AUSTRALIAN CITY ANALYTICS REPORT

For COMP90024 Cluster and Cloud Computing Assignment 2

Abstract

City: Melbourne

The report describes, analyses and discusses about various aspects of the Australian City Analytics system we built for Assignment 2 of COMP90024. Topics covered includes but not limited to scenarios analysed, system architecture and design and any problems occurred during development.

Github repository: https://github.com/k-vikingsson/ccc2017-29

Youtube Video: https://youtu.be/4tLyj2V8MXU

Team 29:

Hangyu XIA <802971> <hangyux@student.unimelb.edu.au>

Hanwei ZHU <811443> <hanweiz@student.unimelb.edu.au>

Jinchao CAI <838073> <jinchaoc1@student.unimelb.edu.au>

Wenzhuo MI <818944> <miw@student.unimelb.edu.au>

Zequn MA <696586> <zequnm@dimefox.eng.unimelb.edu.au>

TABLE OF CONTENTS

1. Introduction

The aim of our system is to analyse a very large number of tweets extracted from Twitter for various patterns of various scenarios at various cities in Australia. Our system can analyse tweets from all around Australia and the cities are defined as cities of statistical area level 3 (SA3) as defined by Australian Bureau of Statistics. However, for our report, only visualisation and analysis of the greater Melbourne region. More information can be found from our front-end interface: http://mzalive.org/CCCFrontEnd/

Any error handling made in our project will be discussed in Section 5. Discussion under each subsection. ·

2. System Scenarios

2.1. Sentiment Analysis of Australian cities

As a minimal requirement, and we find that it is quite interesting to know the variation of people's emotion in different areas of different Australian cities. In social media (such as twitter, for this project), people tend to express themselves freely than in real life. It means that the data collected from these social medias is sufficiently valid for sentiment analysis and may lead to some significant conclusions.

In this project, we have built a system which will give each tweet the sentiment value as a specific attribute (from 0 to 1, which stands for extremely negative or positive). According to the tweets that we have harvested, we develop the method to illustrate the data from different perspectives which we are interested in. As the dataset is sufficiently large, the result will reflect the emotion much more accurate.

This scenario will also be compared with the AURIN dataset of people's salary in Victoria to see if there is any correlation.

We divide the sentiment analysing task into two parts: sentiment analysis from different area and from different time sequences respectively. The first part is trying to figure out how people's sentiment values fluctuate in different weekdays and in different hours in a specific city. The second part is trying to compare the variance between different cities in Australia.

In the first part, we have analysed all cities around Australia with their average sentiment values, and finally visualised on the Australia map. Figure 2.1.1 is an example of Melbourne city.

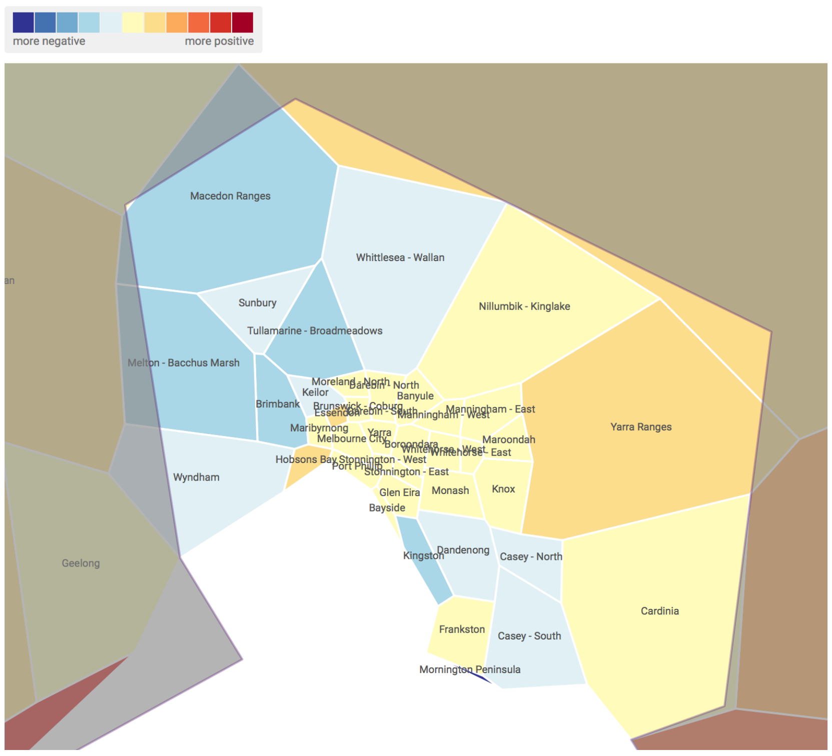

Figure. 2.1.1

From the screenshot on previous page, it can be observed that tweets from more regional areas of greater Melbourne express a more positive sentiment than inner Melbourne.

The grey areas are cities with insufficient data to determine a colour representing the cities' sentiment.

