|  |
| --- |
| COMP90024 Cluster and Cloud Computing |
| **Australian Social Media Analytics** |

|  |
| --- |
| Team members  Ziyue Guan 876871 (ziyueg)  Shiqi Wang 878547 (shiqiw4)  Xing Wei 777677 (xwei4)  Mingyuan Cui 870530 (mingyuanc1)  Adam Elaoud (Subject Withdrawed) |

Content

[Roles and Responsibility 0](#_Toc513649795)

[1. Introduction 1](#_Toc513649796)

[2. Data Sources Introduction 1](#_Toc513649797)

[3. User Guide 1](#_Toc513649798)

[3.1 Deployment 1](#_Toc513649799)

[3.2 Web Page 1](#_Toc513649800)

[4. System Design 3](#_Toc513649801)

[4.1 Automated Deployment 3](#_Toc513649802)

[4.2 Data Collection 3](#_Toc513649803)

[4.3 Scenario Analysis 4](#_Toc513649804)

[4.4 Data Storage 5](#_Toc513649805)

[4.5 Data Statistics 5](#_Toc513649806)

[4.6 Result Visualization 5](#_Toc513649807)

[5. Detailed design and Implementation 7](#_Toc513649808)

[5.1 Scalable automated deployment of cluster VM environment 7](#_Toc513649809)

[5.5.1 Background 7](#_Toc513649810)

[5.1.2 Requirement 7](#_Toc513649811)

[5.1.3 Automated cloud provisioning 7](#_Toc513649812)

[5.1.4 Types of automated tools 7](#_Toc513649813)

[5.1.5 Introduction of Boto and Ansible 8](#_Toc513649814)

[5.1.6 Cloud provisioning 8](#_Toc513649815)

[5.1.7 Ansible functions 9](#_Toc513649816)

[5.1.8 Ansible implementation in this assignment 10](#_Toc513649817)

[5.2 Tweet harvesting and content analysis 11](#_Toc513649818)

[5.2.1 Overall Workflow 11](#_Toc513649819)

[5.2.2 Program Design 12](#_Toc513649820)

[5.2.3 Interaction model 15](#_Toc513649821)

[5.2.4 Other design issues 15](#_Toc513649822)

[5.2.5 Analyses and improvements 16](#_Toc513649823)

[5.3 Detailed Data Statistics: 16](#_Toc513649824)

[5.3.1 CouchDB-Python 16](#_Toc513649825)

[5.3.2 Key Points in Data Statistics 16](#_Toc513649826)

[5.3.3 Implementation Details 17](#_Toc513649827)

[5.3.4 Eliminate the Choice of Spark 18](#_Toc513649828)

[5.4 Web Application 18](#_Toc513649829)

[5.4.1 Basic design 18](#_Toc513649830)

[5.4.2 Technical details 19](#_Toc513649831)

[5.4.3 Application Design 20](#_Toc513649832)

[6. Scenario supported 21](#_Toc513649833)

[7. NeCTAR Pro & Cons 24](#_Toc513649834)

[7.1 Advantages of Nectar 24](#_Toc513649835)

[7.1.1 Pre-built Linux image 24](#_Toc513649836)

[7.1.2 Well-designed API 24](#_Toc513649837)

[7.1.3 Well-designed Web GUI and QA document in detail 24](#_Toc513649838)

[7.1.4 Security strategy 24](#_Toc513649839)

[7.2 Disadvantages 24](#_Toc513649840)

[7.2.1 Unreliability 24](#_Toc513649841)

[7.2.2 Long response time 25](#_Toc513649842)

[8. Source Code 25](#_Toc513649843)

[Reference 26](#_Toc513649844)

# Roles and Responsibility

There are five people in our group, originally. However, due to some personal issue one of our team members (Adam) left one week before the due, and withdrew this subject.

|  |  |
| --- | --- |
| Member | Role and responsibility |
| Ziyue Guan | Automated Deployment |
| Xing Wei | Tweets harvesting and text analysis |
| Mingyuan Cui | Web application |
| Shiqi Wang | Data statistics |
| Adam | N/A. |

# 1. Introduction

The project mainly focuses on developing an integrated big data analysis program. This program can get a big volume of data from the Twitter and store them in the distributed database deployed on the NeCTAR cloud platform. Also, this program is able to process mass data according to the potential demands by Natural Language Processing (NLP) and normal Map/Reduce method. What’s more, the program is capable of implementing data visualization in web pages by the HTTP Server deployed on the cloud platform likewise.

