group_name))
    f3.write('---\nansible_ssh_user: ubuntu')
        for i in range(len(instances)):
        f1.write("host{} ansible_ssh_host={}\n".format(
            i, instances[i].private_ip_address))
        f2.write("{}\t host{}\n".format(instances[i].private_ip_address, i))
```

Code Block 4: IP handling

3 User Guide

3.1 Virtual Machine Setup

In this part, boto (2.48.0p), ansible (2.5.0), docker (18.03.0-ce) were used to setup and automate the environment of our application.

Boto allows us to automate the initialization of the cloud resources being allocated to the team. Code Block 3 shows the basic configuration of each virtual machine (VM)/ instance. Notably, the security group needs to be predefined through the NeCTAR dashboard.

When all the instances are ready to run, we kept a copy of the instances ips on the control machine. In the meantime, these ips were populated to the respective ansible configuration file which requires the sudo privilege to make the modifications (Code Block 4).

The boto script takes the number of instances and group name as arguments from the command line. In this work, we used the following command to run the script:

sudo python3 launch_instance.py 4 servers

As a result, we have 4 running instances on NeCTAR, and the ansible configuration on the control machine is set, which allows us to communicate with the four instances through the ansible. We can test the communication through:

ansible -m ping servers

Then, we apply the ansible *blockinfile* module to insert the text content in ips file on the control machine into *remote:/ect/hosts*. By doing so, the four instances can now identify and communicate with each other.

```
{"all_nodes":["couchdb@115.146.84.171","couchdb@115.146.85.150",
"couchdb@115.146.86.214","couchdb@115.146.86.30"],
"cluster_nodes":["couchdb@115.146.84.171","couchdb@115.146.85.150",
"couchdb@115.146.86.214","couchdb@115.146.86.30"]}
# all_nodes are all the nodes thats this node knows about
# cluster_nodes are the nodes that are connected to this node.
```

Code Block 5: Check CouchDB Membership

Following, we use another boto script to allocate 50GB additional volume to each instance to store the tweets we harvested. Besides, an extra 10GB volume is allocated to the main server to setup the Network File System(nfs) among all the instances. Notably, by storing all of our data on theses additional volume introduces extra reliability of our system. i.e., If any of the server goes down, we can recover our data from these volumes by simply reattach them.

Finally, we run an ansible playbook to mount the volume to appointed path (*/mnt/extra*) on each instance and install the nfs (/share) server/client respectively.

3.2 Docker & CouchDB Setup

To setup CouchDB on the VMs, we use ansible to run the following scripts in /share/dbSetup:

- installDocker.sh
- dbSetup2.sh
- masterNode.sh

Where the first two scripts run on each vm, which installs docker (18.03.0-ce) and runs a docker instance with couchdb 2.1.1 image, the last script runs on a single vm and setup the connection between each couchdb. To verify the connection is all set,we can run the following command on a vm:

curl -XGET http://$user:$pass@$IP:5984_membership

Which should output something like:

3.3 System Guidance for End User

The web system (**http://115.146.86.30:8898/**) is designed for data visualization and provides a two-screen display model for convenient comparison.The screen is consists of four parts (Figure 9).

