Australian City Analytics

--- Scenario analysis based on cloud computing

Student ID: 765821 **Student Name: Can Cui**

Student ID: 653953 **Student Name: Hao Han**

Student ID: 718373 **Student Name: Yige Sun**

Student ID: 682252 **Student Name: Weixiao Liang**

Student ID: 775073 **Student Name: Zhenyu Mu**

Source code link:

https://github.com/itachi1232gg/Assignment2

Video link:

https://youtu.be/h0Sy_dJWFyg

Table of Content

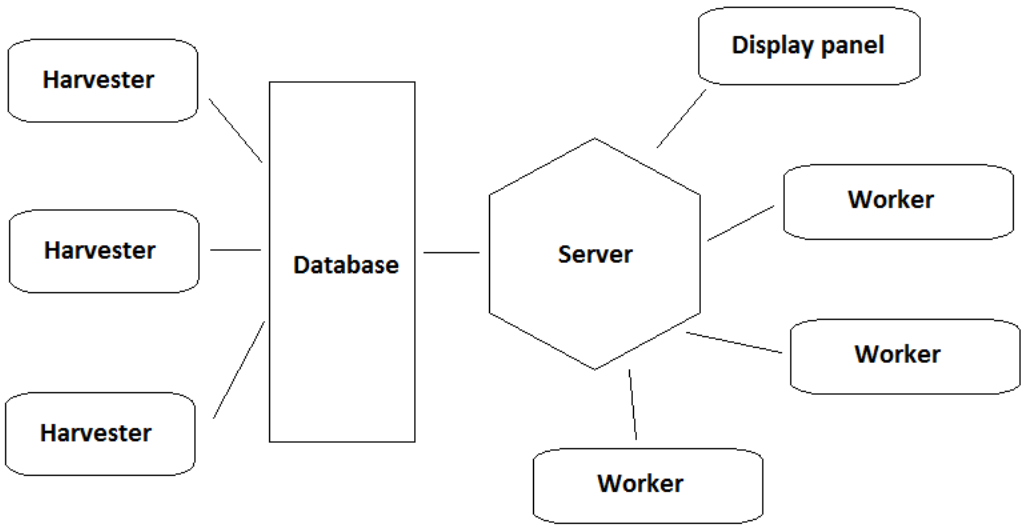
1. Introduction

Nowadays, people are living in a world filled with digital information. When people are receiving data from their mobile devices, they also make contribution to the digital world by sending messages, twittering or liking others on Facebook. All these behaviors can be tracked and stored as data, which makes it possible for others to analyze. Although this results in some privacy issues, its benefits are obvious. People can not only generate a better understanding of traditional knowledge but also make better decision or predict events. This system aims to harvest and analyze the twitter data in Australia and gives suggestion of which part of Australia is suitable for immigration. This object can be divided into three different issues: the happiness index of people in five main cities of Australia, people's attitudes towards immigration and health in Australia, main topics of different cities in Australia. To finish these tasks mentioned above, a cloud-based solution is provided, which is mainly running on NeCTAR Research Cloud with dynamic data harvesting, analyzing and displaying. This report will firstly illustrates the structure and functionality of the system. Secondly it will explain different components and their implementation, which are twitter harvester, sentiment analysis, hot topic analysis, database and back-end server design and front-end web design. Thirdly, the analysis of the result will be given, which is followed by user guide and Nectar deployment. In the end, it will conclude by evaluating this system and the results.

2. System Design

2.1 Architecture

The whole system is consist of 5 parts: harvester, database, server, worker and display panel. The architecture is shown as below:



2.2 Harvesting Tweets

This assignment is basically a data mining and analysis task, therefore collecting twitter data is an essential part of our work. At the beginning, we proposed serval sentiment analysis topics, such as comparing Top 10 topics between tweets collecting from high income suburbs and low income suburbs, analysing university students' moods changing with time (e.g. students may be happier at the beginning of a semester, but feel stressful at the end of the semester). However, after I tried different Twitter API methods, I found some of topics are not feasible, this is because: the number of tweets sent from Australia is limited. By using stream api, I can only collect 20,000~30.000 real-time tweets from Australia each day. Most of tweets do not contain a precise location information, thus we cannot get enough valid data to finish our analysis task.

Stream api can only collect real-time tweets, while rest api search can get the last 7 days tweets. None of these methods can satisfy our requirement for far earlier data. We cannot find relevant data on AURIN to support our research.

So, we discussed together again, and changed some infeasible topics. Finally, we decided to do some sentiment and topic analysis on health, immigration and racism across the top 5 largest cities (Sydney, Melbourne, Perth, Brisbane and Adelaide) in Australia. My aim here is to harvest as many data as possible and pre-process the raw data to get valid data for our analysis task.

2.2.1 Harvesting strategy

In order to harvest as many tweets as possible, I applied three twitter api methods: **Streaming API**, **GET search/tweets** and **GET statuses/user_timeline.** I implemented three harvesters: *stream_harvester*, *search_tweets_harvester* and *user_timeline_harvester*, three harvesters harvest tweets at the same time.

Streaming API provides a real-time access to Twitter's global stream data. It has some advantages, for example, it can always collect the most up-to-date tweets data. The up-to-date data is usually more convincing in a research project. Besides, streaming api has a less strict rate limit. In fact, if you don't keep running and stopping your stream harvester repeatedly, it is very hard to exceed the rate limit. Furthermore, the **filter** function supplied by streaming api is very powerful to search tweets about certain topic or from certain location. On the other hand, it can only collect 20,000 ~ 30,000 tweets per day across Australia and most of them are of no use for our topics. Apart from the number of tweets, the **filter** function is kind of weird, **filter** function in **Twython** (which is a Python implementation of twitter api) can take several arguments such as 'track', 'locations' and so on. I found that if only one argument is given to **filter** function, it works well. However, if 'track' and 'locations' are given at the same time, it collects tweets contains words in 'track' **OR** tweets from the 'locations'! That is to say, if I set **filter (track = 'python', locations = Australia)**, the harvester will collect tweets which contain 'python', but may be sent from the United States instead of Australia. Another limit of streaming api is that it cannot get retweets if the 'locations' argument is given, this is because retweets does not contain any location information.

Rest Api search method can collect the last 7 days tweets, compared to the streaming api, it can get a large amount of data in a very short time. Besides, it can get retweets, I guess the reason maybe it gets 'location' information from the 'user' information part. It's rate limit is strict, only 450 requests every 15 mins for Auth2. When the harvester exceeds the rate limit, it must wait 15mins. Commonly, we can get at most 100 tweets by each request, which means we can collect at most 45,000 tweets every 15 mins.

Statuses/user_timeline is a very powerful method to collect tweets data. It allows us to get all the tweets from a specific user. It takes user_id or screen_name as the input and returns a collection of the user's most recent tweets. The rate limit of it is 1500 request every 15mins, and each request can get at most 200 tweets from a user.

This method can provide us a very large number of data without time limits. We get the user_id from the data we collected by using above two methods, which can ensure the user is still active and the user is in Australia. The drawback of this method is that some very old data has little research value.

2.2.2 Data Pre-processing

We have collected about 2 million tweets from Australia, since some of tweets are useless for our sentiment analysis tasks, I pre-processed the raw data to simplify the analysis job. At the harvesting stage, I added "**emotion**", "**topic**" and "**location**" tags into the raw tweets data. "**emotion**" and "**topic**" are set to be an empty string, which will be filled in after the tweet is processed by sentiment analysis. "**location**" is set to one of top 5 largest cities if I can find any city information from "place" or "user", if the tweet is not from one of top 5 largest cities, "**location**" will be set to "Australia", thus we can use CouchDB methods to pick out all tweets from a specific city.

Besides, a filter program is implemented to help pick out valid tweets for specific analysis topics such as health and immigration. It is implemented by using regular expressions to match tweets with our key words list for each topic.