As for the second part, we are interested in the variation of sentiment value between different hours in a day and between different weekdays in a week. Figure 2.1.2 is an example of how sentiment value of Melbourne city fluctuates in different time periods (day or hour):

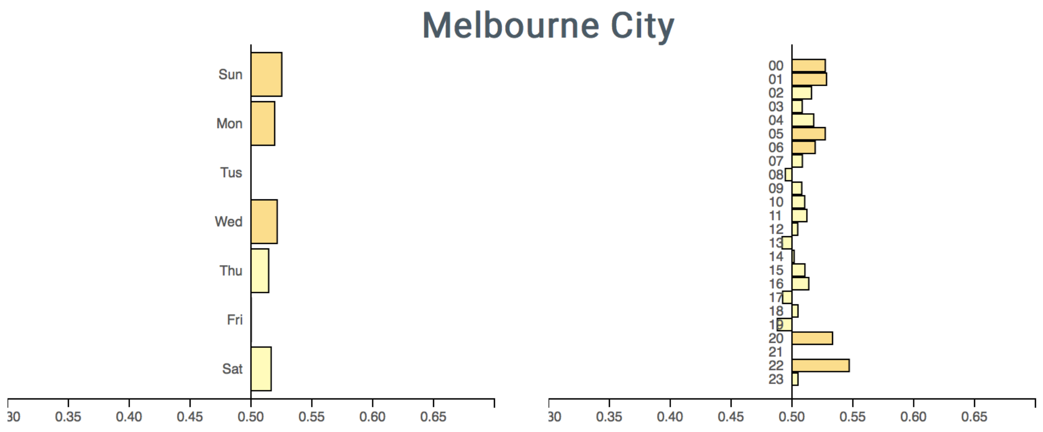

Figure 2.1.2

From the screenshot above, we can see that at most hours of the day, people tends to post slightly positive sentiment tweets. This is also the case with days in a week. From the variance of the numbers, Melbourne city seems a little "boring".

However, it is found for other cities in the greater Melbourne region, the pattern is much clearer. For example, the following screenshot shows the sentiment value at different days and different times for the city Keilor.

As it can be observed, people here likes to post more positive tweets during weekends and more negative tweets in weekdays. Also, between 3pm and 7pm seems to be the most depressing hours for people in this city, and 8pm is most likely for people here to express positive sentiments.

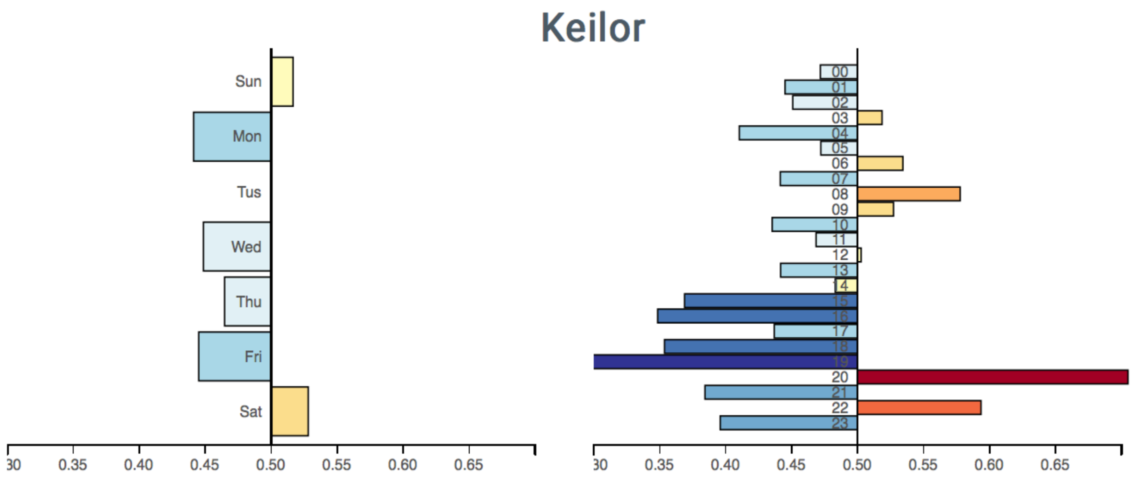

Figure 2.1.3

Now for the comparison between the sentiment values of cities and personal income level of that city. Figure 2.1.5 demonstrates the level of income of cities around Melbourne.

Referring to Figure 2.1.1, it is noticeable there is some correlation between income level and willingness to express positive sentiments such that cities with higher income are generally more likely to post with positive sentiments. However, some exceptions are noticed, for example the city of Kingston, even though it has a relatively higher income level, people tweeted from here are more likely to express negative sentiments (as shown in Figure 2.1.1)

