COMP90024 Cluster and Cloud Computing
Assignment 2
#The Deadly Sins Analytics on Australian Social Media

Team 18
Yizhou Wang 669026
Jiazhen Hu 971800
Tonghao Wang 1039694
Zhiyuan Shen 1033415
Yuxuan Chen 1035457

ABSTRACT

This report contains the effects, results and revision of Team 18 for Assignment 2 of COMP90024(Cluster and Cloud Computing). The system serves as a social media analyser focusing on the seven deadly sins, with Twitter and AURIN as data source and Nectar as a cloud platform, to make insights into the life essence in several metropolises in Australia.

This report contains the effects, results, and revisions of Team 18 on COMP90024 (clustering and cloud computing) assignment 2. The project requirements, team members and individual task assignment are introduced. It explains how to implement server configuration, database storage, data collection, and data analysis. In terms of technology, the system uses Twitter and AURIN as data sources, the nectar cloud as a cloud platform, and as a social media analyst focusing on the seven deadly SINS to gain insight into the nature of life in several major Australian cities. Finally, it summarizes the difficulties encountered in completing the project and the expectation of how to improve in the future.

1.Introduction

As one of the social media platforms of most users, Twitter is a great data resource for data analysis researcher. With a huge amount of text that contains sentiments or thoughts on anything, researchers can effectively make insights through the data. A single tweet may be posted with geo information, which makes correlation study on geographical information and human emotions possible. Besides Twitter, AURIN serves as another reliable data source, containing various kinds of life facts in Victoria.

Our system harvests tweets from Twitter using its authenticated API, stores to databases on cloud servers and provides a front-end website for clients to find out our insights through the data we collect from Twitter and AURIN. To implement the system, we have 2 tweet harvester on the cloud, 1 data provider doing data analysis via CouchDB built-in MapReduce, combined as a clustered CouchDB database of 3 nodes to store tweets, and 1 web server for clients to access results.

Jobs of each member in completing this system is as follows:

Team member	Jobs
Yizhou Wang	Tweet harvesting, Text analysis
Jiazhen Hu	System architecture design, CouchDB
Tonghao Wang	Auto deployment with Ansible
Zhiyuan Shen	Web design, Revision
Yuxuan Chen	Tweet harvesting, Scenario design

2.Related Works

AURIN[1] is a cooperative national project which provides built environment and urban researchers, designers and planners with the electronic infrastructure to facilitate access to a distributed network of aggregated datasets and information services. These datasets and services are essential to understand patterns of urban development and to inform and provide direction to urban growth for a sustainable future.

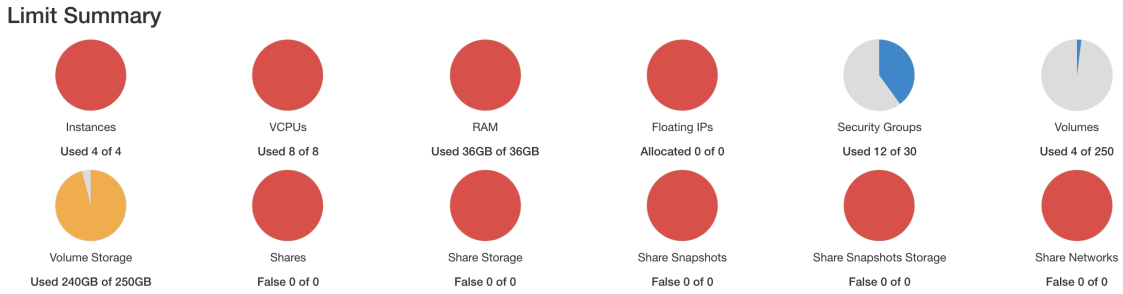
We compare the data we harvest from twitter and the related data on AURIN, to find out some underlying fact between them, and draw some conclusion or make insights into life facts in Australia.

3.System Design

This section will explain some details of this system.

3.1 Resource

Since the whole system is built on the cloud, we have been provided with a certain amount of cloud resources on the Nectar Cloud, consisting of 8 CPUs, 250GB data storage and 30 security groups. The Graph 3.2.1 shows the resources which provided for this project.



Graph 3.1.1 System resources

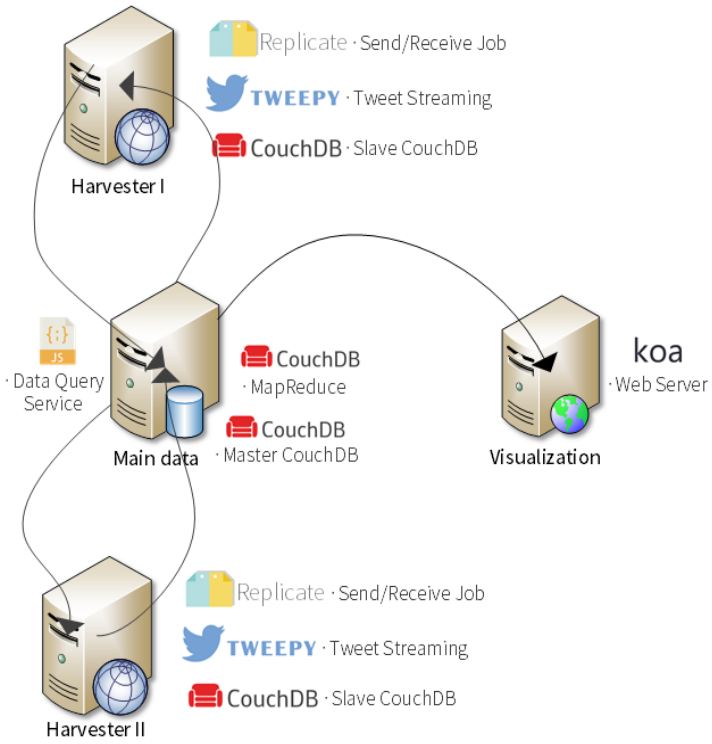
3.2 The architecture of system

In this project, finding all our nodes in the system are of almost average importance and throughput, we have allocated the resources averagely to every node, namely, 2 CPUs and 60GB data storage in each node. Also, we provided three security group for 4 instances to support their operation. Hence, the structure of this system is structured with 3 nodes deployed as a clustered *CouchDB* and 1 node as a web server for user access. Organizing resource in such paradigm is in consideration of both fault tolerance and scale out, which will be discussed in detail in Section 3.3 and 3.4. The graph 3.2.1 shows the view of created instances which include their information.