2.2.3 Data Storing

CouchDB-python is a Python package for accessing and manipulating CouchDB with Python code. We applied a single node setup, since a single node setup can meet our requirement for this assignment, and compared to cluster setup, it is easier to manipulate and maintain. We created two databases: '**australia**' and '**racist**'. The '**australia'** stores all tweets we collected from Australia, which is used to do topics such as health, immigration and hot topics in top 5 largest cities, while '**racist'** stores some possible racist users' timelines.

2.2.4 Remove duplicate data

As I mentioned before, I applied three harvesters to collect tweets at the same time, and I also added some data from ***bigtwitter.json*** into our database. It is inevitable that some tweets might be collected more than once. To prevent duplicate data arising, I took advantage of the uniqueness of the "**_id**" for each document in CouchDB. As we know, every tweet has a unique **id** as well, so we set document "**_id**" as tweet "**id**". If a tweet has already been stored in the database, when the harvesters try to put the same tweet into the database again, CouchDB will throw an exception and abort this operation. The harvester will catch the exception, and continue its job. Therefore, there is no possibility that a database contains duplicate data.

Some people believe retweet is also a kind of duplicate data, but I don't agree with that, since if a user retweeted a tweet, that means the user agreed with what the tweet said. So, I believe retweet is also a way to express the user's attitude on a topic.

2. 3 NeCTAR Research Cloud

NeCTAR is an online infrastructure provided by The National eResearch Collaboration Tools and Resources project, which supports researchers to connect with colleagues in Australia and around the world. This is an excellent platform where researchers can collaborate and share ideas and research outcomes with each other.

2.3.1 NeCTAR deployment

In order to make full use of the resource that tutor allocated to our team on NeCTAR, we launched four m1.medium sized instances: instance1, instance2, instance3 and instance4. We placed CouchDB on instance4, and the other 3 instances are used to run the harvester programs. We also attached 100G volume to instance4 to store the tweets. *(Figure 2).*

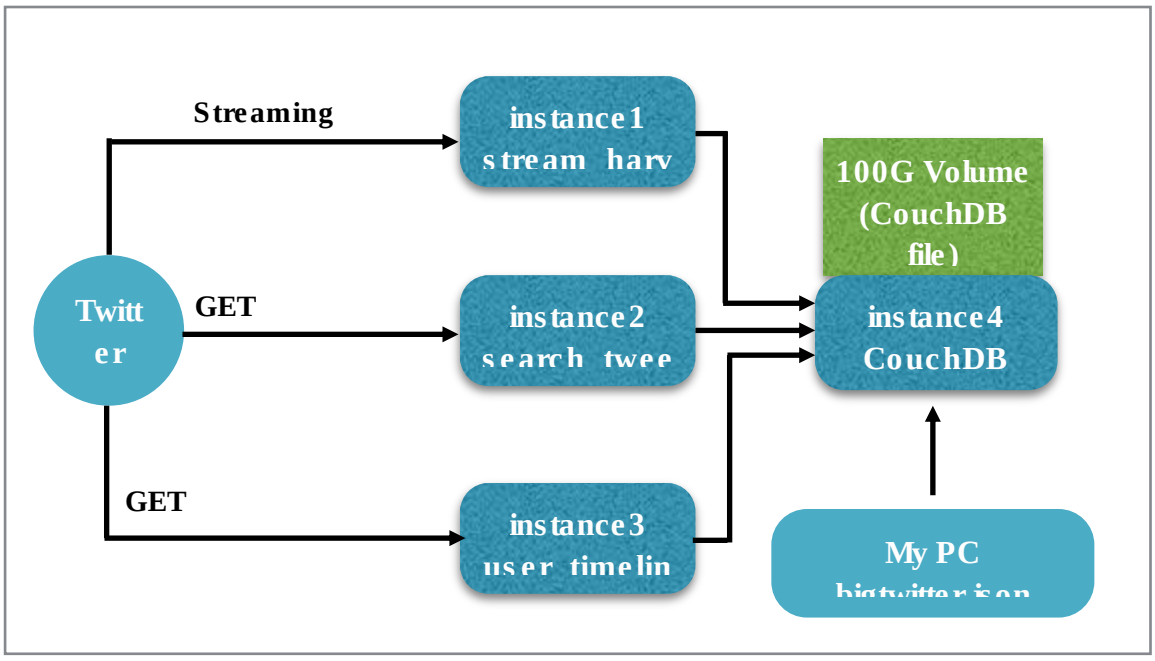


Figure 2. NeCTAR deployment.

2.3.2 Discussion of NeCTAR Research Cloud

2.3.2.1 Advantages of NeCTAR Research Cloud

- **Cost:** Compared with local machine, NeCTAR needs less cost. It can be viewed as pay-as-you-go computing, you pay to rent its service and resource only when you need it. This is much cheaper than buying and maintaining a local device.
- **Resource:** NeCTAR can provide researchers almost unlimited storage and very huge computing powers, and these resources can be scaled up and can be accessed immediately. NeCTAR is also a great platform to sharing resources with each other.
- **Security:** NeCTAR provides good access control to ensure the security of the system. In this assignment, the system was encrypted with RSA encryption, we generated our own key-pair to access our instances.

2.3.2.2 Disadvantages of NeCTAR Research Cloud

- NeCTAR only supports the linux operating system, and the instance does not provide a graphic user interface, for a beginner to linux system, it's very hard to move forward.
- It requires the internet to access. Thus, you can't access NeCTAR if you don't have an internet connection.

2.3.2.3 Tools and Processes on NeCTAR

There are some very useful and powerful tools for NeCTAR, such as BOTO and ansible. These tools are easy to install and can help researchers save a lot of time by avoiding doing repeated deployment. Furthermore, these tools can help guarantee the consistency between nodes and the stability of the whole system. However, as a beginner to NeCTAR, these tools make me confused some time, I need some time to learn how to use them.

2.4 Server

2.4.1 Purpose

According to the requirement of this project, actually server is an optional component in the architecture for every functional part can directly work with the database. However, a server can simplify the design and implementation of other parts on parallel and coordination in order that they can focus on their own functionality. Thus a server is necessary both for the completeness of architecture and the simplification of implementation.

Therefore, the functions of server should include:
1. Perform operations on database (read, write, update, get certain view result etc.).
2. Accept http request and perform suitable response.
3. Promise the consistency and performance of concurrent access by different functional parts.
4. Dynamic setting parameters without shutting down.
5. Display server's current state, instructions.

2.4.2 Technique and Implementation

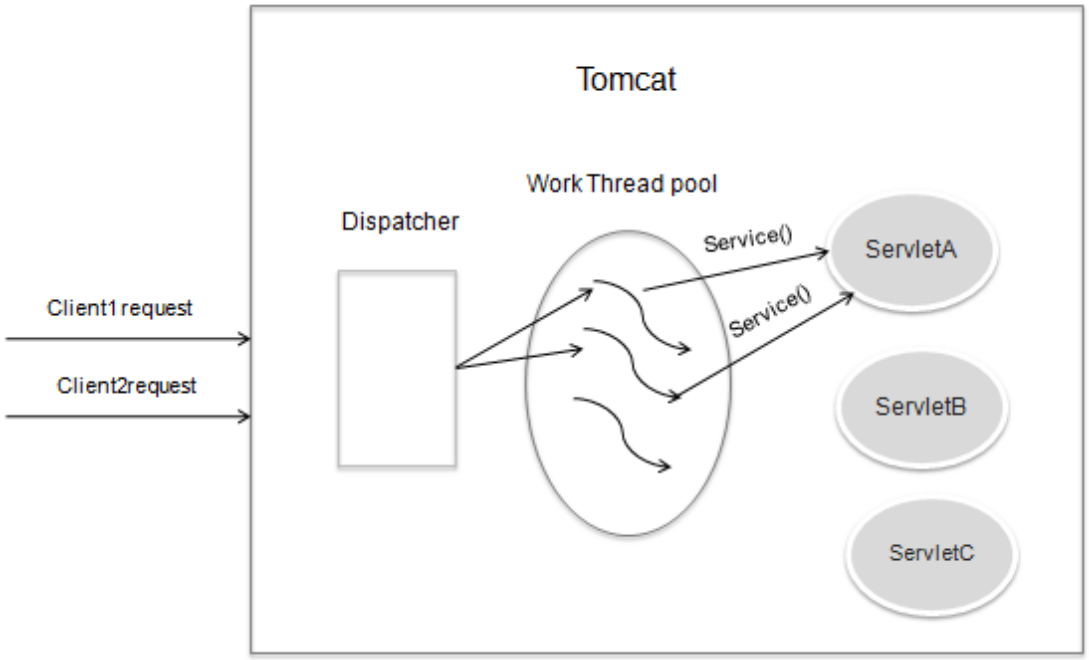
The server is operated on Apache Tomcat as servlets, using RESTful API, implemented by java, import some third parties' jar package (JSON and Apache client) and deployed on cloud server. Each part will be discussed in detail in the following sections.