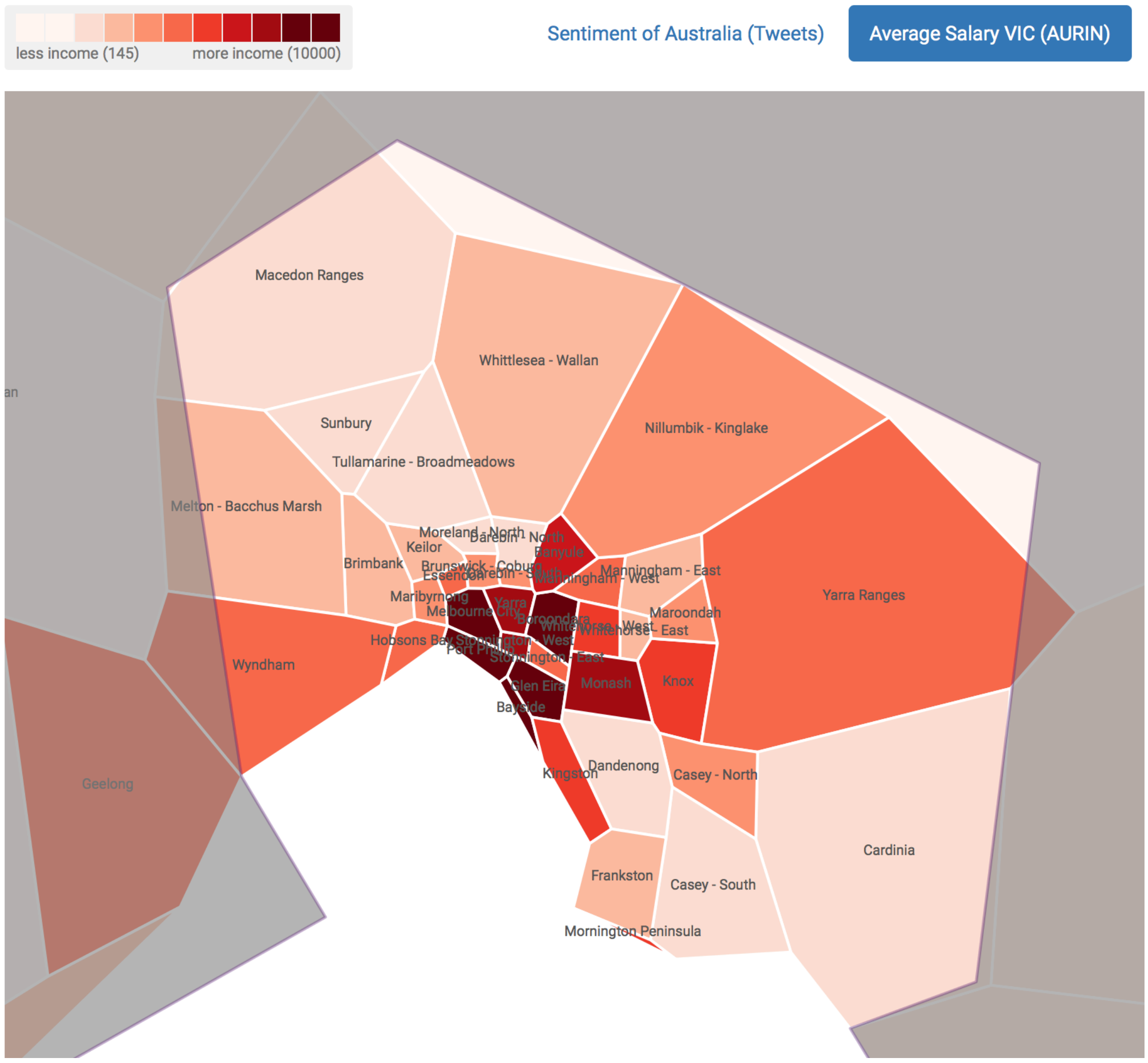

Figure 2.1.5

2.2. The most welcomed emoji – smiley

Nowadays, people are tending to express themselves by using "emoji", which are widely used in electronic messages and Web pages. Emoji were firstly created in Japan around 1998, and it soon has become much popular all over the world. Each emoji is coded with Unicode, and over two thousand emoji are supported on Twitter.

So, we try to figure out the usage of one specific emoji in different Australian areas. In this scenario, we have calculated the proportion of the emoji - "smiley" in different cities. The map following shows the usage of "smiley" around the greater Melbourne region.

From the look of screenshot, there is minimal usage of the smiley emoji around Melbourne area, or at least not very popular among most of the population.