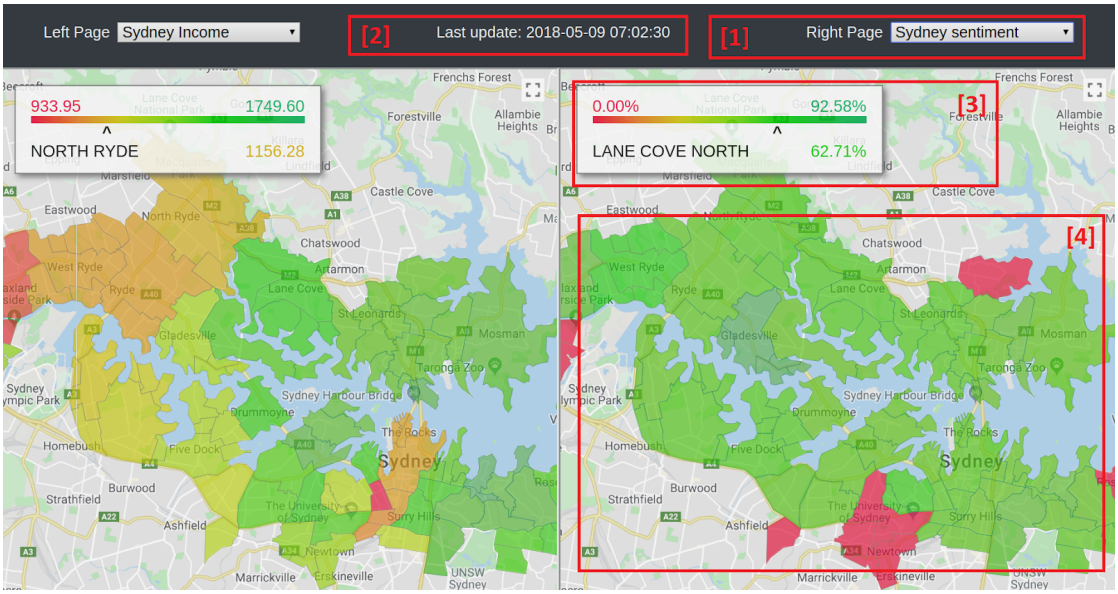


Figure 9: Frontend Application

1. Dropdown List, which is used for choosing the data to display. Firgure 10 shows the detailed choices:
   - Hidden: hide the right pane, this use used to display on result in full screen
   - ABS Data: demographic data from Australian Bureau of Statistics [4]
   - Aurin Data: demographic data collected from Australian Urban Research Infrastructure Network [5]
   - Twitter Data: sentiment analysis result base on tweets content
   - Non-English Languages: Pie chart that shows the top 10 non-english tweet languages of each major city
   - Other Views
     - Twitter Time Chart: a curve line chart showing the post time of tweets collected for analysis
     - Twitter Topic Words: current topic group on Twitter
     - Harvester Status: the operation log of the Twitter harvester, including operation time of the harvester and how many data collected so far
   - Special-Hidden: hide the right pane, this use used to display on result in full screen, as shown below:
2. Update time: this information indicates the last update time of data used in this page
3. Hover window: showing the details data of the area where the cursor points
4. Heatmap: demonstration of analysis results

[4] https://datapacks.censusdata.abs.gov.au/datapacks/
[5] https://data.aurin.org.au/

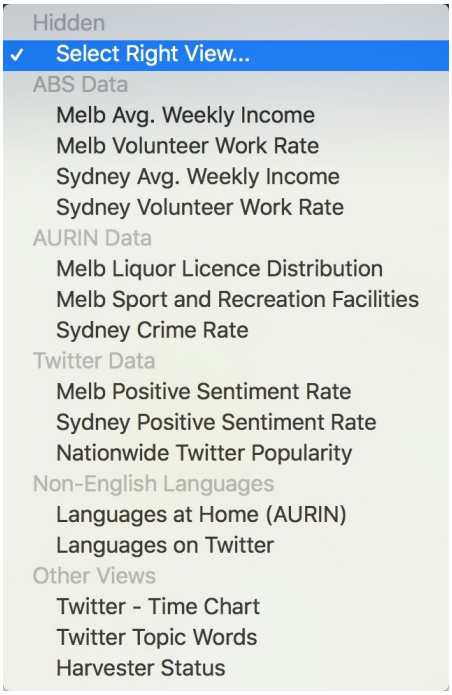


Figure 10: Dropdown List

4 System design & architecture

4.1 System Architecture Intro

Our system implements 4 virtual machines, namely, host0, host1, host2, host3. Below illustrates the functionality of each vm (Figure 11):

- Host0:
  - Login node
  - Network File System (NFS) Server
  - Develop & test environment for our application
- Host1:
  - Application node (only accessible to system administrator)
  - Runs tweeter harvester and analysis program from NFS
- Host2:
  - Application node (only accessible to system administrator)
  - Runs visualization program from NFS
- Host3:
  - Application node (only accessible to system administrator)

– Runs tweeter harvester program from NFS

– Web Server, enables HTTP connection

Finally, a control machine is responsible for managing the above nodes through ansible and scripts. To run applications, the control machine sends out the instruction and runs the respective script on the appointed machine.