2.4.2.1 Apache Tomcat and Servlets

Apache Tomcat software is a free open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies developed by Apache Software Fundation (ASF).[1] Different from Apache server, Tomcat is developed by java and as the extension of Apache server, Tomcat supports dynamic request (e.g. JSP and Servlet). According to the context and requirement of our project, Tomcat is more suitable than Apache because the result is updating and we need to display the result dynamically.

Generally, Tomcat server works as a container of servlets and the servlets is the real functional solution to the requirement. The relationship between Tomcat server and servlet is shown as below:

[1] http://tomcat.apache.org/



Tomcat uses thread pool to deal with the concurrent access. The settings of port and thread usage can be configured in the web.xml file. After the Tomcat server started up, the server listens the income URL requests and uses the configuration to map the requests to certain servlets. The instance of one servlet is created once and only once since the first time it was visited and the instance can be access concurrently by multi-thread. Thus, the solution of concurrent access is simplified and the only thing left is to promise the consistency of writing and the uniqueness of distributing.

2.4.2.2 Operations with database

As the requirement of the assignment, we should use CouchDB in the system. URL is used to access the database, which is the same as accessing the server. In our system, the harvester is a separate part and the server is response for distributing raw tweets, collecting results and delivering result views to the display panel. Hence the operations on database can be categorized into three parts: filter unsolved data, update results and MapReduce for display.

The first section happens when workers require raw tweets. In our system, we add additional key-value pairs to store the results, such as emotion indicator and topic words. Based on such structure, raw tweets and solved tweets are stored in same database. Therefore, a view is necessary to filter what are needed according to the requests from different workers. These views do not need reduce functions and the result they emit is an array of JSONObject contains document ID and text of tweets. However, the result set may too large to be delivered in a single request. Thus a limitation is needed. Additionally, if there are multiple workers work on the same topic, in order to avoid provide repetitive raw tweets, a synchronized parameter is

needed to indicate the skip length. To implement this, every time a request to a certain topic has been made, the parameter records the data size and when next time a request comes, the result set will skip a size of parameter's value. In contrast, when a worker posts the result of certain topic, the parameter will decrease itself by the size of result. The offset parameter should be synchronized in order to promise the uniqueness of data sets for different workers.

In the second section, the server receives the result set from workers and updates them to the database. As described above, when receiving the result set, the offset parameter will change itself. Besides that, the server also need to get tweets by ID in the result set from the database again. The reason is in our system, the data set workers received only contains ID and text and the result set only contains ID and a result key-value pair. This approach is adopted to reduce the network delivery pressure and the complexity of workers' program. Another reason is that CouchDB has a mechanism to protect the write consistency and a special rule when updating a document. The mechanism is that every document has a _rev attribute and every time the document is modified, it will change. The special rule is that when updating a document, it is a completely replace, neither be an insert nor a modification. Hence when the server gets the result, it needs to request the newest version of document, inserts the result key-value pair and sends the complete document to the database to update.

The third section is the key part of our analysis and dynamic result display. The map function works in the same way as that in the first section. The only different is in this section the map function should emit an array of attributes as key and a number (usually 1) as value. Then the reduce function calculates the value by certain rule and produces a statistical result. The view showing the statistical result usually contains both map function and reduce function. Additionally, for the key of map can be an array, a group_level is used to indicate the number of attributes that needed to be calculate. Also the view will update when the database has any changes (insert, update, delete). Therefor every time the display panel requests the result, the server will reply an up-to-date result.

2.4.2.3 Java implementation

The servlet is implemented by Java using eclipse EE. The imported third party jar packages include JSON.jar and Apache http client. JSON.jar is adopted to analyze JSON files because the format of message between parts is JSONObject. Apache http client is adopted to enable the server can send http requests to CouchDB as a client.

In programming, multi-thread is not necessary for the Tomcat is multi-thread based and the mechanism has already been established. The only thing needs to pay attention is the synchronize part. In this server, the offset counter and post method

have been synchronized to guarantee the consistency and uniqueness, which have been discussed above. Another important part in programming is the analysis of URL. Following the RESTful API, the URL contains desired function and necessary parameters. The server can perform accurate mapping to the servlet and make suitable response according to the information contained in the URL.

2.4.2.4 RESTful API

RESTful stands for representational state transfer, was first introduced and defined by Roy Fielding, is a style of providing interoperability between computer systems via internet. In this project, I adopt RESTful as the style of my server's URL to achieve a better user experience and easier way to access. All the URLs can be categorized by functions in three types: state illustration, parameters setting, data/result distribution and collection.

State illustration is a general snapshot of server, including current target DBs, inner parameters and instructions. URLs of this type provide a simple way to know the state of server, making debugging and monitoring easier.

Parameters setting URLs supplies an interface of accessing and modifying the inner parameters of the server. Based on this function, we can simply change the source DB, add topics, manage the size of messages, set offset parameters and handle re-deployment of workers without shut down the server. This approach guarantees the stability and continuity of whole system.

Data/result distribution and collection are the ordinary get and post methods, every desired function can be mapped to an URL to perform appropriate operations on database.

The whole URLs instructions are shown by the API table below:

Description	URL	Format
General:		
ADDRESS	http://IP:Port	
Root	ADDRESS/cloud	
State	Root/state	
Instruction	Root/instruction	
Get:		
Raw tweets	Root/twitter/(topic)	[{"id":(id),"key":(id),"value":(text)}]
Result	Root/result/(view)/level=()	{“rows”:[{“key”:[],”value”:()}]}
Hot topic	Root/location/(location)	[{"id":(id),"key":(id),"value":(text)}]
Set counter	Root/setting/(topic)/(topiccounter)=()	flag
Set size	Root/setting/(topic)/(topicsize)=()	flag
Set source DB	Root/setting/source/(address)=(name)	flag

Set result DB	Root/setting/result/(address)=(name)	flag
Add topic	Root/addfunction/(topicname)	flag
Delete topic	Root/deletefunction/(topicname)	flag
Post:		
Result	Root/result/(topic)	[{"id":(id),"(topic)"=()}]
Hot topic	Root/location/(location)	{(word):(count)}

2.4.2.5 Deployment on cloud

The deployment of server is simple. JDK and JRE are necessary and need to be installed in advance on the cloud instance. The whole dynamic web project should be exported as war file and put into the webapps folder of Apache Tomcat server. System variables and environment variables (like JAVA_HOME and TOMCAT_HOME) should be configured in advance and for there are no easy way to set those variables on the instance on Nectar, I wrote the path in the startup.bat file. After these configurations, upload it, run the startup.bat file and enter the state URL in browser to check whether the server is working.