# 2. Data Sources Introduction

In this project, data is harvested from the social media provider—Twitter. The project team developed a Twitter harvester to obtain raw data from Twitter API, and added extra keys that is corresponding with scenarios after analysis and disposal. We access the Twitter API under the License from Twitter Developer Agreement. (https://developer.twitter.com/en/developer-terms/agreement)

# 3. User Guide

## 3.1 Deployment

Deployment: a bash script is used for deployment. Only with one click, the user can create an instance on nectar and deploy it with environment to setup a new system or join already existing system. The script accepts three parameters, the first is command whether you want to setup or terminate instances. When command is setup, two additional parameters are needed. By that means, the second is the number of instances and the third is the volume size.

Example:

control.sh setup 1 0

## 3.2 Web Page

**Web page: Please visit the pages after connecting to the network of University of Melbourne directly, or VPN will be needed.**

**The front page can be accessed by typing 115.146.86.154/index.html into browser.**

**The link of our presentation video:**

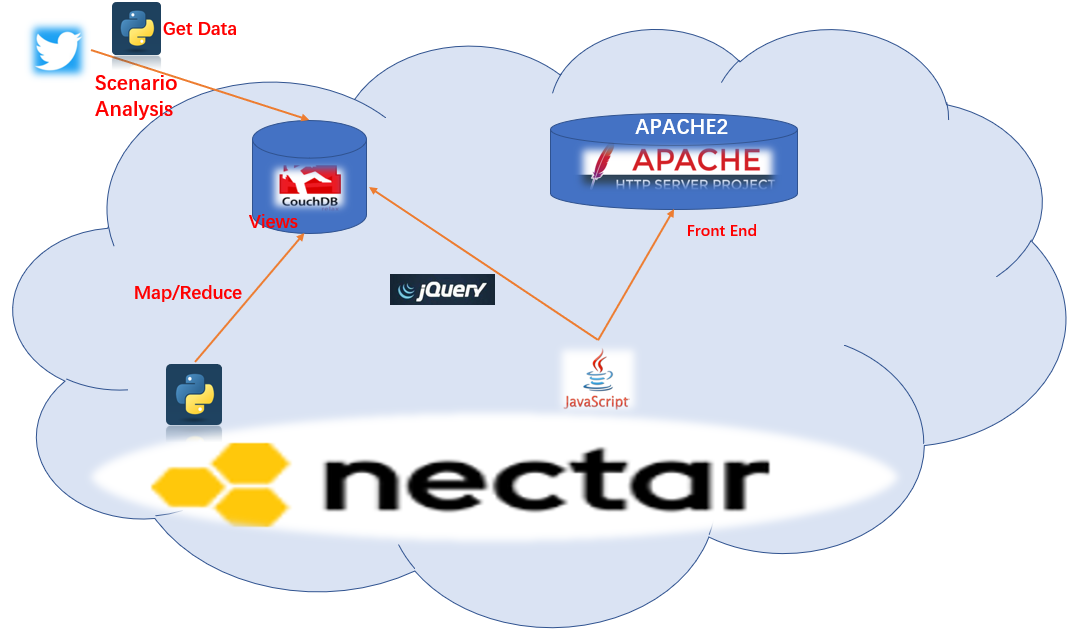
[**https://www.youtube.com/watch?v=-mI2W9Q7o-s&feature=youtu.be**](https://www.youtube.com/watch?v=-mI2W9Q7o-s&feature=youtu.be)

# 4. System Design

In order to develop a complete big data analysis program, the project team used the following processes:

**Figure 4.1** Processes to design the whole program

The whole system of the project is involved with several developing tools and platforms and these are shown in the Figure 2 as follows:



**Figure 4.2** Main tools used in the program

## 4.1 Automated Deployment

NeCTAR is an OpenStack cloud platform, Boto is a good choice to manage it. It directly accesses the API offered by NeCTAR with access token. Besides, Ansible is used for deployment tool. It is a free software and works under Linux/Unix based systems.