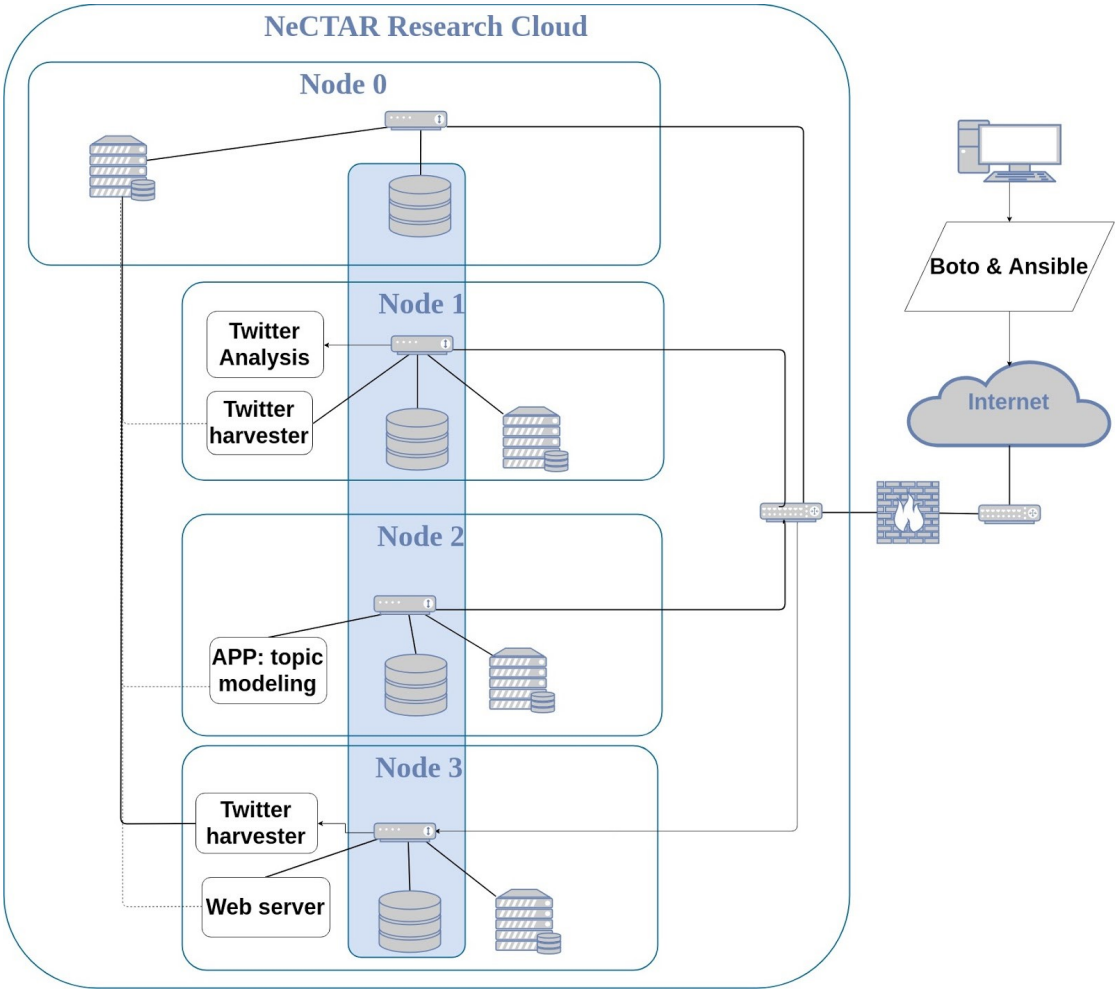


Figure 11: System Architecture

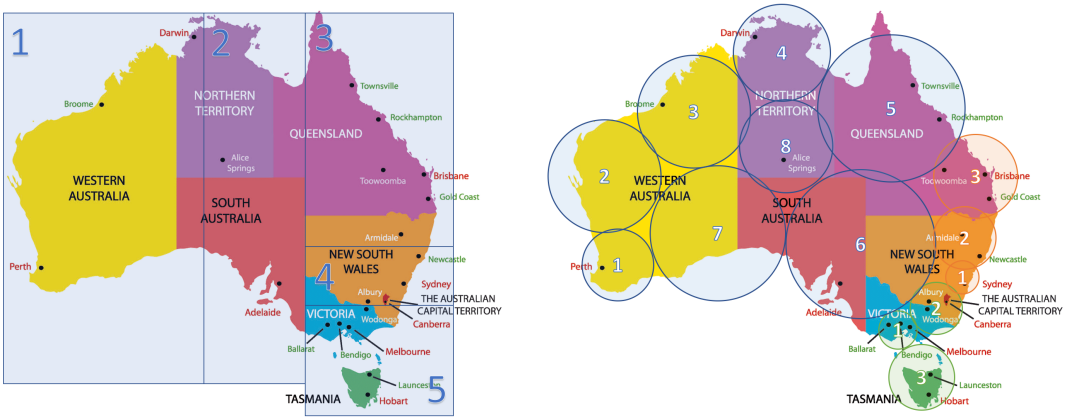
4.2 Harvester

4.2.1 Twitter APIs

Twitter provides several APIs that allow us to retrieve tweets, aka statues. Since we want to harvest tweets filtered by their location, *Real-time Stream* and *Search* are two ideal ways to do that. The real-time stream API will push fresh tweets through a long connection, while the search API returns tweets within 7 days (for free users) filtered by given query.

However, either API has a rate limit. To harvest more tweets, we applied many API keys and run harvesters with them simultaneously. In this project, we got 14 API keys in total. Also, we divided the whole Australia into disjoint parts to reduce replicated tweets.

The real-time stream API supports filter by latitude and longitude values. The number of tweets harvested by streaming is not very big, so we only used 5 keys for this API which is quite enough. In comparison, search API requires a centre point and a radius. We used 14 circles with API keys to cover the continent (roughly).



(a) Stream (b) Search

Figure 12: Geographic areas for each API keys

4.2.2 Harvesting

Our harvester is written in Python. The API keys and its associated geographic area are stored in JSON files which will be loaded dynamically when the harvester started. Thus, we can easily run multiple harvesters in the cluster by assigning them different keys files. Each harvester manages a bunch of background worker threads to send requests.

CouchDB-Python is a great package to invoke CouchDB with python and it brings us a lot convenience. The harvester has a cache (a python queue) to store incoming tweets temporarily and upload upload a group of packaged tweets in a single request to improve performance.

Removing replicated tweets is the most important feature of our tweet harvesting system. The harvester achieved this by assigning the "*_id*" field of each uploaded documents with the tweet id. By doing this, requests which uploads replicated documents (with existed "_id") will fail and got documents filtered naturally.

4.3 Data storage

4.3.1 Why CouchDB