2.3.3 Improvement

Although the server works well now, there are still some points need to be improved.

As discussed before, the message between workers and server only contains necessary information. However, when the system starts to operate, the bottleneck is the speed of storing result into database because for each result, the server needs to request the whole document and restore it again. This approach must be synchronized or there will be data lose. Thus the writing process is working like a single thread. The better way to do this is separating the source DB for each topic and passing the complete document in order that when the result come back, the server can upload the documents in bulk.

As limited by the storage process, the multi-worker mode become meaningless because all the workers stuck in the uploading process. Actually in this case, the synchronized block make the case worse for the servlet only have one instance and the synchronized block for different topic will block each other's access. The solution is to separate the functions into more servlets in order that synchronized blocks in different topic work on different servlet instances and no longer interfere each other.

Finally, the synchronized block may not necessary if the worker's producing speed is much higher than the read and write to the database. The reason is that if the _rev consistency cannot be promised, the system can ignore the unsuccessful update and

let the workers do it again for they are fast. This assumption will work even better if the source DBs for different topic are separated.

2.4 Front-End Design

Based on sever-end Node.js and front-end google chart Javascript package, the web application can display the results in different kinds of charts, which makes it easy for users to read. To make the implementation of this web application simple and fast, the framework of Express is used, which simplifies the router configuration. When it comes to the route design, RESTful is applied and the the router rules can be shown as below:

Route Design

Name	URL	Verb	Description	Content
INDEX	/au	GET	Display the results at the scope of Australia	- Happiness Degree among Australian cities - Attention Degree towards immigration and health among Australian cities
SHOW	/au/mel	GET	Display the detailed results at the scope of Melbourne	- Hot topics in Melbourne - People's attitude towards immigration and health.
SHOW	/au/ade	GET	Display the detailed results at the scope of Adelaide	- Hot topics in Adelaide - People's attitude towards immigration and health.
SHOW	/au/syd	GET	Display the detailed results at the scope of Sydney	- Hot topics in Sydney - People's attitude towards immigration and health.
SHOW	/au/per	GET	Display the detailed results at the scope of Perth	- Hot topics in Perth - People's attitude towards immigration and health.
SHOW	/au/bri	GET	Display the detailed results at the scope of Brisbane	- Hot topics in Brisbane - People's attitude towards immigration and health.

Server is listening to users' requests. For this application, the display-server only handle "GET" request and set routes for them. When users want to retrieve the results, which are html files combined with CSS and Javascript, they type in accurate URL address and send it through HTTP protocol. In response, the server will generate the corresponding html file and send it back. In this application, Embedded JavaScript templates is used to generate html file, which takes the role of binding the data from server-end to front-end.

2.5 Automatic System Deployment and Configuration

The automatic system deployment and configuration is basically accomplished with BOTO and Ansible. To be specific, BOTO is mainly used for creating new instances, creating and attaching volume to an instance, and writing our a hosts file for Ansible to use. On the other hand, Ansible is used to automatically install and configure the software environment and the CouchDB. Moreover, Ansible helps deploy and run the harvester, the sentiment analysis and hot topic analysis, as well as deploy and start our web server to show our results

In this project, four instances are created, one of them is deployed with CouchDB and cloud server, the other three instances are used to run twitter harvesters and analysis program.

In order to automatically deploy a ready-to-run system, at first user have to install ansible on their local machine, type the command below in your command line interface:
$ pip install ansible.

After Ansible is installed on you computer, run ansible-playbook -i hosts ‘master.yaml’ to deploy the whole system. You can run part of the system as well through running other yaml files.

3. Data Analysis

3.1 Sentiment Analysis

When people post tweets, they usually want to, more or less, express their personal emotion towards some stuff, like recommendation, admiration, sharing the moments making them happy or sad, criticizing some events or politicians etc.. To know whether a tweet is positive or not is not only interesting, but also worthy since it can produce useful information to be further analysed for event prediction, market analysis, public security and so forth. Consequently, the sentiment analysis of tweets plays a crucial rule on the data analysis.

3.1.1 Basic Workflow

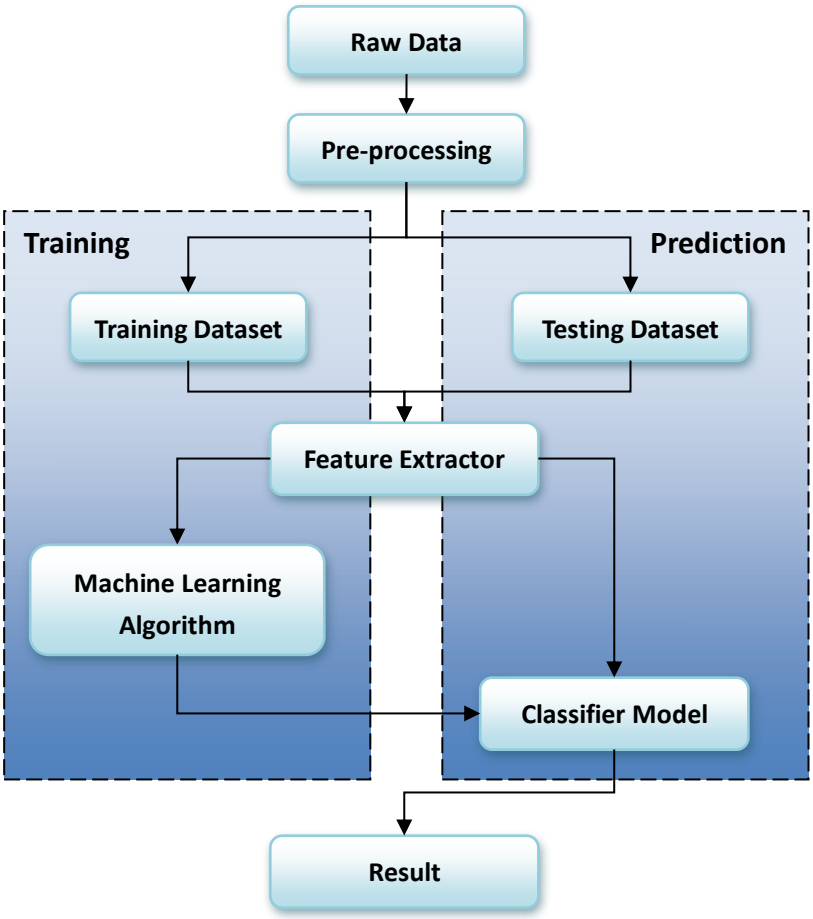


Figure 1. Overall workflow of the sentiment analysis

The raw tweet data can be obtained from the Internet or manually gathered and classified, which contains the raw tweet content and a classified label in each record. Normally, for the purposes of reducing the computational process in the following

steps and handling the encoding problem of the tweets, it is better to pre-process the raw tweet data before training and prediction. Then, the whole dataset will be divided into training and testing dataset, with a proportion of 9:1 in their tweet number.

In the training section, the feature extractor will take the training dataset as its input and count the word frequency to figure out the "features" for each label(e.g., positive and negative.). These "features" can be one word(also called unigram), two continuous words(also called bigram), multiple continuous words(also called ngram) or even any combination of them(e.g., the combination of unigram and bigram). Next, the algorithm will select a set of the most popular features of each label and label each word in each record of the training dataset if the word is in the popular features set. Then, the labelled training dataset will be delivered to the machine learning algorithm for training. Ideally the machine learning algorithm will learn the inner correlation among these features and generate a classifier model to predict the label of new tweets.

The testing dataset can be used to know the accuracy of the generated classifier model. Similarly to the training dataset, the testing dataset will go through the feature extractor to get themselves labelled. In the next stage, it will be delivered to the classifier model to get the predicted label result. Finally, by computing the ratio of the number of right predictions to all records, we can identify the accuracy of the classifier model.