Instance Name	Image Name	IP Address	Flavour	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
Group18-Instance4	NeCTAR Ubuntu 16.04 LTS (Xenial) amd64	172.26.38.79	uom.mse.2c9g	Group18	Active	melbourne-qh2-uom	None	Running	3 days, 6 hours	Create Snapshot
Group18-Instance3	NeCTAR Ubuntu 16.04 LTS (Xenial) amd64	172.26.38.138	uom.mse.2c9g	Group18	Active	melbourne-qh2-uom	None	Running	3 days, 6 hours	Create Snapshot
Group18-Instance2	NeCTAR Ubuntu 16.04 LTS (Xenial) amd64	172.26.37.52	uom.mse.2c9g	Group18	Active	melbourne-qh2-uom	None	Running	3 days, 6 hours	Create Snapshot
Group18-Instance1	NeCTAR Ubuntu 16.04 LTS (Xenial) amd64	172.26.38.155	uom.mse.2c9g	Group18	Active	melbourne-qh2-uom	None	Running	3 days, 6 hours	Create Snapshot

Graph 3.2.1 View of created instances

And the graph 3.2.2 demonstrates the architecture of system.



Graph 3.2.2 System architecture

3.2 Roles of Nodes

Each role in the architecture above is allocated a different role, to make sure that we make the full use of our resource. The detail of roles is introduced below:

3.2.1 Master Data Node

We have set Instance1 as the master data node, among all the 4 instances on nectar. The tasks of a master data node are as follows:

- Save tweets
  The master node has the main CouchDB database installed and data from Twitter populated into all the time. The data is harvested by slave data nodes, and filtered out to make sure that all tweets contain geoinformation, thus being useful.
- Remove duplicate tweets
  We have set the id of a single doc to the tweet ID, so that all same tweets, including retweets, has a same CouchDB doc ID in the same database. Therefore, when saving a new doc into a CouchDB database, the CouchDB system will cancel the saving process if the ID already exists in the database and proceed the saving process if there are no documents with the same ID. This is the way we remove duplicate tweets and retweets. To make sure this way works, we need to save all newly harvested tweets to one database as the origin.
- Mapreduce
  With data stored inside the local CouchDB in the master data node, we can

do MapReduce on the data to provide basic analysis results to the web server.

- Make replicate rules
  After the tweets start to be populated into the main data node, it will create replicate rules, give replicate privilege to the qualified slave nodes, and then dispense the origin database content to all these nodes. The reason why it's implemented in this way will be discussed in section 3.4.2 System Scale-out analysis.

3.2.2 Slave Data Node

We have set the Instance2 and the Instance3 as the slave data node, among all the 4 instances on nectar. The tasks of a slave data node are as follows:

- Harvest tweets
  The main task of a single slave node is to harvest new tweets from Twitter, filter out useless tweets, which are of no geo information and save the remaining tweets into the main CouchDB.

3.2.3 Web Server Node

We have set the Instance4 as the web server node, among all the 4 instances on nectar. The tasks of a web server node are as follows:

- RESTful API for analysis results
  The results, as well as the original data, can be accessed by the RESTful API on the web server.
- Visualization
  Besides getting data and analysis results directly via RESTful API, clients can also take a look at our own visualization for the insights into the data we have collected, by accessing our website. See Appendix section for detail.

3.3 Fault Tolerance Analysis

- Master Data Node Failure
  If the master data node fails, we can start a poll among all slave nodes or just select a particular slave node as the new master data node. The way we save new tweets into the master node database can maintain the property that no duplicate tweets will be saved. Since all the slave nodes in our small system have a replicate of all database in the master node, we can simply switch the address from the failed node to the new master node, so that the web server can then continue to get data while the slave nodes can still save harvested tweets to it.
- Slave Data Node Failure
  Since there are 2 homogeneous slave nodes, even if one of them fails, we can still have another running and keeping the system functioning. However, this situation cannot last for too long, because when 1 slave node has to do the workload for 2 slave nodes, the probability that it also fails will increasing. In case, both of them failed, we need to switch the slave node that's over-workload and restore the failed node as soon as possible.

- Web Server Node Failure
  The web server in our system is the only single-point-of-failure. Due to the fact that there is no alternative or substitute for this node, clients can no longer access our website if it fails. However, the backend service will still be functioning, so we just need to restore the web server node so that the web service can be back to normal.

3.4 Scale Out Analysis

The description of the node expanding.

3.4.1 Node Level Scale Out

- Master Data Node
  If we only have master data node to be scale out, the workload of all the slave data nodes will increase since more databases for them to populate data in emerging. To solve this, we can group data nodes, assign each slave data nodes group a master data node, thus averaging the pressure of slave nodes.
- Slave Data Node
  If we only have slave data nodes to be scale out, the workload of the master data node will increase since more data is populated into its database. To solve this, we can select a new master data node among all slave data nodes and also divide them into groups, with 1 master among each group.
- Web Server Node
  If we only have web server node to be scale out, the workload of the master data node will increase since more date query requests are sent to it. If we select a new master node, the balance between master and slave data nodes will vanish. So, there will be trouble if the only node to be scaled out is the web server node.

Fortunately, a single node scale out rarely happens. The discussion above is just to demonstrate the way we scale out at a node-level analysis.

3.4.2 System Level Scale Out

When we need to perform a system-level scale out, which is more common than single-scale scale out, there are 2 main tasks we need to do to adjust the system.

- Quick instance creation and configuration
  Since we have implemented the automatic deployment with Ansible and OpenStack, new instances with different characters in the whole system can be quickly created and configured.
- Rebalance between nodes
  With more nodes comes more issues. Keeping the original architecture that all slave nodes populate data into all master nodes is not wise anymore. We need to group the nodes, just as discussed in 3.4.1. Then, the original architecture can be treated as a reliable setup for a subsystem in the whole, newly-scaled-out system. In this scenario, every slave data node will have different privilege to access different databases in different master data nodes.