In recent years, major social media platforms like Facebook and Twitter started to adopt nosql database techniques to get rid of intuitive issues from relational database they initially deployed, which indicates that nosql database can provide better performance on retrieval and

analysis on social media data. In this system, since we need to process json file from Twitter, a nosql database called CouchDB is deployed in order to get a better performance. There are several benefits of using CouchDB here, first, CouDB is good at processing json file. Second, it is a schema-less database with flexible document-oriented structure. In this system, we can just throw the harvested data in it without worrying about data structure. Third, it provides HTTP-based Restful API that allow access to database easy to understand and deploy. Lastly, there is embedded mapreduce function that is very helpful in the following analysis.

4.3.2 Installation Choice

There are three ways of installing CouchDB. The binary package installation cannot success since the package provided from Apache website is incomplete. The source code installation is complex as it needs to install lots of dependencies and only get an ancient version of CouchDB (1.6). Therefore, we choose the docker method, which provides quick and convenient installation of the newest version of CouchDB (2.1). A more important reason of using docker is that the database is in an isolated container, easy to start and stop, and also convenient to modify and update.

4.3.3 Database Cluster

In our system, a 4-node CouchDB cluster is build to increase the fault-tolerant ability. The cluster specification is set to "replication=3, sharding=8", which means that the database will be shard into 8 parts, and will have three copy. Thus, there will be total 24 shards, and distributed to 4 node evenly, each node will store 6 shards. This setting result in each node holding 6/8 or 75% of the total data, which means the system can withstand up to three node failure and still have most of data available. When a cluster member recovered from a crash, CouchDB will automatically rebalance the data among the cluster. Despite of high availability, the no-central feature of CouchDB cluster allows harvester to upload data from multiple nodes concurrently, which indeed help balance the load between nodes, and speed up cluster synchronization.