## 4.2 Data Collection

The Twitter harvest API that the project team chose is the Tweepy, a streaming API provided by Twitter, and it is an easy and efficient Python library for accessing the Twitter. The advantage of this streaming API is easy to get real time data. For collecting data, data comes from 4 main cities of Australia, which are Sydney, Melbourne, Brisbane and Perth respectively.

There are 4 scenarios in this project and these are shown in the table as follows:

|  |  |
| --- | --- |
| Scenario | Description |
| 1 | The content of tweets containing the information about beer |
| 2 | The content of tweet containing vulgarity |
| 3 | The content of tweet containing the information about crime |
| 4 | The analysis of sentiment. |

**Table 4.1:** Description of 4 scenarios

The analysis result about these 4 scenarios has been included in the gathered Json documents for statistic and analysis.

## 4.3 Scenario Analysis

For scenario analysis, the 4 different scenarios are harvested by 4 approaches.

A python library called vaderSentiment is used to analyze the sentiment scenario. VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media (VADER Sentiment Analysis, 2018). The method of manual annotation was used to determine the emotional polarity and intensity of commonly used 7000+ emotional words. The strong point is this method’s dictionary takes into account commonly facial expressions to solve the emotional discrimination of not standard sentences from Twitter. Also, this method is able to consider abbreviations for common emotions such as WTF, LOL.

A python library called WordNet is used to analyze the crime scenario. WordNet groups words according to the meaning of the entries. Each group of words with the same meaning is called a synonym set. WordNet provides a brief, synoptic definition for each synonym set and records the semantic relationships between different synonym sets (WordNet, 2018). The library can find whether the content of Tweets contains the crime words, such as “crime”, “robbery”, which are defined in a synonym set.

A Python library called profanity is responsible to check for (and clean) profanity in strings (Ben, 2013). If the content of Tweets contains some profanity words such as “shit”, “fuck”, the outcome would be true.

For beer scenario, there is no existing python library for analyzing this subject. The solution is creating a collection which contains all names of beer and all kinds of red wine. Using this collection to judge whether the content of the Tweets has information about beer.

## 4.4 Data Storage

In the project, the gathered data is stored in a CouchDB database cluster which has been deployed on the cloud platform. The CouchDB is a document-oriented database system developed with Erlang. CouchDB can distribute the storage system to many physical nodes, and it can coordinate and synchronize data read and write consistency between nodes. This database supports the Multi-Version-Concurrency-Control(MVCC) and the convenience of this mechanism is that reading and writing do not need to lock the database. What’s more, it is a storage system for web applications and it provides RESTFUL HTTP/JSON interface. With usage of CouchDB, members of project team not only have a data server that allows to dispatch, but also have a simple web server.

In CouchDB, project team can directly use JavaScript to access data, and then use html and Cascading Style Sheets(CSS) to display the data, after that, project team can also write the data back through JavaScript. This implements a simple content publishing system. Also, CouchDB supports REST APIs that allow to manipulate CouchDB databases using JavaScript and can also write queries in JavaScript. Especially, views are used to query and render documents. Through the user-defined View, it is easy to aggregate and get statistical data, using a Map / Reduce process. The developers of the project team can make use of the views in the database to get statistical data related with these 4 scenarios conveniently.

## 4.5 Data Statistics

In this project, the data analysis tool is Map / Reduce in the views of CouchDB.

Map/Reduce is a model that uses distributed computing to process and generate large data sets. This model can be divided into two steps: Map and Reduce. In the Map step, the master node receives the document and divides the problem into multiple sub-problems. Then publish these sub-problems to the work node, process it and return the result to the master node. In the Reduce step, the master node receives the results from the worker nodes and merges them to get the overall results and answers that can solve the initial problem. The foundation of the statistical classification is the city name and some special conditions meeting these 4 scenarios’ demands. Then the aggregation step is summing each scenario’s data for the corresponding city.