A simple CouchDB cluster will share all the database automatically in a cluster. However, we cannot simply add all data nodes in a subsystem into a CouchDB cluster, because it's not secure to share all database content in such a peer-to-peer style. Rather, we would like to maintain the master-slave hierarchy, so that every slave node in a subsystem, and the administrator of this subsystem, can only have access to the database needed, to maintain the security of the whole system.
This is why we let the masters control the access right of their databases, rather than letting CouchDB do so automatically.

4.User guide

This section mentions the processor of deployment and introduces the tools which has been used.

4.1 Purpose

Due to the gradual expansion of project requirements and the limitations of a small number of resources, the resources of a single server or host and its stability are not sufficient to support the long-term operation of large applications. But using a cloud server can effectively solve or alleviate this situation. Through the cloud server, we can build a multi-node, scalable distributed system by deploying multiple virtual machines to support the stable operation of various programs. In order to meet the increasing demands of the program, the cloud server's fast response, security, convenience and convenient storage features can easily meet various needs. By using the Ansible tool to operate the server. users only need to run a script file to automatically create a virtual machine and deploy the required runtime environment.

4.2 Ansible

Ansible is an emerging automated operation and maintenance tool that combines the advantages of many existing operation and maintenance tools to implement batch operating system configuration, batch program deployment, and batch run commands. The user only needs to install the ansible program on the management workstation to configure the IP address of the managed host, and users can automatically manage the corresponding virtual machine. In addition, Ansible has a modular design feature, calling a specific module to accomplish a specific task, itself is a core component, short and fine. Ansible is simple to deploy, works in master-slave mode supports custom module functions, uses the playbook to operate in the order in which tasks are set, and expects each command to have a power feature.

4.3 Docker

Docker is an open source application container engine that allows developers to package their applications and dependencies into a portable container, then publish them to a virtual machine, or virtualize them. The containers are completely sandboxed and interact with each other. There won't be any interfaces between them. And docker automates repetitive tasks, such as building and configuring

development environments, freeing developers to focus on what really matters. At the same time, it also features fast start-up, flexible scaling, rapid expansion, and easy migration to help users easily deploy the operating environment.

4.4 Playbook

The playbook is a collection of ansible commands, which are written in YAML language, run the process, and the ansible-playbook commands are executed in order from top to bottom. Simply put, playbooks are the foundation of a simple configuration management system for multi-machine deployment systems, ideal for deployment of complex applications. In addition, playbooks can be used to declare configurations, in the playbooks can be arranged in an orderly execution process, even in a group of machines, orderly execution of specially specified steps, and can be initiated synchronously or asynchronously task.

4.5 Role of playbook

As individual playbook files get larger, we need to reorganize Playbooks. We can split a large playbook into several small playbook files, and then include these small files in the main configuration file by include. This is called the inclusion of playbook. We can also put a certain type of task to be executed in a directory according to certain rules, and divide the playbook into several files according to the types of tasks, files, templates, vars, etc. in this directory, and store the corresponding files. In the corresponding directory, this organization is called the roles of the playbook.

4.6 Advantages and disadvantages of UniMelb Research Cloud

When we were using the UniMelb Research Cloud to construct the operating environment of the system. We found that there are many advantages of using the cloud server, such as:

1. The cloud server can flexibility to expand or shrink resources at any time according to application requirements.
2. We can quickly access the required resources, in a few minutes to get an integrated cloud server.
3. By using the research cloud, we only need to maintain the internal problems of the cloud server, which can avoid the operation and maintenance energy and cost when using the traditional server.
4. Due to the server is deployed in many hosts, it is not easy to crash completely. Tolerance is stronger and can guarantee a long time online.

Besides, there are some disadvantages of using the research cloud.

1. Cloud services are too dependent on the Internet and have potential information security risks.
2. We have less control over the cloud server, only through the interface to access the server, and cannot fully control our data.

4.7 Processer for image creation and deployment

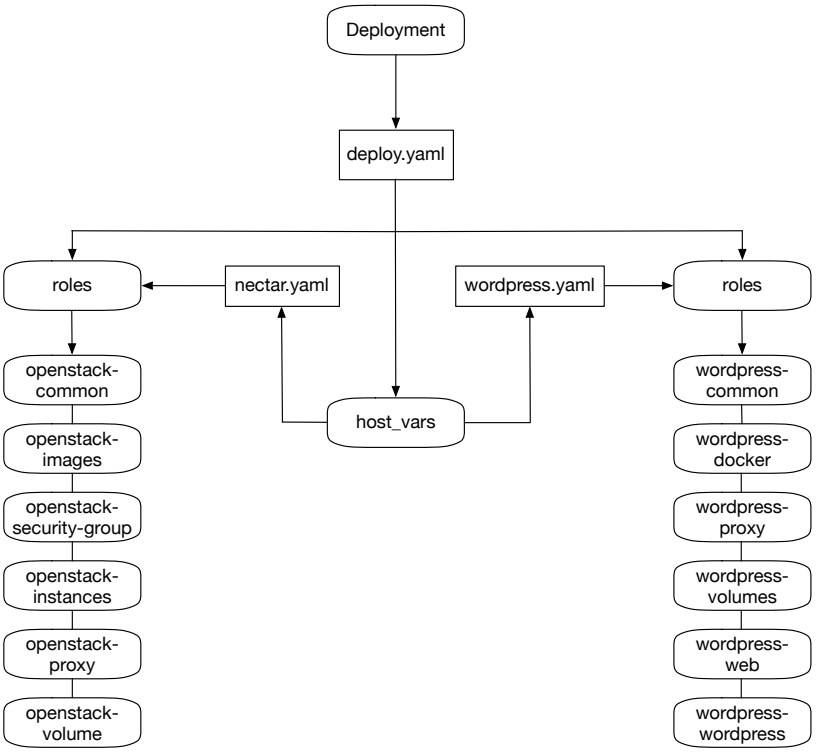
By deploying the servers, we need to run a shell script to create instances on the nectar cloud.

First, we will create 4 instances on the cloud. We set up a detailed directory to indicate the hierarchy of the different files, and we call the *"deploy.yaml"* file in the folder root directory to execute the *"main.yaml"* file in different roles by executing the run.sh script file. At the same time, *"nectar.yaml"* and *"wordpress.yaml"* files were created in the *"host_vars"* folder to make global declarations for the variables invoked under different roles so that the server could be created successfully.

```
sudo ./run.sh
```

Second, we will configure different software and environments for different hosts according to project requirements. Since we have set up the system structure, we will configure the *Python*, *Git*, *Docker* and other software for 4 hosts at the same time. Then deploy the *CouchDB* to one of the hosts and set it to master node. Second, deploy *CouchDB* and set it to slave nodes for the other two hosts. Then the three hosts are connected by cluster mode. Finally, configure the web server on the last host to allow easy access for users.