4.3.4 Map-reduce

The view function in CouchDB provide the map-reduce method for query. We use this function to retrieve data from the database for analysis. There are several tasks that is achieved using mapreduce. Firstly, using map function we filter the documents based on content and only retrieve the necessary part, instead of the whole document. One example is the following view that filters out tweets whose coordinates located within Melbourne and Sydney (Code Block 6):

Besides of filtering, the map function allow us do analysis based on document content and return the result with the query. One example is the following view that calculates the popularity of each tweets (Code Block 7):

Combine map and reduce function we can do calculation based on the query result, one example is to calculate how many tweet of each language(Code Block **??**):

However, there are also issues of using map-reduce on CouchDB. Each time the view is queried, an index is built that can speed up future repeat query. Although the index can be incremental based on newly added data, the initial index building takes a lot of time when data is large (2 hours for 6 million documents). And this add difficulties for making new queries latter. Our mediation here is to filter the data before pushing them into the database, so that we can avoid indexing on useless data.

```
function (doc) {
  if (doc.coordinates != null) {
    var coords = doc.coordinates.coordinates;
    var lng = coords[0];
    var lat = coords[1];
    if (144.80 < lng && lng < 145.15 && -37.9350 < lat && lat < -37.672032) {
      emit(doc.coordinates, doc.text);
    } else if (150.9589 < lng && lng < 151.36 &&
    -33.96 < lat && lat < -33.7633) {
      emit(doc.coordinates, doc.text);
    }
  }
}
```

Code Block 6: View_Coordinate

```
function (doc) {
  if ((doc.coordinates != null) && ((doc.favorite_count > 0) ||
  (doc.quote_count > 0) || (doc.retweet_count > 0)||(doc.reply_count > 0)))
  {
    var pop = 0
    if (doc.favorite_count > 0)
      pop = pop + doc.favorite_count
    if (doc.retweet_count > 0)
      pop = pop + 3*doc.retweet_count
    if (doc.quote_count > 0)
      pop = pop + 3*doc.quote_count
    if (doc.reply_count > 0)
      pop = pop + 2*doc.reply_count
    emit(doc.coordinates, pop);
  }
}
```

Code Block 7: View_Popularity

```
"Map":
"function (doc) {
  if (doc.place.full_name.indexOf("Melbourne") != -1) {
  emit(["Melbourne",doc.lang],1);
  }
  else if (doc.place.full_name.indexOf("Brisbane") != -1){
     emit(["Brisbane",doc.lang],1);
  }
}"

"reduce": "_stats"
```

Code Block 8: View_Language

4.4 Data Analysis

4.4.1 Sentiment Analysis

1. Training Dataset:
   We choose supervised learning to do the sentiment analysis of the tweets we collected. The training data comes from the sentiment 140 dataset [4]. It contains 1,600,000 tweets extracted using the twitter api. The tweets have been annotated(0 = negative, 4 = positive). We randomly select 1,000,000 tweets from the sentiment 140 dataset as the training dataset.

2. Pre-processing
   For feature extraction of the tweets, firstly we do some pre-processing steps. Considering the unique characteristics of twitters(be of limited length, be filled with slang and misspellings), pre-processing can decrease the size of the feature set and remove some irrelevant information. The pre-processing steps contain hashtag, handle, URL, and emoticon replacement, punctuation and repeating character removal. Secondly, we use porter stemmer [5] for stemming.

3. Feature
   Finally, we use n-grams and negation handling to get the feature.

   (a) N-grams:
   N-grams can be used to predict the next word given a current context of words. In our project, we use Bigrams to get feature.