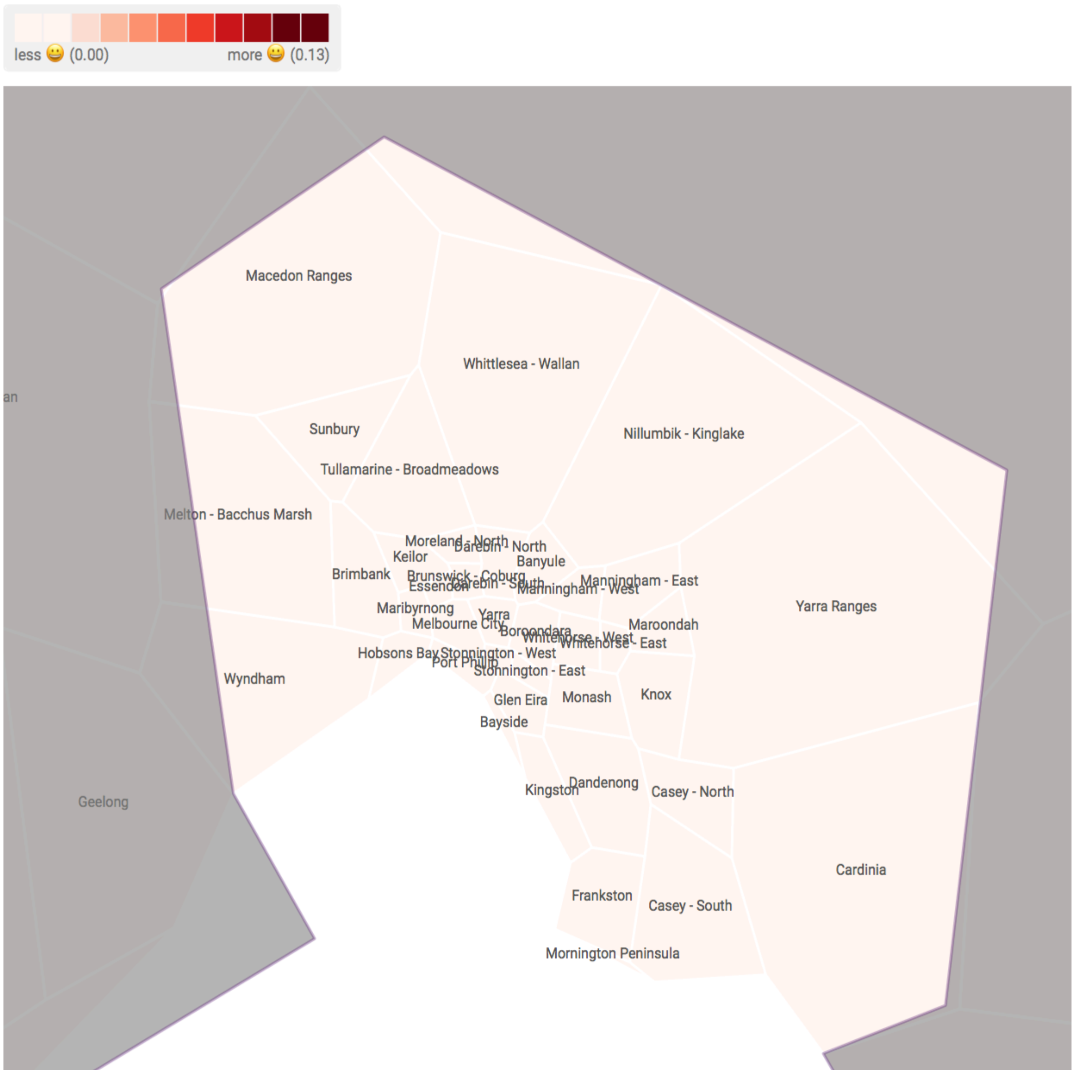

Figure 2.2.1

This is not the case for Australia. As it will be shown in Figure 2.2.2, in the city of Burnet in Queensland, the smiley emoji is extremely popular.

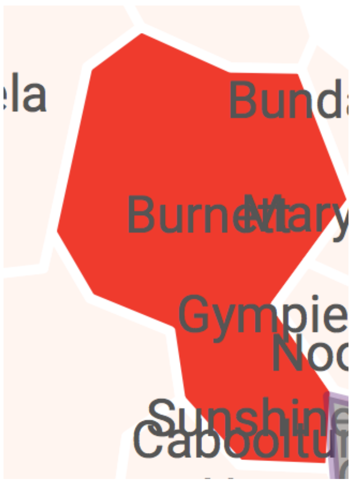

Figure 2.2.2

2.3. The language Proportion in Australia

As a nation of immigrants, the population of Australia is consisted of many people who have non-English language background. Under this circumstance, we have analysed the proportion of tweets with different language.

The analytic result we gathered will be compared with the dataset from AURIN is based on data for Language spoken at home for 2011 census to check if there is a correlation between people's speaking language and people's tweeting language. This dataset counts of all persons on Census night based on place of usual residence. This list of languages consists of the most common Language Spoken at Home responses reported in the 2006 Census.

Figure 2.3.1 shows the proportion of English tweets from each city in the greater Melbourne region, where deeper red indicates more English tweets.

The outcome distribution is pretty much as expected since, after all, this is a country whose official language is English. However, there is still some cities with less proportion of English tweets posted, for example the city of Whitehorse West.

Figure 2.3.2 shows the visualisation of the dataset retrieved from AURIN. While comparing with Figure 2.3.1, it appears that even though people around Melbourne may speak different languages, but they all tweet in English.