The Graph 4.2.1 shows the inventory of the deployment files.



Graph 4.2.1 Ansible Procedure

4.8 Application Invocation

After the 4 instances have been successfully created and configured, users can invoke our applications on each node with different character by running the

corresponding startup scripts. Before running the startup scripts, users should ensure that the GitHub repository of the latest version is completely cloned onto every instance.

To start the slave nodes, change working directory to "*xanthic/harvester*" and run.

```
nohup python3 Twitter.py
```

To start the master node, change working directory to "*xanthic/harvester*" and run.

```
nohup python3 replication.py
```

To start the web server, change working directory to "*xanthic/service*" and run.

```
npm install

forever start app.js
```

Due to the difference of working environment, unexpected errors may take place due to many reasons ranging from network instability to local and remote machine variance.

4.9 Challenges

Difficulties encountered in completing the project and solutions.

4.9.1 Security Groups

Due to we are assigned to use the Ansible and OpenStack to deploy the virtual machines, we need to use the SSH protocol and HTTP protocol to connect with them. Thus, when we created 4 instances, we also created the SSH security group and HTTP security group and insert them into the instances. Also, when we tried to access the Couch database, we found that the connection is refused. Then, we created a new security group to open the port 5984 to permit users to visit the Couch Database.

4.9.2 Proxy

During our using Ansible to deploy the virtual machine, we found that when using the "apt-get" command to install a package, there is an error shows "connection unreachable", so need to set the proxy in the file which its PATH is "*/etc/environment*", so that the virtual machine is connected to the software resource. Also, another proxy configuration should be set for Docker. We used two methods to set up the proxy, one of them is using the "content" element and another one is using the loop to set up it, but we should notice them in the "*host_vars*" files. The graph 4.4.2.1 shows the proxy setting.