3.1.2 Method Comparison

The classifiers for comparison includes: Bernoulli Naive Bayes(BernoulliNB), Multinomia Naive Bayes(MultinomiaNB), Maximum Entropy Classifier(Maxent) and Support Vector Machines(SVC).

Our method comparison program, which is also our original sentiment analysis program, is implemented in Python with the library of nltk and sklearn, using the dataset provided by sentiment140(Alec G., Richa B. and Lei H., 2009). We extracted some number of tweets with equal number of positive and negative ones, and divided them into training and testing dataset with the proportion of 9:1. For example, for 10,000 tweets samples, the number of both positive and negative ones is 5,000 and 4,500 for training and 500 for testing respectively. Besides, we combine unigram and bigram for feature extraction.

Table 1 indicates accuracy changes for different machine learning methods under various number of dataset. And the trends are shown more clearly in Figure 2.

As indicated in the graph, SVC has the worst performance in all the cases while the other three classifiers are similar. MultinomiaNB performs the best in the case of 10,000 tweets, with about 80% of accuracy. However, as the training and testing

dataset getting bigger, its accuracy decreases, so does BernoulliNB. Although the accuracy of Maxent also experiences a downward trend at first, it suffers less influence and its accuracy grows faster as the dataset size increasing. Thus, Maxent seems to be the best classifier if the dataset is big.

Accuracy		Total tweets number of training and testing set			
		10,000	20,000	30,000	40,000
Machine Learning Methods	BernoulliNB	79.20%	77.60%	76.33%	76.28%
	MultinomiaNB	80.10%	78.20%	76.50%	76.63%
	Maxent	79.30%	77.85%	77.43%	77.98%
	SVC	64.80%	61.90%	65.17%	64.35%

Table 1. Comparsison of classifiers

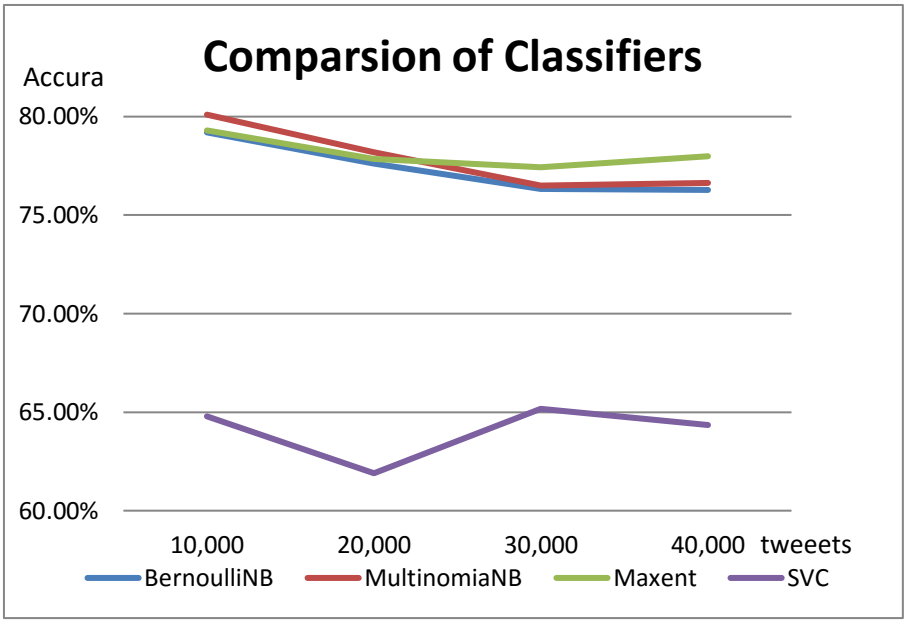


Figure 2. Comparsison of classifiers

3.1.3 Our Approach

In the process of searching the training dataset and sentiment algorithm, we have found an online sentiment system created by , sentiment140, to analyse the tweets. It can classify the tweets into three different types: “positive”, “neutral” and “negative”, while our original program can merely label the tweets with “positive” and “negative”(the training dataset only contains positive and negative tweets). Moreover, according to the implementation description[2] of this online system, it is trained by Maximum Entropy Classifier, which is the best classifier among four classifiers as we have discussed in the section 2. Plus, a size of 16,000,000 tweets training dataset was used to generate its classifier model, enabling the feature

[2] More detail at http://help.sentiment140.com/for-students

extractor accurately figure out the most related features for each label which made the classifier more precise.

As for its accuracy, after we tested it with a dataset of 1,000 tweets labelled with "positive", "neutral" and "negative", about 62% of them are correctly classified. Notice that three labels are used here, meaning that the correctly blind pick rate is only 33.3%. Thus, this online system almost doubles the correct rate, while our original algorithm only performs 1.5 times better.

Hence, with the consideration of more classified categories, a well-performed classifier, a huge training dataset and a good accuracy, we decided to use the online analysis system instead of our original one.

Figure 3 shows the overall process of our sentiment analysis. It is similar to the basic workflow shown as Figure 1, with additional interaction with database. However, in order to achieve a higher accuracy, we added emoji and emoticon sentiment analysis in the pre-processing section.

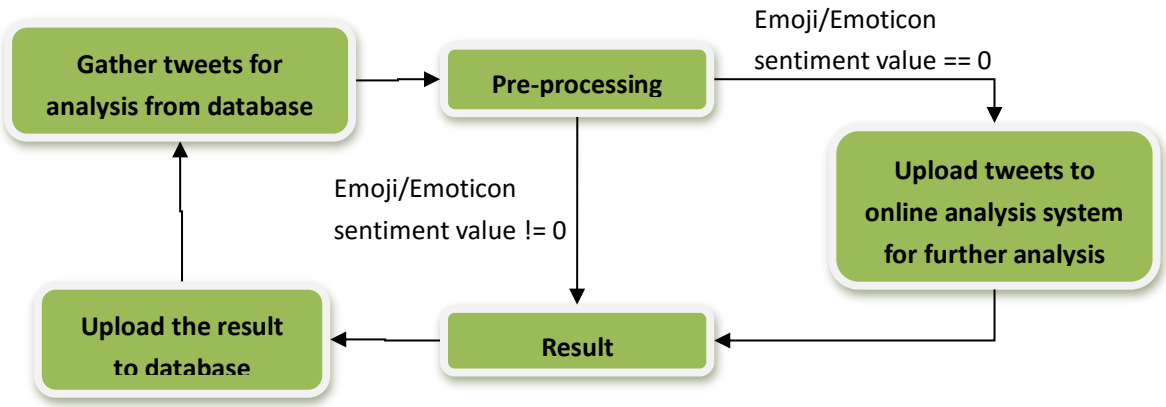


Figure 3 The workflow of our sentiment analysis

3.1.3.1 Pre-processing

Pre-processing is important because it can handle the encoding problem and significantly reduce the data amount to be processed in the following steps. Figure 4 shows its workflow.