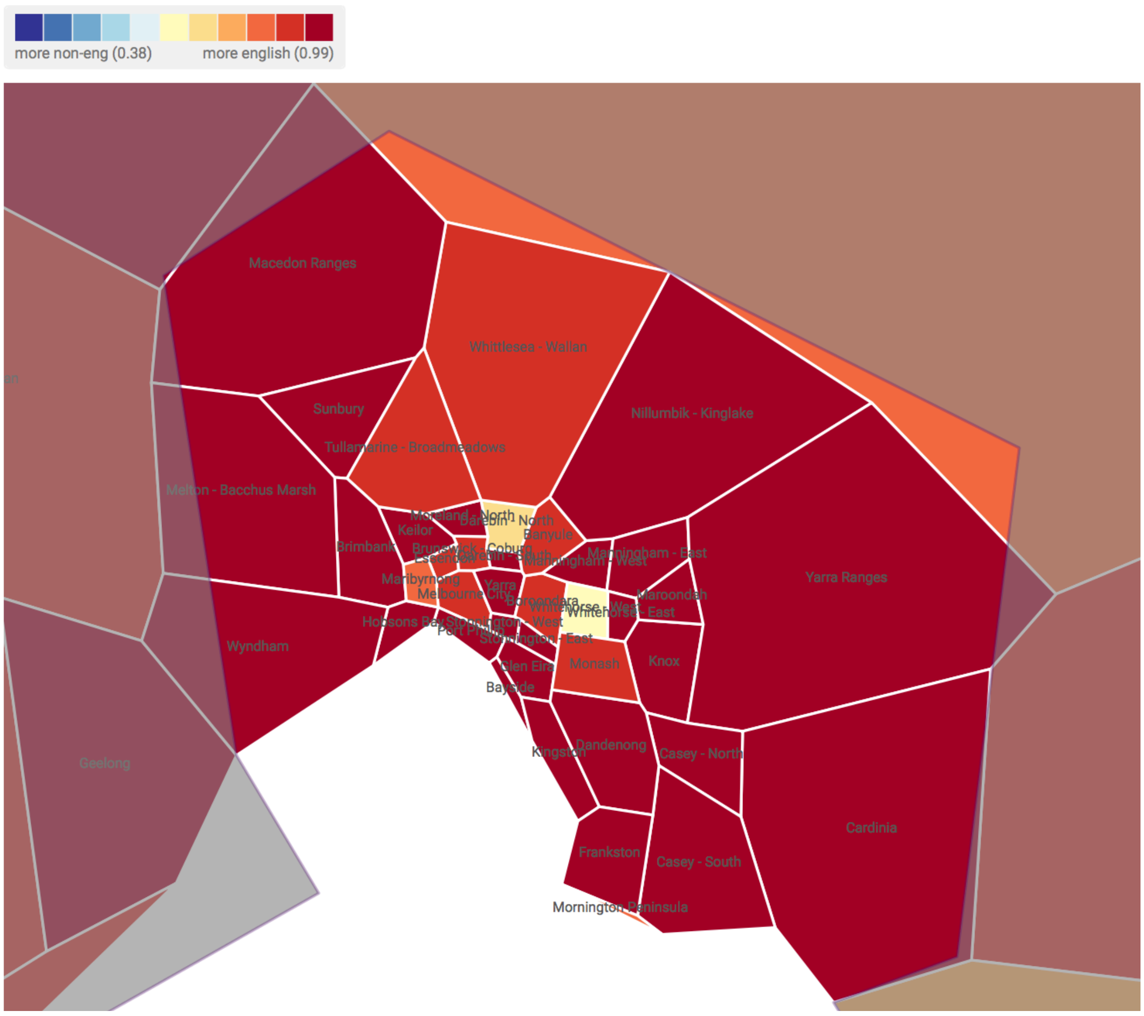

Figure 2.3.1

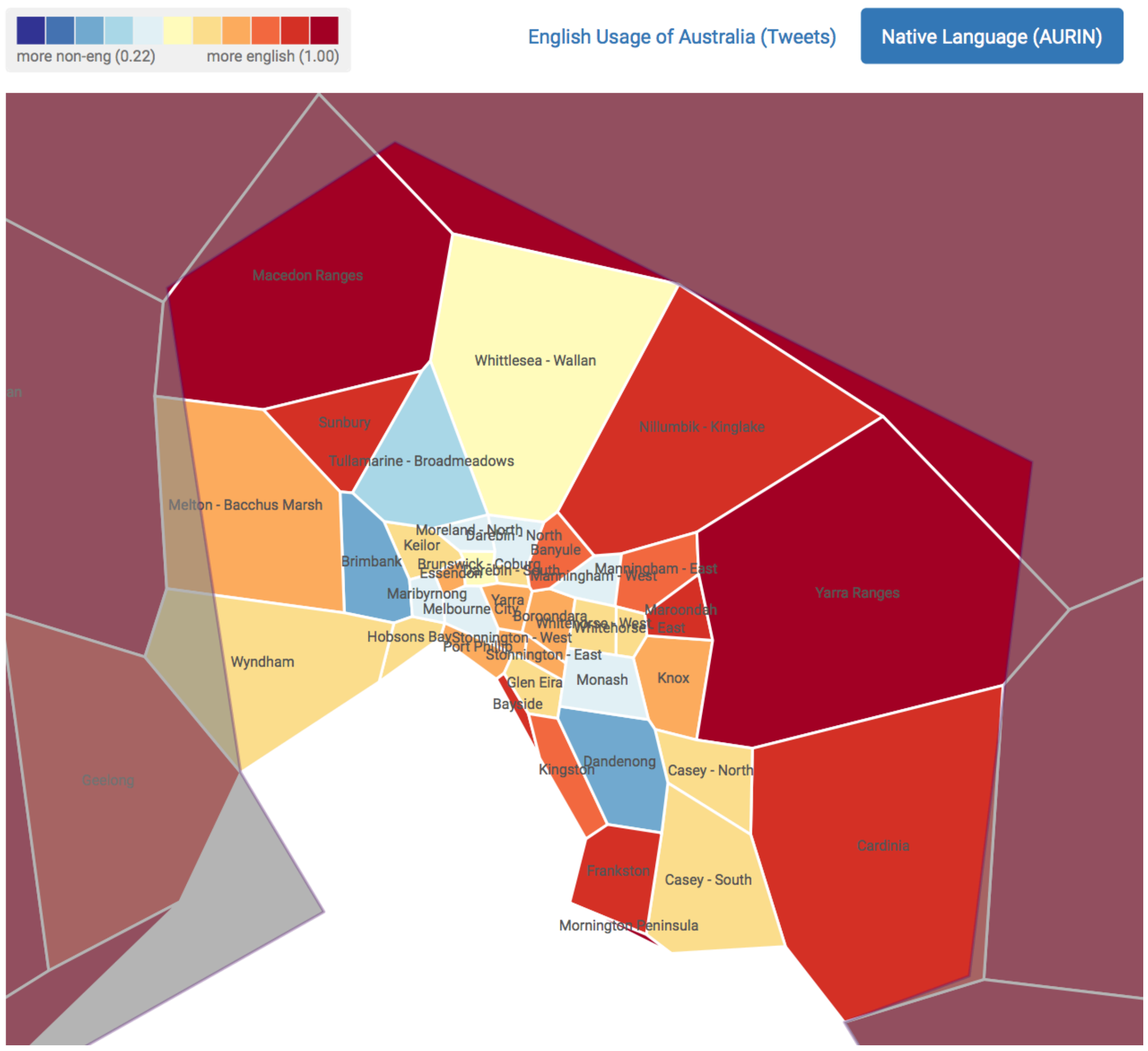

Figure 2.3.2

3. Deployment

3.1. Description

The deployment of virtual machines and the setup of each virtual machine was pretty much implemented by scripts. We divided these works into the following steps:

3.1.1. Instance Initialization

Instances on nectar is initialized using Python with Boto library. Simply specify a file as the input argument of the Python script. The file is ini format, and needs to give the information about the number of virtual machines, the name of the key that will be created in nectar and will be used to use to connect to virtual machines, the access key id and the secret access key, the username and password of CouchDB. The virtual machines starting part including identity authentication with nectar, public and secret key pair creation, instances creation will be done automatically. If succeeded, the ip address of each virtual machine would be returned and saved for the following steps, which is setting up the environment of virtual machines and running the tweets crawler.