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/u
sr/local/games"
http_proxy="http://wwwproxy.unimelb.edu.au:8000"
https_proxy="http://wwwproxy.unimelb.edu.au:8000"
ftp_proxy="http://wwwproxy.unimelb.edu.au:8000"
no_proxy=localhost,127.0.0.1,127.0.1.1,ubuntu
```

Graph 4.9.2.1 Proxy Setting

5.Twitter Harvester

The description of Twitter harvesting.

5.1 Data Collection

The twitter harvest program is built based on the "*tweepy*" module. All the tweet information is extracted by the streaming API offered by "*tweepy*". The reason to choose this API is due to the factor that it can get recent data, timely more relevant for our analysis. Furthermore, the other API such as search query is not appropriate for this scenario because there is no specific query term available in this case. To filter the tweet texts, filter method is applied to get tweets in the specified geolocation and select the tweet in English. We use the *stream.filter()* function for the selection of data posted in English in Australia. The bounding box is [113.338953078, -43.6345972634, 153.569469029, -10.6681857235]. We use this bounding box to restrict the Geographical range that we obtain the tweets.

5.2 Text preprocess

In order to fit the purpose of the sentimental analysis, the raw tweet text has been carefully preprocessed. The general regular expression and the lowering procedure are considered to remove any useless words.

Original text: @kagij Jacob this should have been redeveloped a long time ago. Don't forget the Heritage listed South Fremantle p… https://t.co/yTS8LhK5hE

After preprocessed: agij Jacob this should have been redeveloped a long time ago don't forget the heritage listed south fremantle p

The characters and punctuation symbols such as '@', '.' are effective removed also with the website link. Most importantly, this transformation helps the analyser easily give the relevant similarity score.

5.3 Volume of Tweets

The number of tweets is approximately one hundred thousand. Although it is not as much as expected, the volume is still suitable for us to do the analysis. The volume is still growing slowly as the "*tweepy*" API crawls the most recent tweets. In addition, the crawling from the provided database on the Nectar has been tried but it only has the archived tweets. Therefore, we decide to go for the stream method designed by ourselves, providing much timely relevant data but at a slow speed limited by "*tweepy*". The graph 6.1 shows the procedure of text analysis.

5.4 Duplicate data

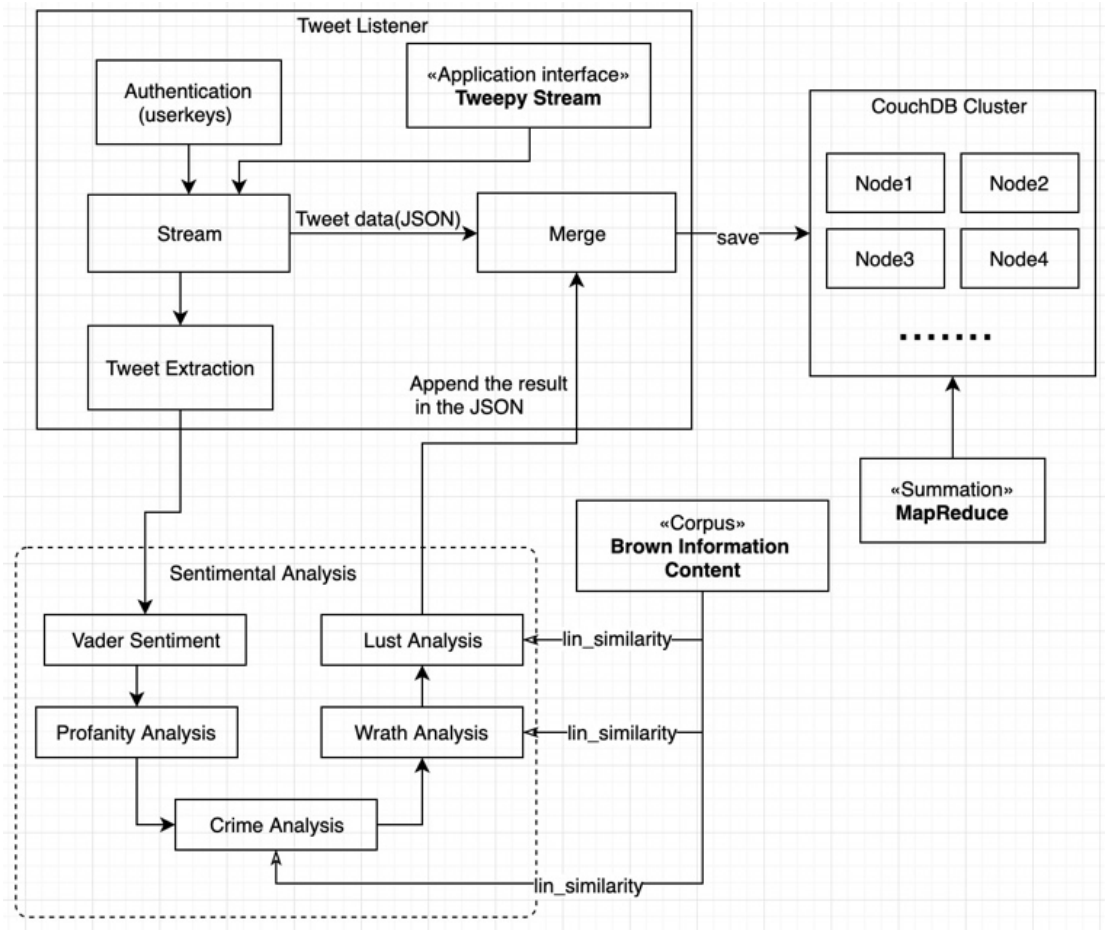
Duplicated data is needed to be prevented as it will cause the data to be inaccurate and biased. Instead of implementing any algorithms to confirm the redundancy of the data, we add the unique identifier to be able to distinguish the repetitive gathering. The specification of the mechanism is by adding two new keys into the JSON document, involving "_id" and "_rev". And simply taking the default unique identifier from "id" and implant into the "_id" we have manually created. CouchDB will immediately notice if there is any data goes into the database with the same

"_id".

5.5 Inaccurate sentimental analysis

As our approach to classify the tweets is generally fundamental, and the context of the language or expression can be far more complex. For instance, sarcasm is the most common expression in daily life, it needs a totally different language model to be analysed and be surmised by combing the context sometimes. Therefore, the prediction result is limited if there is any sarcasm occurring.

6.Data Analysis



Graph 6.1 Text Analysis Procedure

6.1 Sentiment Analysis

There are several scenarios that could show a clearer picture of analysis.

- The tweet content covers the information regarding to the crime
- The tweet content covers the information regarding to the wrath
- The tweet content covers the information regarding to the lust
- The tweet content covers the information regarding to the dirty words
- Overall sentimental score related to the whole tweet sentence