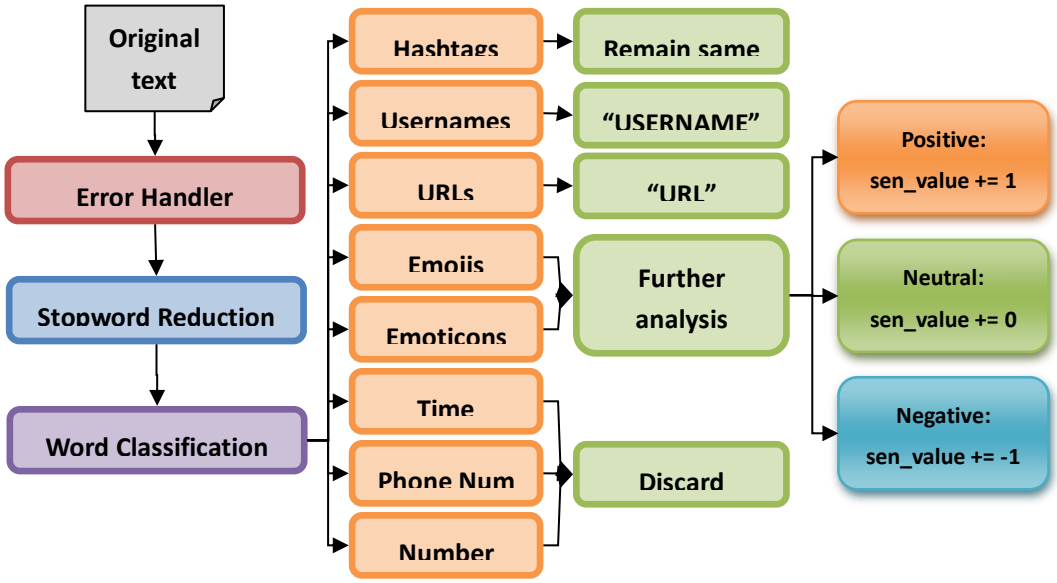


Figure 4. The workflow of pre-processing

1) Error Handling

Twitter is well-known as a global instant communication software. As a result, it may contain different languages encoded in different mechanism, especially in the multicultural country Australia. It is not uncommon to find out a tweet comprised of English and other languages at the same time. To overcome this problem, our program will try to convert each word in the text into Unicode. If it is unsuccessful, just discard it(note that we only focus on English).

2) Stopword Reduction and Word Classification

Some common words may have no useful information for further analysis, such as "it", "this", "I" etc., which are also called "stopword". In the pre-processing, every tweet will discard these words by going through a list of 127 stopwords.

Furthermore, our pre-processing utilises regular expressions to break every single tweet into segments and classify them into different types, including "hashtag", "username", "URL", "time", "phone number", "number", "emoji" and "emoticon". And they can be replaced by whatever words needed. For example, all the words with the type "username" can be replaced to the string "USERNAME" instead of its original form(e.g., "@cloud_computing"). In our program, all usernames are replaced as "USERNAME" and all URLs are transferred to "URL", while "time", "phone number" and "number" are discarded.

3) Emoji and Emoticon Sentiment Analysis

Our program can recognise 185 emoticons(e.g., ":)", ":(", ">_<") and all emojis(e.g., 😀, 😘, 😒). An emoticon can be classified as "positive", "neutral" and "negative" by checking positive and negative emoticon lists(manually collected

from Wiki), and about 1,000 emojis can be also classified by computing the possibility(based on a table[3] found on the Internet). Once an emoticon or an emoji is classified, a sentiment value of 1(positive), 0(neutral), -1(negative) will be added to the tweet. If the sentiment value of the tweet is greater than 0 after pre-processing, it will be directly classified as "positive"; if the value is less than 0, it will be "negative"; if the value is 0, it will be sent to the online sentiment analysis system for further classification.

We extracted the tweets with emojis and emoticons to test its accuracy, and 86% of correct classification rate was achieved, which is a great outcome.

3.1.3.2 Result

Each tweet will be granted an integer to represent its label. Negative tweets are "0", neutral tweets are "2" and positive ones are "4". The result will be sent to the database for further data analytics and the program will come back to data gathering again.

3.2 Hot Topic

When the system gets numbers of twitters to handle, it may need to find out which topics people mostly care about. Hot Topic function is to extract the keywords of each twitter and finally find out the top interesting topics from the data. Result will be shown in the scenario section, and the workflow is shown as below:

[3] https://dematerializer.github.io/emoji-sentiment/emoji-sentiment.stable.html

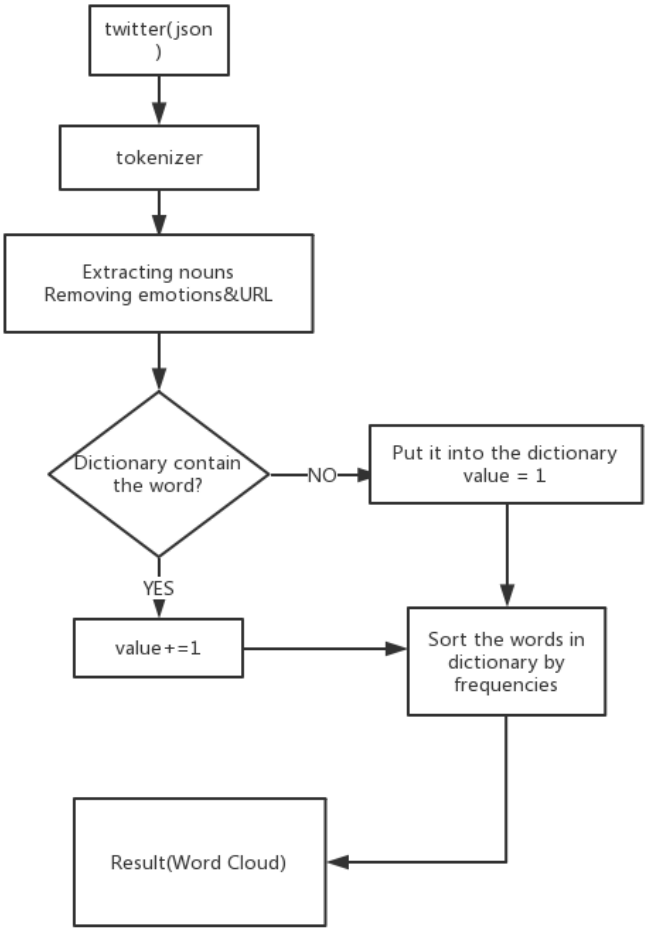


Figure 1.flow chart of Hot Topic function

3.2.1 Pre-processing

Pre-processing is to handle the encode and decode problems(for example,utf-8, gbk, unicode transformation problem)and to get rid of the words which obviously can not be the topics. In the program, it will change all the text to Unicode and then tokenize them to prepare for the next step. This step will be explained later in this report.

Moreover, the pre-processing will also extract the nouns and phrases from the text to improve the efficiency of the program. All the emoticons and URL will be removed from the text at the same time because apparently they can not be helpful to get the hot topics.

3.2.2 Code transformation

It is widely known that twitters may contain different kinds of languages so that the code might be different. But in the program, the code has to a single type to save into the database.

To solve this problem, it can not be simply using the function “encode” or “decode” in python to convert them from one to another.

Unicode, which can be called python 3’s string, is often been used to take care of the text.It should be noticed at first that Unicode is not encoded. On the other hand, string is just a sequence of bytes.In python 3,it simply calls string “bytes”.Therefore,Unicode can be regarded as a general way to represent a text which is also the most appropriate one to be used in Hot Topics function.

UTF-8 is a character encoding capable of encoding all possible characters, or code points.It is an effective way to represent Unicode. While GBK is specialized for Chinese and it can use less bytes to represent Chinese data than average, but more for other languages.

Above all,the program should use convert the texts in Unicode first,and then handle them in different ways.