3.1.2. Environment Setup

After starting virtual machines, the second step is to set up the environment of each node for system to work properly. Still, the script used in previous step then will run an Ansible command in shell to do that. This step was divided to four sub-steps. The first one is to install CouchDB and its dependent packages. We used Linux packages management tool apt-get to install the dependent libraries that CouchDB needs, like build-essential, pkg-config, erlang and so on. Since we are going to use the cluster mode, we downloaded the latest version of CouchDB, considering that the package management tool can only get us non-latest version of CouchDB, which doesn't contain the cluster feature. After installing the CouchDB to each virtual machine, the second part is to revise some CouchDB configuration files. For example, username, password, hostname, ip address and port that CouchDB listen on need to be set properly before the CouchDB service to be started. Thirdly, now it is time for setting up cluster mode, we used Restful API of CouchDB to realize. In specific, we sent HTTP request to enable cluster mode in shell. The Final sub-step is to install dependent Python packages that the tweets crawler needs to do its job. The tweepy library is used to grab tweets. The sklearn, numpy, scipy nltk libraries are used to provide machine learning algorithms to analyze tweets. The couchdb library is used to access database for crawlers and parsers.

3.2. Deployment Instructions

System Requirements:

- Python 3 or above (with Boto)
- Ansible

Configuration File:
A configuration file will need to be created by user before running automation. The configuration will need to look like the following:

```
[config]
key_name= <the key name will be
created by Nectar to connect
virtual machines>

admin = <username of CouchDB in
virtual machines>

password = <password of CouchDB
in virtual machines>
node_number = <how many virtual
machines are going to started>

instance_type = <type of
instances>

python_script = <the path of
python script will be run after
setting up in this machine>

aws_access_key_id = <access key
id to Nectar>

aws_secret_access_key = <secret
access key to Nectar>
```

To Run Automation:
In terminal (remotely):

```
./System-setup.py -f
/path/to/config/file
```

3.3. ReSTful Interface User Guide

The ReSTful server is configured such that only GET request will be processed.

To get a list of resources which can be queried:

```
curl -X GET
http://115.146.93.125:5000
```

To query for a specified dataset:

```
curl -X GET
http://115.146.93.125:5000/view
s/<resource_name>
```

4. System Architecture and Design

4.1. Architecture Diagram

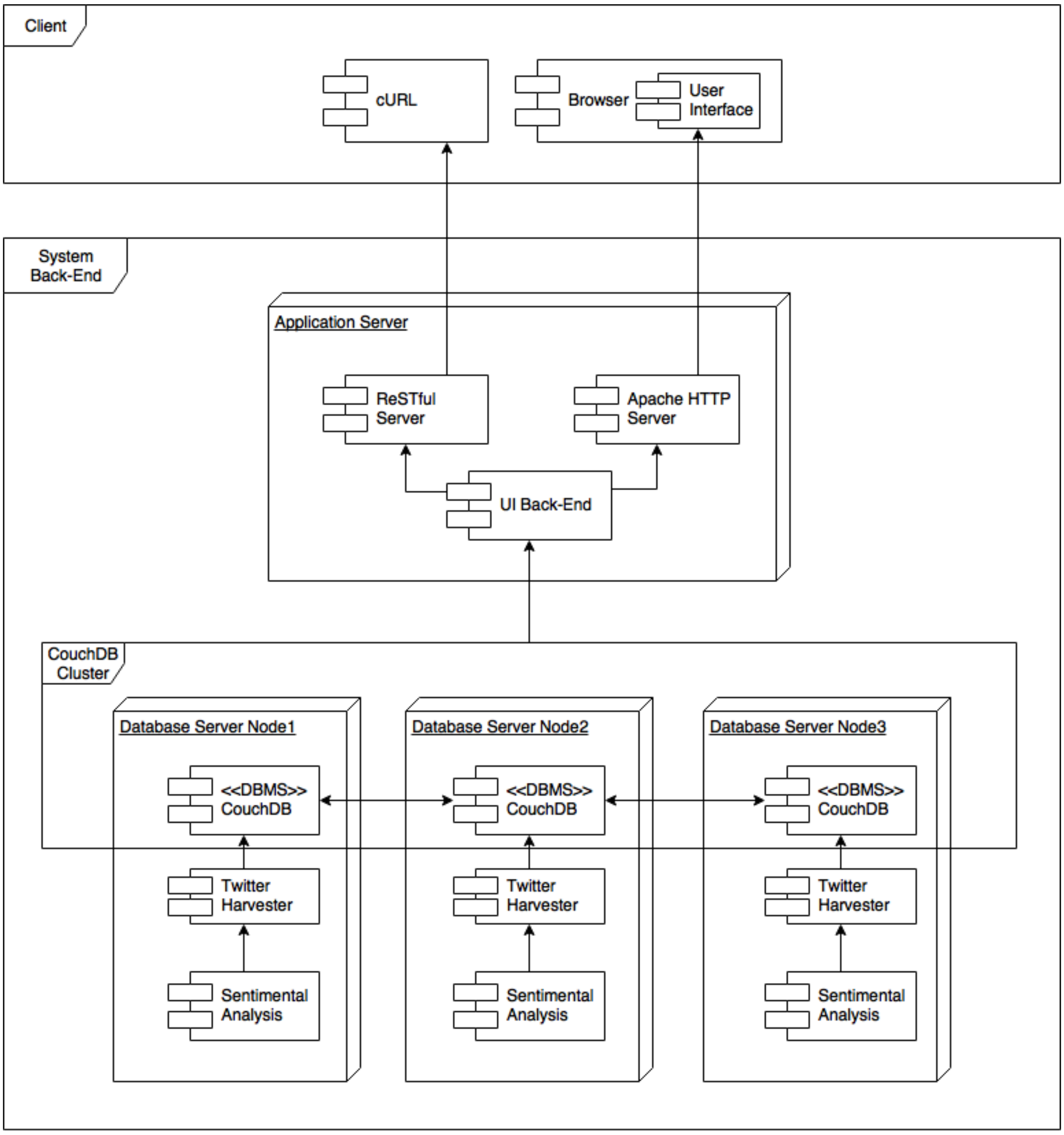

Figure 4.1.1

4.2. Motivation

The system architecture is a Service Oriented Architecture (SOA), it is designed based around the core ideas and addressed the following principles of SOA:

Standardized service contract: The application server component of the system provides predefined service of presenting various analytic scenarios as described in Section 2. Back-end modules provides services to other modules shown as directed arrows between modules.