## 4.6 Result Visualization

The data visualization tools are Google Map JavaScript API and ECharts. The former is Google's own programming API that allows programmers around the world who are interested in Google Maps to develop their own Google Maps-based services and build their own map sites. It allows our team to embed Google Maps data into a website without having to set up their own map server. Thereby implementing a map service application embedded in Google Maps, and providing location services for project team with Google Maps map data.

Echarts is a free and open source visualization library using JavaScript and is compatible with most browsers (IE8/9/10/11, Chrome, Firefox, etc.). Echarts provides rich visualization types, such as line chart, histogram and pie chart. The community of this library is active and the related documents are rich. With this library, the data obtained from the CouchDB can easily be displayed by various visualization types. It is clear and obvious to show differences.

The project team then deploys the APACHE 2 on the NeCTAR and uploads related code in the directories of the APACHE 2. It is simple, fast, stable, free, and can be used as a proxy server. Hence it is chosen by the project team to be deployed on the cloud and show the content of front end.

# 5. Detailed design and Implementation

In this section, for each part of our application, the detailed design and the implementation will be described.

## 5.1 Scalable automated deployment of cluster VM environment

### 5.5.1 Background

Due to the limited speed of hardware development, currently the power of an individual server is far away from supporting a large-scale application. With such a background, cloud solutions, a multiple-node, scalable distributed system, is necessary for programs development. To meet various needs from different types of applications and the need of setup and configuration of a large amount of VMs in cloud service, many professional automation management tools are developed.

### 5.1.2 Requirement

Besides scalability and automated deployment, we also have other requirements about an automation tool. Firstly, good compatibility is important evaluation metric for automation tools. They are expected to be compatible with different cloud platform, hardware, OS, environments, tools and software. Secondly, a successful automation tool need a good solution to avoid any security problems during operation. Thirdly, as the users and potential users of cloud service may come from different fields and the usability and cost should be considered.

### 5.1.3 Automated cloud provisioning

As the management tools base on the API offered by cloud platform, although there are third-party programs, the tools recommended by the platform are usually first considered. For Nectar, a private Openstack cloud platform, Boto is a good tool to manage Nectar.

### 5.1.4 Types of automated tools

#### 5.1.4.1 Automated tools at system level

Many famous automated tools working at system level. They can be categorized by the methods they work on remote machines. Some are agent based which means an agent should be firstly installed on remote OS and the commands work base on the support of agent. There are also other tools working by pull some small program or a short code working like a software to achieve remote control. Many well-developed tools in this area, like Chef, Fabric and Ansible.

#### 5.1.4.2 Container technology

Container technology is an idea different with virtualization, which works within OS. It builds virtual environment according to the software and application user need and isolates resources above system level for each container. The automated deployment is a derivative function during this concept formation. There is a trend that container is become popular. In automated deployment it has some advantages which is that the usage of image simplifies the difficulty of configuration of some common but complex software. However, in contrast, the principle of virtual environment makes the deployment relevant to OS more complex. It also need more professional specialists, which means worse usability. Docker are up-rising solution in this area.

### 5.1.5 Introduction of Boto and Ansible

In this assignment, Boto used as a library in python, which is user-friendly with the widespread of python language. It directly accesses the API offered by Nectar with access token. Besides, Ansible is used for deployment tool. It is a free software and works under Linux/Unix based systems. There are a lot benefits of using Ansible. Firstly, it can work on various operating system as remote system, including windows, different kinds of Linux/Unix based system etc. Secondly, it works without any pre-configuration on remote machines. Thirdly, it uses SSH as connection, which solves security problems in connection process. Fourthly, the language, YAML, is simple enough for users with different level of computing skills.