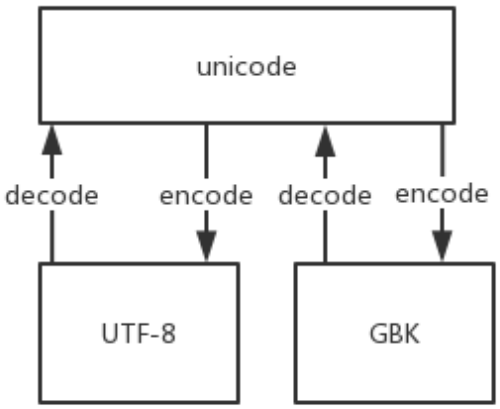


Figure 2.Relationship of Unicode&UTF&GBK

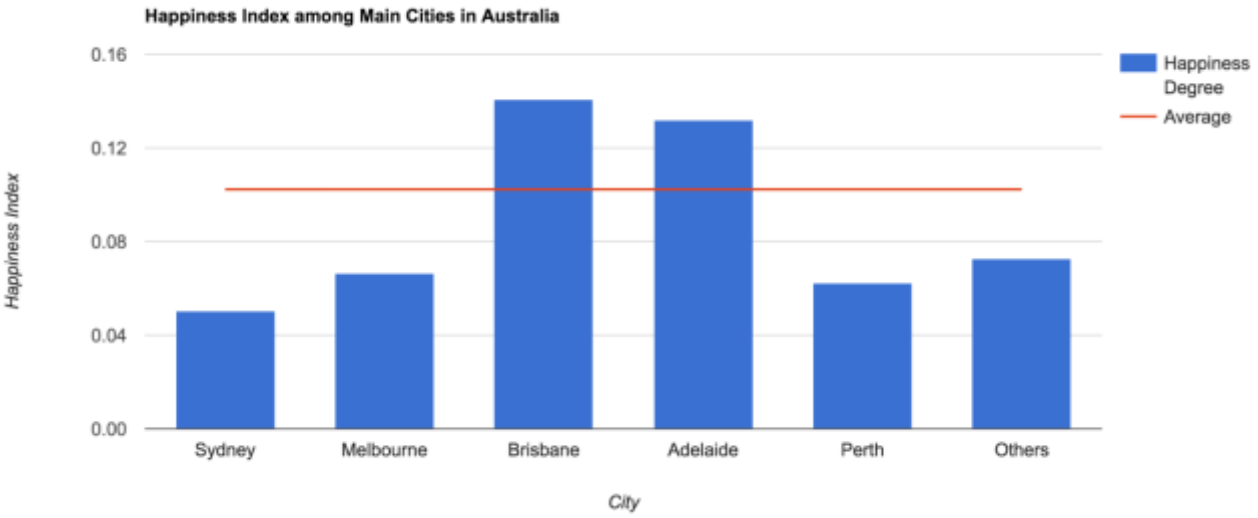
4. Scenario Analysis

The object of this system is using twitter data to illustrates which city of Australia is most suitable for immigrants moving in. There are five candidates in scope, which are Sydney, Melbourne, Brisbane, Adelaide and Perth. Three variables are made into consideration, which are happiness index, attitudes towards health and immigration and hot topics in specific cities.

4.1Happiness Index of People in Australian Cities

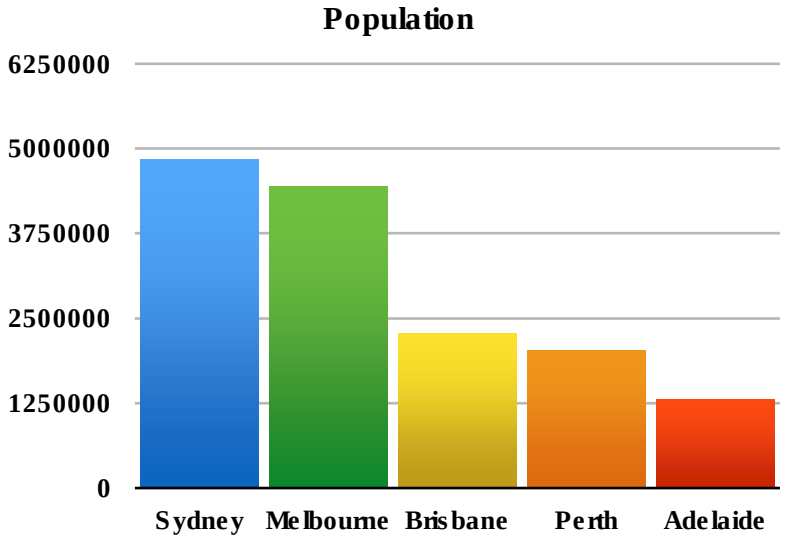
In this scenario, we assume the sentiment value of each tweet represents the owner's feeling towards specific topic, which can be positive, negative or neutral. To a large scope, if more people feel good in a city, this city will be better than others even people's feelings are towards different topics. Based on this point, we generate 'Happiness Index', which equals to the number of positive tweets dividing the total number of tweets in this area: num(Positive)/num(Total). To get this number, we

harvest the tweets within Australia and analyze their sentiment values. The result is



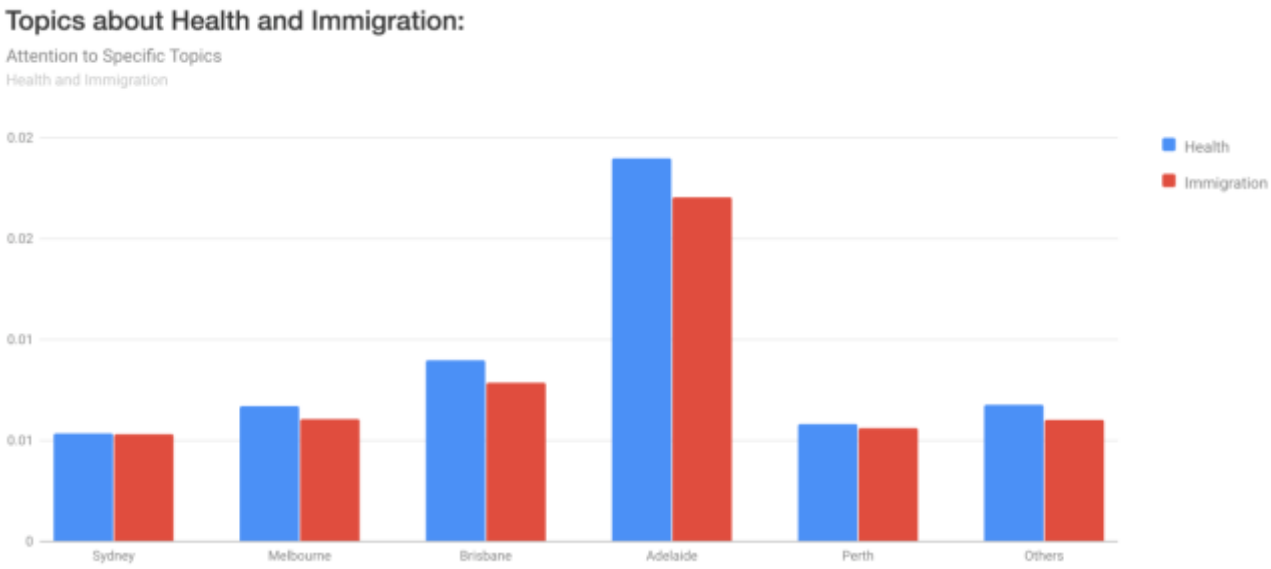
shown as below:

In this bar chart, we can see that more people in Brisbane and Adelaide have positive feelings, with the number 0.14 and 0.13 respectively. These numbers are significant larger than others, which increase the average happiness index of Australia. According to the population data from Aurin, On the other hand, although Sydney has the largest population, people there feel worse than any other city in Australia. This result exposes the problem that people living in larger cities may not feel better. On the contrary, they are faced many problems result from the large population, for example traffic jam and high rental fees. Therefore, in our scope, we believe Brisbane and Adelaide maybe two better choices for immigration.