Service low coupling: As shown in the architecture diagram, the application server queries and extract required data from the CouchDB clustering as a whole. For example, any modification of nodes in the cluster doesn't affect the performance of the system. Since each database node has its own module to harvest tweets and analysis sentiments, there are no couplings between the database nodes other than the clustering itself.

Service abstraction: Consumers can either query the described service through a web application with a user interface or query for required data through a ReSTful interface. However, any analysis that had been done in any of the database servers modules will be hidden to anyone other than the application server.

Service reusability: Deployment automation includes CouchDB deployment, the twitter harvester module and sentimental analysis module. These can be reused when/if there are more nodes added to the clustering.

Service autonomy: Each component is designed to contain only necessary logics and doesn't depend on other services for the component to execute the component's governance.

Service statelessness: None of the defined services keeps any state information.

Service composability: Each component in the architecture is designed independently of others. The logic in services provided by each system (either to consumer or other systems) remain unchanged regardless of the number of systems in the composition, in this case the CouchDB clustering.

Service granularity: The granularity of services is at a fine level as all components in the architecture provides services only of exactly one function, for example sentimental analyser only provides a sentiment score to a given text.

Service location transparency: The application server components are open to the Internet so a consumer can invoke the services regardless of its location in the network.

4.3. Other Design Aspects

4.3.1. CouchDB Views

Views are the primary tool used for querying and reporting on CouchDB documents. There are two different kinds of views: permanent and temporary views. In this project, we adopt permanent, which are stored inside special documents called design documents view to implement the Map-Reduce feature. It can be accessed via an HTTP GET request to the URI /{dbname}/{docid}/{viewname}.

For each scenario analysis, we wrote a script on Javascript to support it. All the results are stored in couchDB previously, only the documents which have changed that will be updated. After the server queries, the view will return the Map-Reduce results in Json format.

4.3.2. Twitter Harvester

For the harvesting of tweets, we used a wrapper library of Twitter API on Python named tweepy. We used the streaming API to grab the tweets as they are posted, analysis its sentiment status, find its belong city from its coordinates and extracted used emojis (for our second scenario), finally saved to a CouchDB database. The grabbed tweets were limited to Australia by location coordinates to filter the tweets from around the world which we don't need for our solution.

Each node has a twitter harvester running with the same logic. When a processed tweet (added a sentiment score) is saved to the database, its id in the database is set equal to the "id_str" element of the tweet. Because of this, any duplicate tweets harvested will not be saved due to document update conflict.

4.3.3. Sentiment Analyser

To analyse the sentiment of a tweet, an analyser is run on text of the tweet. The analyser utilised a machine learning model named multinomial Naive Bayes which is implemented in the python library sklearn (scikit-learn). The model is trained with a corpus of 30000 classified tweets from the python library NLTK (Natural Language Toolkit). The training and harvested tweets are all pre-processed before going through the model to maximise accuracy rate of classification.

This module is accessed the Twitter harvester with a single query to give a text the likelihood of the text expressing positive sentiments.

4.3.4. ReSTful Server

The Flask framework for python was used to implement our ReSTful interface. For our system, only GET request is allowed. The architecture of our ReSTful server is as shown below:

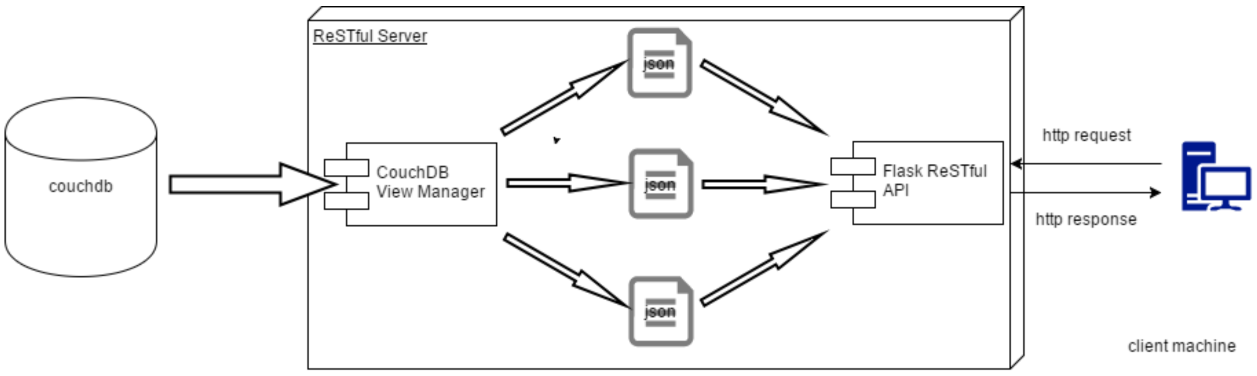

Figure 4.3.4

4.3.5. Front-End Web Application

To visualise the result, we used D3.js and JQuery to implement the visualisation. D3.js is a JavaScript library for manipulating documents based on data. D3 can efficiently manipulate documents based on data. It is extremely fast, and supports large data sets and dynamic behaviors for interaction and animation.

5. Discussion

5.1. NeCTAR

NeCTAR provides us easy access to it at any time from any location and to collaborate with each group member.

Because that the NeCTAR Research Cloud is a free for Australian education and academic research, not all openstack services are available on it. Comparing to the Amazon web services which is the mainstream cloud platform, it is not as powerful as AWS, but NecTAR has sufficient features to support our system.