### 5.1.6 Cloud provisioning

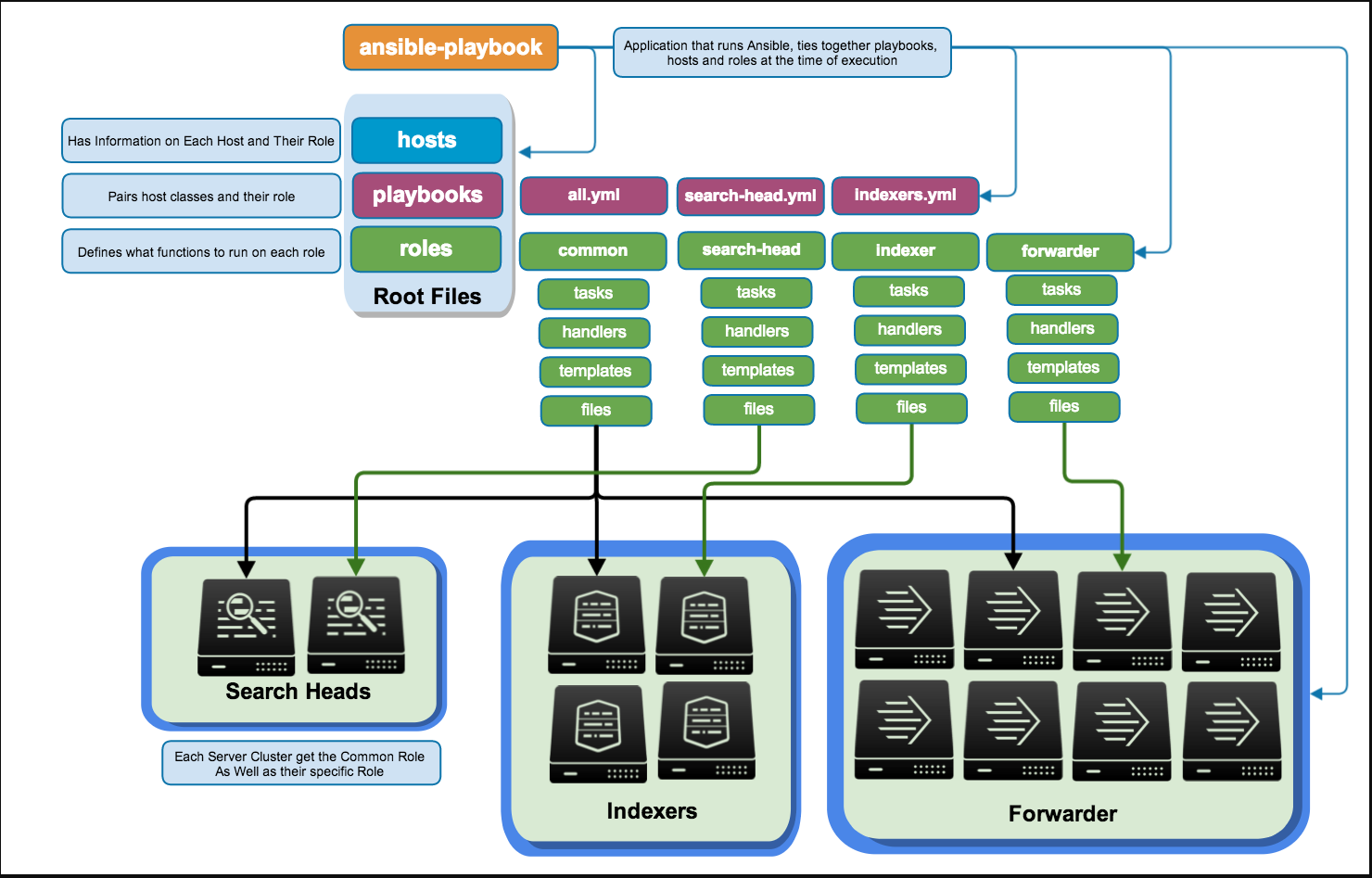
In this assignment, Boto in python is used for creating or terminating instances in Nectar. The script accepts three parameters including ‘-c’ which specifies the user’s need to create or terminate instances, ‘-n’ to indicate how many instances user want to create and ‘-v’ to indicate the volume size to be attached for each instance.

In boto script, a connection to nectar is first established and method ‘run\_instances’is called for creating instances while ‘terminate\_instances’ is used for terminating instances(Boto cloudhackers Document,2018).