4.2 People's Attitudes towards Immigration and Health

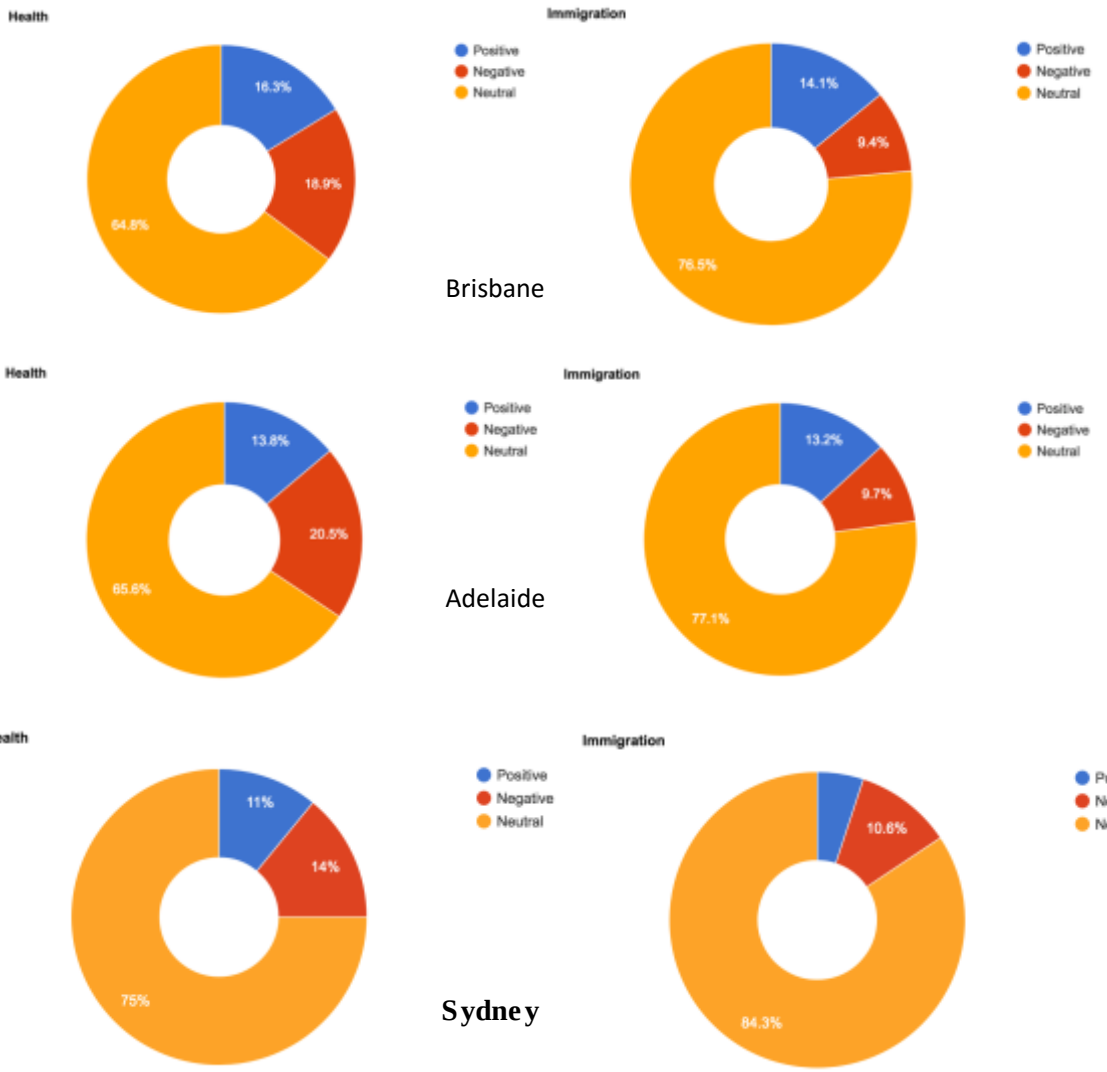
In this scenario, we want to explore which city is the hottest place for immigration. To deal with this problem, it is assumed that if more people in one place talk about a topic, people their care more about this topic and further they have a closer connection to this topic. Therefore, in out scope, we get the value by using the number of immigration related tweets to divide the total tweets in a specific area: num(Immigration)/num(Total). To achieve the first number, we analyze each tweet by using a topic related word dictionary. In this way, each tweet is assigned with true or false to specific topic. Then the number is counted from database. The result is



shown as below:
We can see from the bar chart that although the number varies in different places, they are all quite small. This result indicates that immigration and health are both small topics in twitter. This result is quite different from our thinking. Because during the twitter harvesting period, the immigration policy changed a lot, we expected more people talked about this topic in twitter. However, it shows that at least Australian twitter users do not care about it. As for the topic of health, the reason to choose it is that we believe wellness is a popular topic in Australia. However, it shows people who use twitter care about wellness even less than immigration.

Even though these number are all small, difference exists in them. Based on our assumption, people from Adelaide are more concerned with immigration and health compared with other cities, which is followed by Brisbane. These two cities are still the top two in our scenario of immigration. However, people talk about it does not mean people like it. For example, if people in one place all dislike immigrants, the

immigration related topics may be hot, but this area is not the ideal place. In this way, we need to get people's attitudes towards these two topics.



Based the work above, it is needed to check people's attitudes towards immigration in these five cities. When comparing Brisbane and Adelaide, the former has a better performance than the latter with more positive and less negative tweets. However, as for the topic of health, the negative tweets are more than the positive ones in both of these two cities. This phenomenon also appears in other three cities in Australia. There are many possible reasons for this. For example, people are more likely to complain about illness rather that declaring they are all good.

In order to find the best place for immigration, we care more about how people think about immigrations related things like immigrants, policies and multi-cultures. In this scope, Brisbane has the largest ratio of positive ones. Especially, when compared with larger cities like Sydney and Melbourne, this advantage become more obvious.

In conclusion, based on these two scenarios mentioned above, we believe Brisbane and Adelaide are the best two places for immigration. People in these two places are more satisfied with their lives and have more positive attitudes towards immigration. However, when people go to another city, they should know more about their

destinations' cultures, which can be get from their most common topics. Here comes to the third scenario, which is main topics of different cities.

4.3 Main Topic of Different Cities:

In this scenario, we analyze each tweets from different cities to collect the most frequent word of each city. In our assumption, different from the topic with '#' in twitter, which represent some special trends, these words we collected represent the key words of people's everyday life, which offer a clue of their own culture.

Sydney:"Trump","Australia","Sydney","media","music" and "health"
Melbourne:"Trump","Australia","Melbourne","music","game","education","rain" and "wind"
Adelaide:"Trump","Australia","beach","dance","game" and "media"
Perth:"Trump","Australia","Perth","house","music","weather"and "free"
Brisbane:"Trump","Australia","Brisbane", "beach" and "food"

Form the result, we can easily find out that "Trump" is the hottest topic in most of the cities. People in each city care more about Australia and their own city's news. However, because of different life styles and culture of each city, the hot topics vary from one to another. In Sydney, music and health are more interesting topics.

Melbourne people care more about music, game and education.The weather in Melbourne might be rain and wind regularly in most of year. On the other hand, the people in Adelaide are more likely talking about dance beach and game.This may be caused by the location of the city and the life style of people. Perth includes “weather” and “free”,which might means that the weather in Perth changes a lot during the season or even in a day. For Brisbane, suffering and eating are their styles.

This scenario provides people a better understanding of each place’s culture and contributes to the decision of immigration.

In conclusion, after the analysis of these three scenario, we believe Brisbane and Adelaide are better choices for immigration and people may make their own choices based on the culture of each city.

Reference

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

[2] Dematerializer.github.io. (2017). emoji sentiment data v1 (stable). [online] Available at: https://dematerializer.github.io/emoji-sentiment/emoji-sentiment.stable.html [Accessed 1 May 2017].

[3] En.wikipedia.org. (2017). List of emoticons. [online] Available at: https://en.wikipedia.org/wiki/List_of_emoticons [Accessed 1 May 2017].

[4] Mishne, G & Glance, N (2006), "Predicting Movie Sales from Blogger Sentiment" , AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs

[5] Dowling GA, Burr RL, Van Someren EJW, Hubbard EM, Luxenberg JS, Mastick J, Cooper BA(2008) Melatonin and bright light treatment for rest-activity disruption in institutionalized patients with Alzheimer's disease. J Am Geriatric Society 56:239-246.