WordNet library is a large English vocabulary database developed by the Clinton University of Cognitive Science Laboratory in the United States in 1985. It is a linguistic ontology library and a semantic dictionary. It has been widely used in natural language processing research. WordNet differs from traditional dictionaries in that it organizes words according to semantics, and WordNet uses synonym sets to represent concepts, and these concepts are connected to each other through specific relationships. Each synonym set represents a concept, represented by a unique index number, with corresponding explanations and example sentences.

WordNet library is used to help us to find the tweets that we want. It classifies English words according to nouns, verbs, adjectives and adverbs. Under each main category, words with similar meanings will form a synonym network, and different synonym are also interrelated. We use this dictionary crawls tweets, and then performs similarity analysis between all the captured words and the keywords we set. According to the results it returns, we select tweets and store them in CouchDB. In this way, we can get tweets that separately contain crime, wrath and lust.

Profanity is a library for checking for defamatory or offensive language in strings. It is a machine model that builds a simple model that can detect dirty words directly without having to enter a filtered word list in advance. This is a strong and efficient library. We use this library to grab tweets with dirty words.

The Vader library is a lexical and rule-based text analysis tool that is widely used in social media text analysis. It uses manual annotation to make emotional and intensity judgments for commonly used emotional words (including adjectives, nouns, adverbs, etc.), and Vader differs from other dictionaries in that it considers common facial expressions and abbreviations, as well as proverbs, are used to deal with the emotional discrimination of social media non-standard sentences such as Twitter. It is fast enough and performs well with online streaming data.

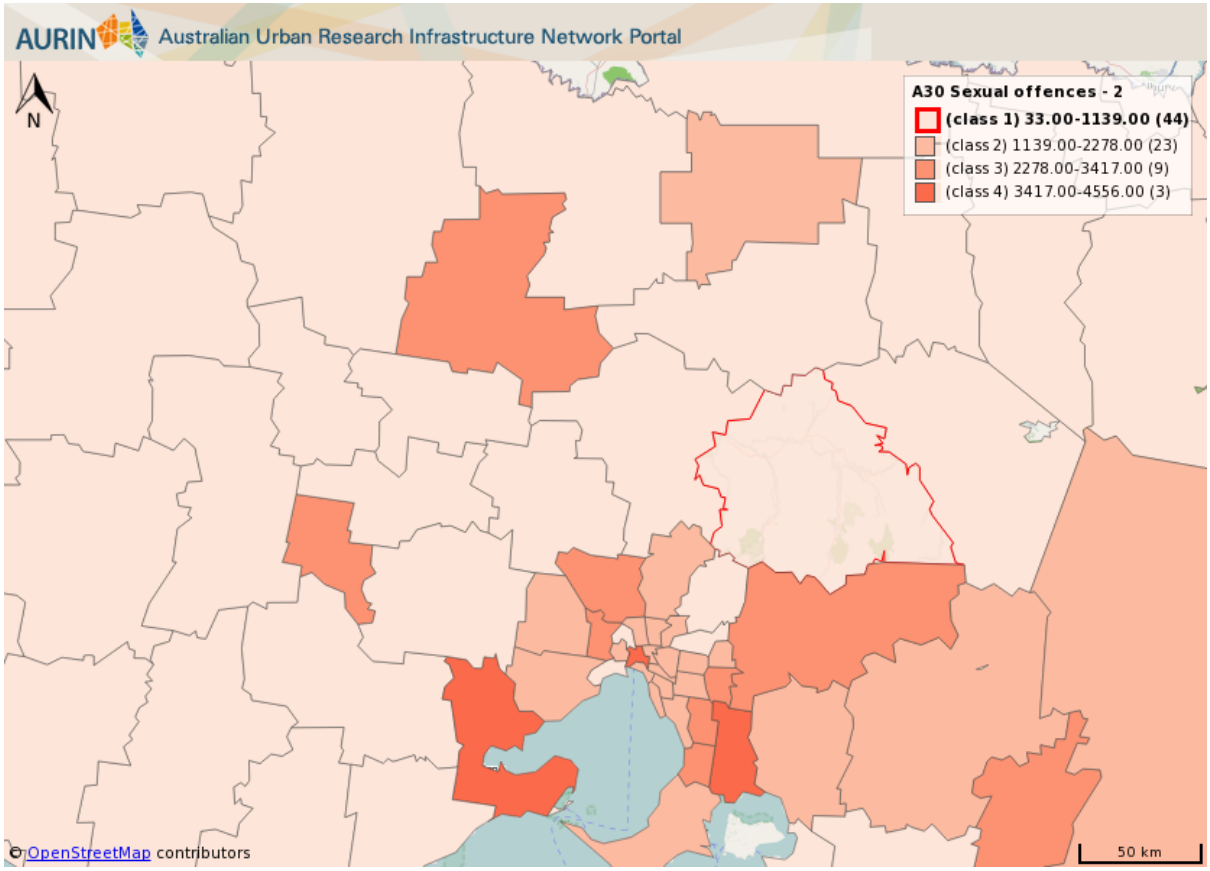
6.2 MapReduce

The data analysis tool used in our scenario is called MapReduce technique. This specifically implements the view in CouchDB and the view is given as the definition of the extraction of a certain type of data according to the MapReduce function. The procedure is divided into two steps. The first is to map the resource across the machines, the typical operations are filtering and sorting. Then the second one is to reduce the work hierarchically in that way we could get the results. The analysis is implemented on the basis of the city classification, which means the tweets are categorized referring to its location. Then for each city, five scenario analysis are all executed in order to give interesting results.

6.3 Aurin

We find data from Australian urban research infrastructure network portal (Aurin), which is a platform that builds by the Australia government for Super Science initiative. Researchers can obtain data from multiple sources. They will research and analyse the data that they collect and modelling based on data. This website

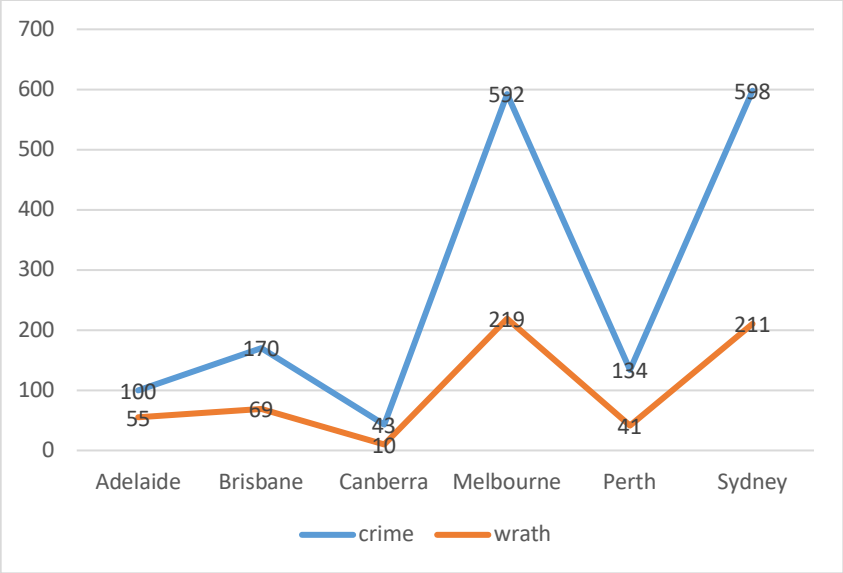
can also use the data to achieve visualization, which is very convenient for users to get the results of data analysation. We obtained the crime data from Aurin, which contains the crime category, the occurrence year and geographic location. Based on the data set that Aurin provides us, we select the data set which is sexual offences that occurred during 2010-2017 in Victorian and the data set about tweets related to crime in South Australia as control groups to compare with the results of our analysis. The graph 6.3.1 shows the view of Aurin.



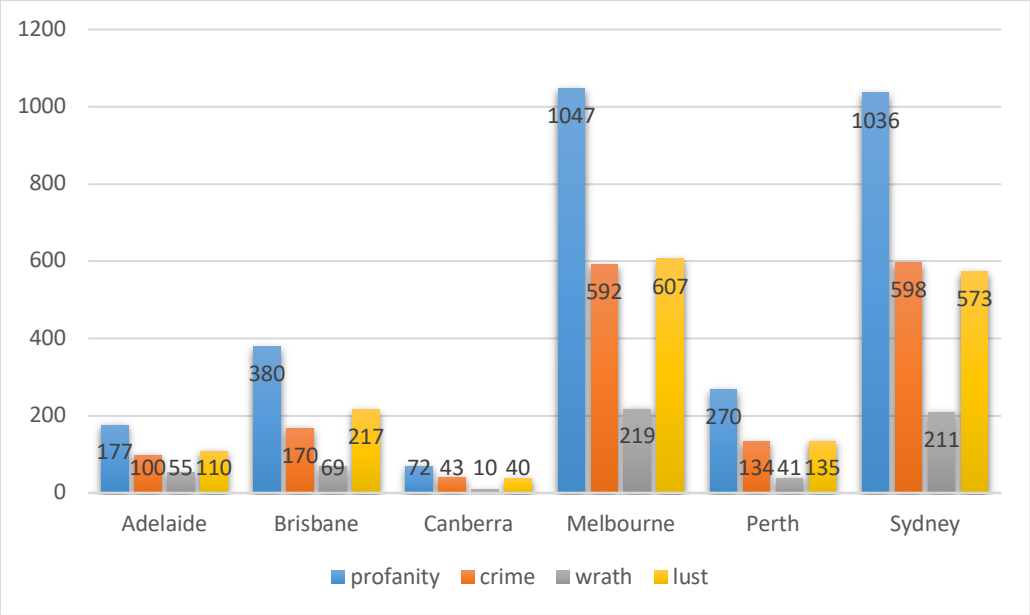
Graph 6.3.1 Aurin Visualization

6.4 Results Analysis

The graph 6.4.1 and graph 6.4.2 shows the relation between Crime and Wrath and statistics on several on deadly sins.

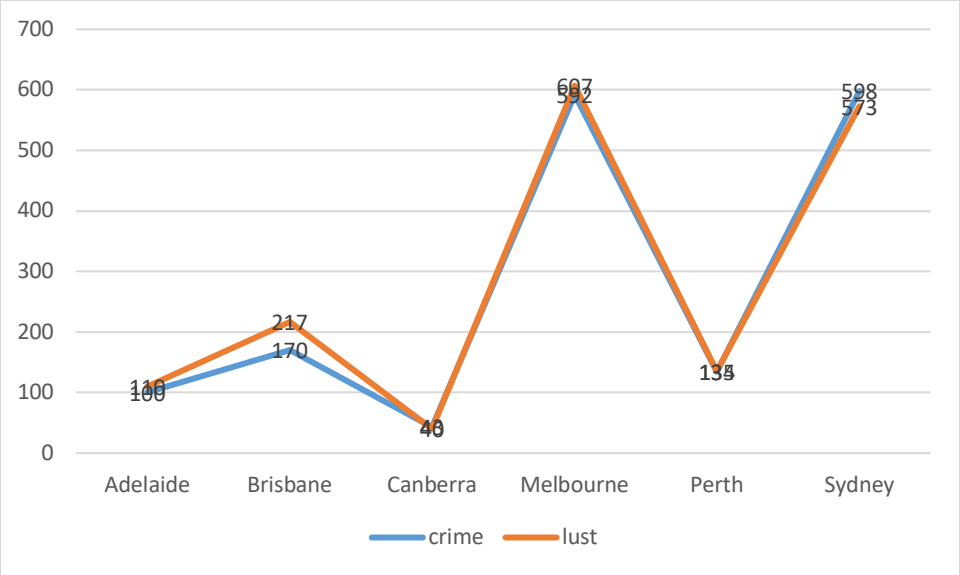


Graph 6.4.1 Relation between Crime and Wrath



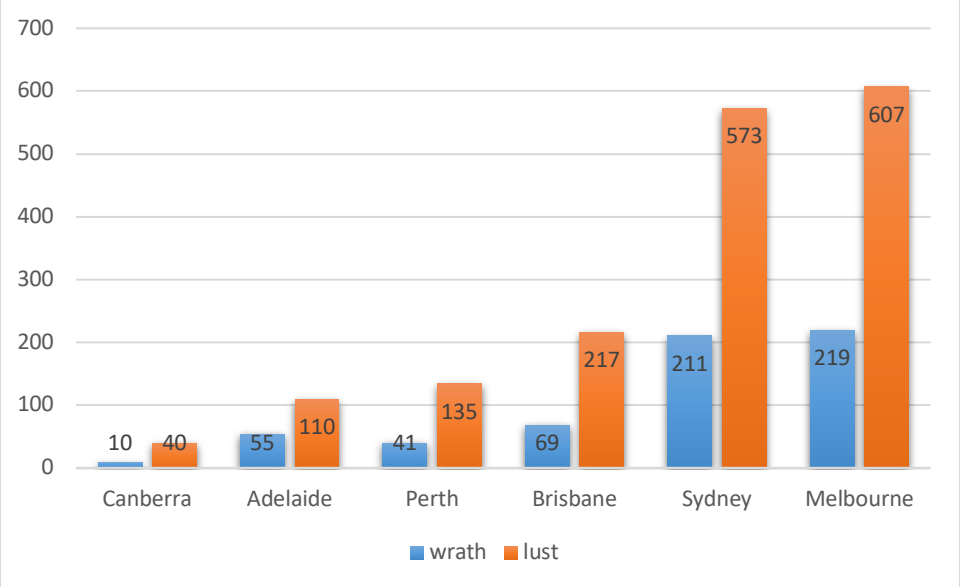
Graph 6.4.2 Statistics on Several Deadly Sins

As we use excel to analyse the correlation between the tweets we define as crime and the tweets that we define as wrath. There is a significant correlation between the crime and wrath since the correlation between them is 0.9938>0.8, which is considered to have a significant correlation between these two parameters in statistics. The graph 6.4.3 shows the relation between Crime and Lust.



Graph 6.4.3 Relation between Crime and Lust