In addition to these methods, the concept in nectar is showed in Boto as object. An ‘instance’ object contains nearly all parameters of an instance on nectar as attributes, including private\_ip\_address, status etc. By access these attributes, user can use program language to operate nectar, by which user can achieve more powerful and efficient operation than GUI.

### 5.1.7 Ansible functions

Ansible comprises several components. An inventory file is used to store information of candidate machines. A playbook script to call Ansible modules to operate remote machines (Ansible documentation, 2018).



**Figure 5.1.1** Components of Ansible (https://www.splunk.com/content/dam/splunk-blogs/images/2014/07/Ansible.png)

#### 5.1.7.1 Inventory

The default inventory in Ansible is hosts file located at /etc/ansible. In inventory level, Ansible offers several functions. First is group function, which means the machines in the inventory files can belong to different group. In our work, a machine is grouped into five groups by its role in the system. The five groups are common, CouchDB, harvestor, web, spark\_slave. Some configuration can also be set in inventory level. For example, the path of private key file is specified in inventory as our group members have different keys for different instances. This not only save the time to input the file path as parameters and avoid complex shell command but also lower the chance to expose the path of private key.

#### 5.1.7.2 Roles in playbook

To improve the structure, reuse and readability of code, the roles in playbook is designed for VMs with different functions. Each role has its own main.yml in corresponding tasks folder to list the commands it need to do. Besides, files folder stores files which are used during deployment. Templates folder is used with the template module in ansible. There are also some other folders supporting other functions, like handlers and vars, but they are not used in this assignment. In root folder, a root yml file gives instructions to VMs by indicating their roles as well as other command.

#### 5.1.7.3 Template and Jinja2

Template is a module in Ansible, which allows retrieve contents of file after templating with Jinja2. Accompanied with system variables in Ansible, files in which each needs individual value can be distributed with different value. In this assignment the template files are spark configuration file, which the variable ’spark\_local\_ip’ is assigned with remote host IP address.

#### 5.1.7.4 Variables in Ansible

Variables include system variables and custom variables. Custom variable can be defined in different levels to meet various needs. For example, variable in inventory, in group, in playbook. In this assignment, the system variable ‘ansible\_ip4\_address’ is gathered to specify CouchDB configuration and spark configuration.