   (b) Negation handling:
   Negation handling is also implemented in our sentiment analysis. Our negation handling contains two parts: detection of explicit negation cues and scope of negation. To detect the explicit negation cues, we search all negation cues defined in the tweets. For scope of negation, we define that the words immediately preceding and following the negation cues are the most negative and the words that come farther away from the negation cues lie out of the scope of the negation of the cues. We define left and right negativity of a word as there are chances that meaning of that word is actually the opposite. Left negativity depends on the closest negation cue on the left and right negativity depends on the closest negation cue on the right.

   (c) Classifier:
   We use Naive Bayes classifier as our classifier to give tweets a sentiment tag. Naive Bayes classifier is the simplest classifier. Considering that we need to tag more than 4,000,00 tweets, a fast classifier is necessary.

   (d) Tools and Language:
   Our analysis codes are written in Python, Natural Language Toolkit(NLTK) is used for symbolic and statistical natural language processing.

   (e) Limitation:
   The main limitation of our sentiment analysis is that our training data has only two types: positive and negative, which means that all tweets are classified into only two polarities. Actually, a large part of tweets don't have apparent sentimental polarity and should be regarded as neutral. However, training a classifier with high classification accuracy needs a large training dataset, which can not be manually collected. In the future, hashtag can be used to identify positive, negative and neutral tweets. In this way, a large dataset with 3 types of polarities can be collected and used as training data to improve the performance of sentiment analysis.

4.4.2 Popularity

The popularity analysis is to find out where are those popular tweets sent in Australia. To give a general view of influence of a tweet [6], we give each tweet a influence grade based on a weighted scoring system:

$$Totalgrade = Favoritecount * 1 + Retweetcount * 3 + Quotecount * 3 + Replycount * 2$$

4.5 Web Application

The web application is a website that displays the analysis result with human-friendly visualization. There are two main parts of the web app: web pages and data preprocessing program.

4.5.1 Front-end

The front-end part of the web app is consisted of two parts: web pages and json data files. Nearly all features are developed by javascript, web pages use javascript and ajax to get data from json files and display them with some simple process. Actually the website is totally served statically without normal “backend programs". The preprocessing program will take charge of providing required data.

The index page has two iframe elements to load and display content pages. The user can choose display one or two content pages for comparing data clearly. With this iframe architecture, arranging content pages are unbelievably easy. As for each content page, they will load required json data file and display them with Google Maps or charts.

4.5.2 Data Preprocessing

The output data of the analysis program is not ready for displaying, there is still much work to do before we can visualize them. Intuitively, this heavy work should not running on the client browsers. So, we need to do some prepossess on the server to make the “raw" analysis results suitable for the front-end. For Google Maps pages, the data are filtered and packaged into geojson files which can be loaded directly into maps. For other pages, they are formatted into json objects or arrays, depend on the content.

One challenge of data preprocessing is convert postcodes or coordinates (longitudes and latitudes) into cooresponding suburbs. Google provides an API which accepts coordinates and return postcodes and suburbs, however, it is extremely slow and have a low rate limit. In the begining, we tried to handle data classified postcodes and found it really difficult. Later, we adjusted the programs and used “state suburb code, SSC" to replace postcodes and solved some problems.

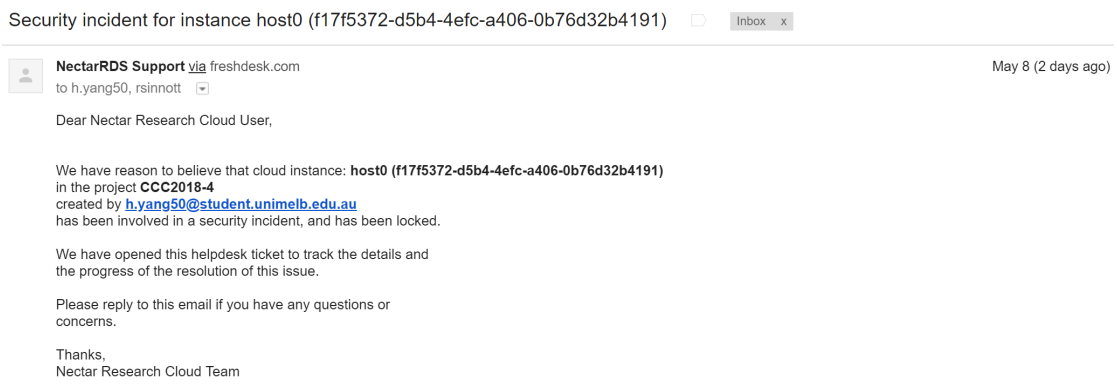
It will take a long time for each execution of the analysis program, especially when the database become larger and larger. We decided to run the analysis program followed by the preprocessing program regularly, for example, twice a day, and cache the result into the web app. Every time we access to the website, we will load the cached semi-fresh data directly.