Similarly, we compare the relationship between crime and lust by using the data that we collected from twitter. After analysing the linear graph and the correlation number (0.9958>0.8). There is a strong correlation between crime and lust. The graph 6.4.4 shows the statistics on Wrath and Lust.



Graph 6.4.4 Statistics on Wrath and Lust

We also analyse the relationship between the number of tweets and cities. And we find that there are significant differences in the number of tweets that sent between different cities. It is obvious that the larger cities send more tweets, such as Sydney and Melbourne, and we can see that the number of tweets that related to lust which sent by Sydney and Melbourne is significantly higher than in other cities. We speculate is related to the city's development. The economic situation of developed cities is generally better than other cities, and people are more likely to express

their own lust.

We use the data set from Aurin and compare the relationship between the number of tweets which is related to crime and the number of sexual offences. It is obvious that the people who live in places where have higher sexual offences will send more crime related tweets. We have found the positive correlation among the crime, lust and wrath, can help us derive the positive correlation between sexual offences, lust and wrath. It seems that who live in the cities which have more sexual offences are likely to send the tweets about lust and wrath. In addition, according to comprehensive analysis from the data set, there is a positive correlation between the city size, the number of tweets and the number of sexual offences.

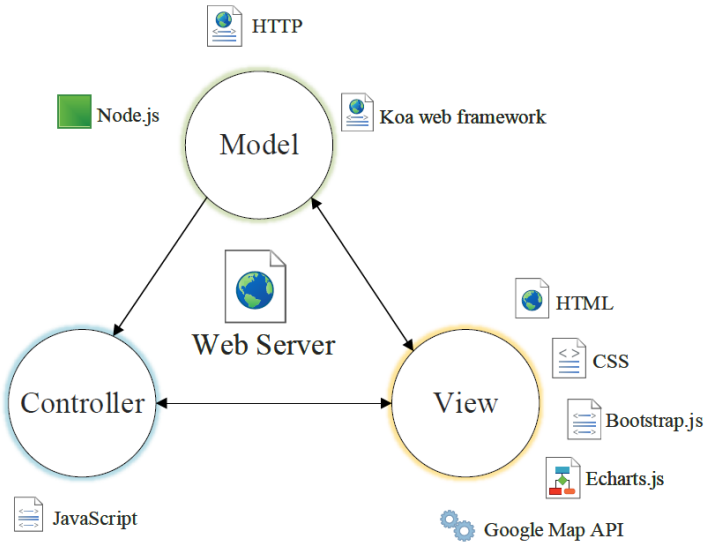
7.Web server

This part describes the operation of the web server.

7.1 Web server architecture

By deploying a web server, websites visualize the data and present our work and results of the analysis.

Web server employs the MVC model to build a layered system, the figure is a structure chart of this module. In model-layer, the server is developed by Koa[2], compared with Django, which is a smaller and more expressive web framework. Moreover, it can support developers to write servers fast. In view-layer, Using Html + CSS + JavaScript to build web pages. In addition, Bootstrap provides a method to create a clear layout quickly. Using Google Map API[3], which visualizes the result of analytics data and geographic data. Echarts.js[4] is used to draw intuitive and interactive visual charts on web pages. Web server is an excellent method to help us display the task and result of our research. The graph 7.1.1 shows the structure of web server.



Graph 7.1.1 Overall Web Server

7.2 Function Design

According to the structure of the system, the data is stored in a database (CouchDB). When users request to browse the content of websites, the browser is ordered to send a request asking the database to get data and images. Due to the MVC model and RESTful API, a Koa (a web framework) script was written by developers, which define the methods of GET, Request and router, and stored in the cloud server. When the system receives a request, a resource can simply be obtained and displayed.

7.3 Data Visualization

Undoubtedly, it is clearer and more logical to use charts and maps to demonstrate results of analysis and trend. Hence, the system mainly employs Echart.js and Google map to support to visualize data.

Echarts.js is a popular and powerful library to generate dynamic charts, based on JavaScript. By inserting this library into webs, it can draw bar charts following the given data.

Google map API has a property, which could allow the developer to combine coordinates with data to display marks and information on a true e-map. Moreover, it provides the user a method to use GeoJSON to save geographic information and any other necessary data. In our system, we display the strength of #Lust and #Wrath in six cities in Australia.

8.Conclusion and Revision

The conclusion of this project and the report.

8.1 System summary

We have implemented a cloud-based system, to harvest tweet data posted in Australia and do immediate sentiment and text analysis. The CouchDB database storing these data follows 2 principles in CAP theorem, consistency and availability. Therefore, our presented system is of high fault tolerance and good ability to scale out, but involves considerable extent of redundancy as a tradeoff to achieve this.