However, there might be some issues with NeCTAR or Boto Library, regardless our system was designed to allow users to reuse their key pairs that already exist in the Nectar, we couldn't retrieve the key pair from Nectar according to its name. Therefore, we had to come up with only creating one key pair each time.

It is also difficult to implement automatic environment setup because different systems use different directory structure and package management tool. For example, Debian branch uses apt-get and Red Hat branch uses yum. Consequently, we decided to used Ubuntu only.

While setting up cluster, configuring the security group on NeCTAR caused more issues than expected, i.e. servers were not able to communicate with each other initially even with specified ports open on every node. However, when this was fixed, everything else run smoothly without any modification on NeCTAR.

5.2. CouchDB

The tweets which have been processed are stored in a CouchDB database as well as the dataset which we gain from AURIN. In this part, we used the CouchDB API on Python to save and update the dataset.

The advantage of CouchDB is that we can do the map-reduce function easily inside CouchDB. For each scenario, we can design a view of it in the document.

But the disadvantage is that the Map-Reduce feature are not very powerful. We can only use some simple function in javascript. Therefore, for more complexed analysis task may need more powerful distributed framework (hadoop, for example), the built-in Map-Reduce feature are not efficient and effective.

Besides, we found that when create a new view document during the development on the database with size of 1.1GB, it takes nearly half hours to do the Map-Reduce. It may blame to that most of data are stored in node 2, so that will decrease the performance dramatically. In addition, different processes are sharing the physical resources (such as our web server, twitter harvester etc.) in node on Nectar with only two cores, which also will reduce the efficiency of our system.

5.3. Twitter Harvester

The methods provide by the API were very convenient and easy to use.

However, an issue with Twitter API is that there is quota on the number of tweets can be harvest per account per day. Our solution was to execute three harvesters with three different accounts in parallel with one at each node.

Because for each node we have, there is a Twitter harvester running on it, there is a possibility of duplicate tweets being saved to database. This was avoided by setting the "_id" attribute of the tweet to be same as its "id_str" attribute, which is unique for each tweet. Then if a harvester tries to save a duplicate tweet, CouchDB will throw a document update conflict exception and preventing the document from being saved. The exception will be overlooked by harvester, so it can continue harvesting other tweets without being interrupted.

It was also quite hard to define a desirable set of restrictions on tweets so we only harvest the tweets might be useful for our system. For our current setup, the harvester is restricted with a bounding box containing the entire Australia. The problem with this is that some of tweets were harvested from places other than mainland Australia which were unneeded.

Another issue with harvested tweets was that not many tweet contains the exact coordinates. Therefore, there were not too many useful tweets harvested. To work around this issue, we decided to reuse the bigTwitter.json file from our first assignment.

5.4. Sentiment Analyser

By observation, the analyser makes reasonable sentiment score for about 60% - 70% of all English tweets harvested. Unfortunately, this is only the case for tweets written in English. For tweets written in other languages, it seems our analyser mark them all with neutral sentiment (0.5). This was expected because there were no tweets in training set of other languages. Even if there was any, our pre-processing on tweet texts will not be able to process them. As a result, this may cause our judgement of sentiment of some cities to be inaccurate. However, given the overall high proportion of English tweet, this is not too big an issue in the national scope.

Another limitation is that in English, same sets of words don't always express the same emotion, i.e. people uses sarcasm. However, there was no way for our model to recognise this behaviour because essentially, a word has only one meaning to the machine. This issue may also cause inaccuracy for calculation of sentiment value. For this limitation, we assumed only very few people would use sarcasm in their tweets, and we believe this assumption is quite realistic.

5.5. Deployment Automation

There are some parts of the system deployment procedure that might occur errors. First, if there are some parameters missing or wrong in the configuration file, the process will stop and return error message. For example, if the access key id is not given or wrong in value, the program will stop and return error messages. Second, during the instances creation part, if the given key name is already existed or if there is not enough available resource for requirement, the program will stop and return error messages. The environment setting section is much less likely to have problems since newly created virtual machines are pretty much the same with the virtual machines on which we tested this part of program. But to be secure, some actions are still taken to reduce the probability as less as possible. If there is not enough disk space to use when Ansible tries to install dependent packages for CouchDB or modules for Python, an error will be returned. Once errors happen, all steps have already carried out will be rolled back.

6. Workload Distribution

This section describes the role of each member in the team for this project.

6.1. Hangyu Xia

Hangyu's main task was to oversees construction of our ReSTful server. He also re-formatted the raw data from CouchDB views to a format which can be easily processed by front-end web application to be visualised for consumers. Consumers can also access this re-formatted data via the ReSTful interface.

6.2. Hanwei Zhu

Hanwei's main task was to construct views on CouchDB and write MapReduce functions to produce necessary data to be used for various scenarios.

6.3. Jinchao Cai

Jinchao's main task was creating Ansible script for automatic deployment of our system, including CouchDB clustering and sentiment analysis environment. During these work, some error handling problems and scalability problems were considered and taken care of by him.

6.4. Wenzhuo Mi

Wenzhuo designed and implemented the front-end web application of our system. His tasks include grabbing the data prepared by Hangyu and visualise these data scenario by scenario.

6.5. Zequn Ma

Zequn designed and implemented the Twitter harvester component, including sentiment analysis module and tweet locator module. His tasks also include populate the database with documents containing any analytical results required by various scenarios.

7. Conclusion

Overall, we believe our outcome for this project is quite successful and interesting. Even though sentiment analysis result may not be perfectly accurate, but as discussed in Section 5.4, the outcome is satisfactory.

Also since our system maps the tweets to entire Australia, if more tweets are harvested. Our system will surely show a set of more accurate tweet analytical results which can be mapped onto the country's map.