5 Discussion of NeCTAR Research Cloud

In this part, we conclude some benefits and challenges of the NeCTAR Research Cloud according to our deployment and application of the system.

Benefits: NeCTAR cloud provides a reliable and secure computing environment for the project system to deploy and operation continuously. By running the system in cloud, we avoid the worry about private machine failure, which is much more possible than a cloud node failure, since the cloud is well-maintained. The cloud also provides additional computing capacity that can be used for data analysis. Its stable configuration can avoids uncertainties of analysis and increase the repeatability. Besides, for group project, NeCTAR cloud is convenient for team members to access the project at anytime and collaborate with each other form their own laptop.

Challenges & Error Handling: The biggest challenges we've encountered is security and user permission management. For the security aspect, our login node that enables the password authentication was reported involving in a security incident (Figure 13) and hence locked by the system administrator two days before the assignment due date. Before that, all of our four instances have been running for 18 days (Figure 14). In our case, since it was not possible to resume the instance due to system wide security concerns, we had to rebuid our login node (host0). Fortunately, all of our important data are stored in the extra volume, which won't be affected even if the instance was deleted. Consequently, we can "restore" the login node through our automation process, recover the data from extra volumes and resume CouchDB by joining the newly created node into the existing cluster.


Security incident for instance host0 (f17f5372-d5b4-4efc-a406-0b76d32b4191) Inbox x

NectarRDS Support via freshdesk.com May 8 (2 days ago)
to h.yang50, rsinnott

Dear Nectar Research Cloud User,

We have reason to believe that cloud instance: **host0 (f17f5372-d5b4-4efc-a406-0b76d32b4191)**
in the project **CCC2018-4**
created by h.yang50@student.unimelb.edu.au
has been involved in a security incident, and has been locked.

We have opened this helpdesk ticket to track the details and
the progress of the resolution of this issue.

Please reply to this email if you have any questions or
concerns.

Thanks,
Nectar Research Cloud Team

Figure 13: Reported Security Indicent

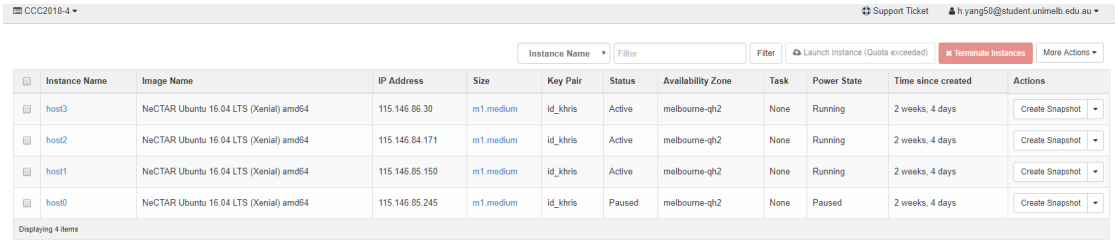


Figure 14: Runing Instances

6 Appendix

GitHub: `https://github.com/senior88oqz/AusSocialMediaAnalytics.git`

YouTube Demo(Part 1): `https://youtu.be/3HvMOxogNCQ`

YouTube Demo(Part 2): `https://youtu.be/im-OT926a_g`

References

[1] BigTwitter, UniMelb, May 2018

[2] https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/

[3] https://radimrehurek.com/gensim/distributed.html

[4] Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(12).

[5] Porter, M. E. (1998). Clusters and the new economics of competition (Vol. 76, No. 6, pp. 77-90). Boston: Harvard Business Review.

[6] Cha, M., Haddadi, H., Benevenuto, F., & Gummadi, P. K. (2010). Measuring user influence in twitter: The million follower fallacy. Icwsm, 10(10-17), 30.