Taking into account that fault tolerance and system scaling are extremely important and common in cluster and cloud computing, we have implemented an automatic instance deployment application based on OpenStack and Ansible, so that even large number of instances can be restored or created and configured efficiently.

Hoping that our system could provide valuable insights into the life essence in Australia, we have collected over 100,000 tweets through Twitter API and removed all the duplicates inside. With several NLP and text analysis tools, we have illustrated our insights under various categories and sketched proper graphs to help readers understand the results of analysis easily. Further demonstration is carried out by constructing a website to offer web clients data visualization and expose API to access data in our databases.

8.2 Challenges and Solutions

Although we have successfully implemented the whole system, challenges are inevitable.

Duplicate tweet removal is important to the whole system, since same tweets and retweets can only use up a large proportion of volumes but of no use. To tackle with this, we make the harvesters save tweets, of which the doc id is set to the corresponding tweet id, into an original CouchDB database that can automatically handle id conflict, and then dispense it to authorized nodes.

While using Ansible to implement automatic instance deployment, we need to keep all 4 IP addresses in run time and group them as per their roles. Doing unit test in such a run time setting is none trivial, because every time error occurs, the IP address we obtained and kept in memory no longer exists, forcing us to delete the instances and start deployment again. To tackle this, we save a "*hostslist*" file temporarily and pass it as an argument to notify Ansible mapping of hosts and groups so that Ansible can "memorize" and continue the progress of deployment.

Challenge occurs not just during the development of our system, but the evaluation on the development cycle of each job involved as well. Our team has seriously underestimated the difficulty of automatic deployment, resulting in consequential delay of other jobs. Fortunately, thanks to great effort of all our team members, the whole project is completed as wished.

8.3 Future Improvement

However, there are still a few drawbacks in the whole system.

The web server node as a single-point-of-failure, is so vulnerable a component that no nodes can replace it when node failure takes place. More nodes are required to solve this problem.

The Twitter API we used in harvesting, has rate limitation to a single API user send requests too frequently. Thus, if the system is going to be transplanted into an enterprise scale, an API user switcher is necessary, to assure a stable tweet harvesting frequency.

The fact that AURIN cannot provide comprehensive datasets nationwide, is a limitation for us to compare emotions contained in tweets with life facts of Australian Twitter users. We select six capital cities in Australia, but AURIN can only provide data from Victoria and South Australia in a few years. Therefore, we'd suggest an expansion on data collection range for AURIN.

APPENDIX

Demonstration Video

Readers could watch a demonstration video on YouTube.

The link of project demonstration is: https://youtu.be/_pCDZ7O-2gY

Project Code

Readers could refer to the project code on GitHub.

The link of our project is: https://github.com/dogtwofly/xanthic.git

Web Access

Before visiting the web pages, readers are expected to connect to the environment of University of Melbourne, maybe VPN (Cisco AnyConnect Secure Mobility Client) will be needed. The demonstration web is: http://172.26.38.79:8080/index.html

BIBLIOGRAPHY

[1] Australian Urban Research Infrastructure Network (AURIN). Retrieved from:

https://docs.education.gov.au/system/files/doc/other/20150430_information_sheet__australian_urban_research_infrastructure_network_aurin.pdf

[2] Koa documentation and examples (2019). Retrieved from:

https://github.com/koajs/koa

[3] Echarts.js documentation (2019). Retrieved from:

https://echarts.baidu.com/option.html

[4] Google map API documentation (2019). Retrieved from:

https://developers.google.cn/maps/documentation/javascript/tutorial