### 5.1.8 Ansible implementation in this assignment

#### 5.1.8.1 CouchDB setup and configuration

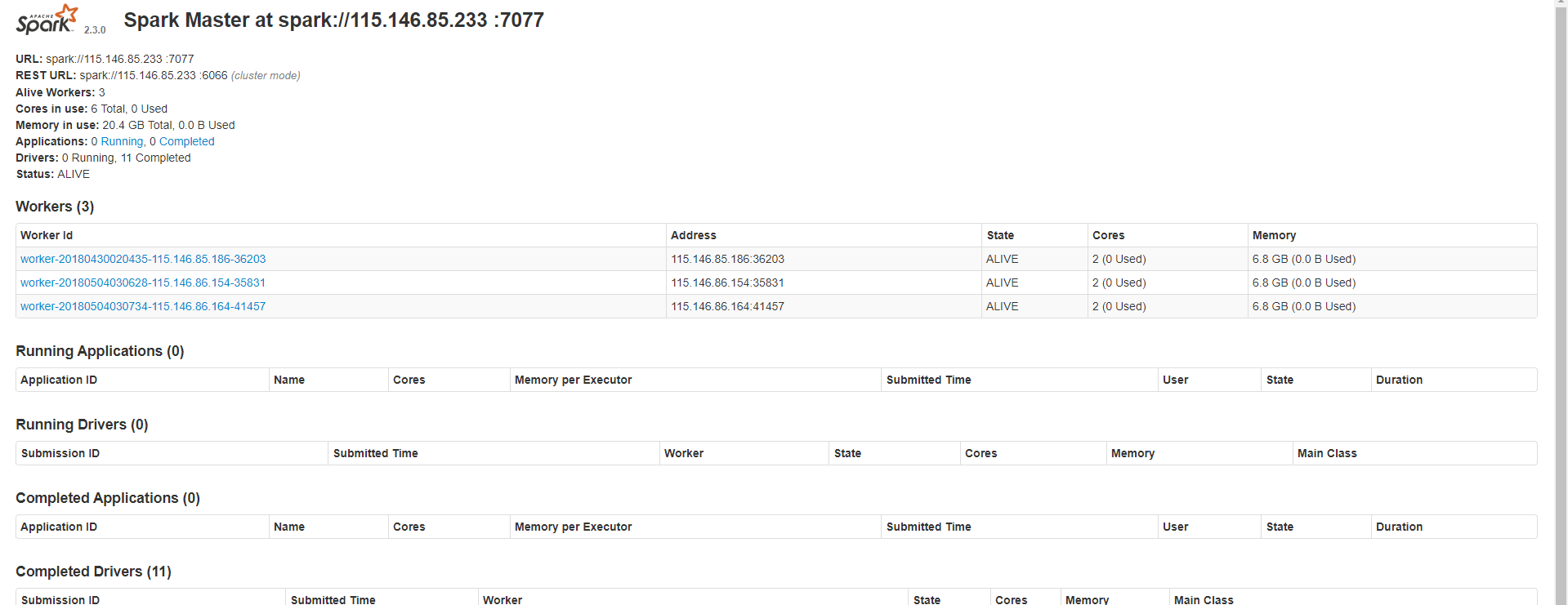
CouchDB is built from source code, which don’t need interaction but only command in the process. After that, some local configuration is set by modified the configuration file ‘local.ini’ in etc folder in CouchDB root folder. In addition to this, cluster setup is set up by http command through http API offered by CouchDB. The reason why two kinds of configuration is different is that local configuration is less likely to change after setup, but there may be more nodes joining this cluster. To keep the best scalability and availability, http API command which doesn’t need to restart CouchDB is better for cluster setup.

#### 5.1.8.2 Fault tolerance in CouchDB

A database should be stable enough to ensure its availability. Couchdb recommend run-it in documentation but it is already out of date. In this assignment, systemctl script is used as service in previous Linux system. Restart type is always.

#### 5.1.8.3 Spark setup

In this assignment, spark is installed for a good choice for parsing big data. Though we didn’t choose it actually, the spark cluster was successfully setup. Spark uses master-slaves structure in cluster mode. Both master and slaves are built from source code while they use different configuration file to start. A master node was setup to listen all the joining request from slave node at port 7077. And spark web UI is available at port 8080. A service is also setup to keep both master and slaves working. For worker in spark, they just need to join master and wait the resource manager to allocate work. In this assignment, resource manager is standalone.



**Figure 5.1.2** Spark Setup

#### 5.1.8.4 Harvester setup

We choose python as develop language for harvester due to the good support from massive natural language processing libraries. The environment for the harvesters is quite simple because the excellent package management from pip. As python is built-in software in Linux system. What the deployment need is to install packages through pip. A simple pip module in Ansible can easily solve this problem.

#### 5.1.8.5 web server

Our web service is based on apache. The web page is a static structure written in JavaScript. The apache service is installed via apt tool in Ubuntu.

## 5.2 Tweet harvesting and content analysis

In this section, the design of tweet harvesting will be described in detail. Apart from this, it will also be shown how the text is analyzed.

### 5.2.1 Overall Workflow

In our application, tweet harvesting and analysis is completed at the same time. The mission of tweet harvesting is easy to understand, capturing tweets from twitter APIs. While text analysis here is to check whether the tweet is relevant to the scenario supported.



**Figure 5.2.1** The workflow of tweet harvesting and analysis

As shown in Figure 5.2.1, each instance will run a program in the background and it can continuously acquire tweets from twitter API. For each tweet, the instance will preprocess the text first, so that it meets the requirement of analysis. Then, several analysis approaches will be applied on pre-processed text. After this, each tweet will be converted into the same format, and the reason for this is that it will be convenient for executing map reduce process. Apart from this, duplicate tweets are also handled in this process.

### 5.2.2 Program Design

The design of the program will be discussed in detail, in this section.

#### **5.2.2.1 Tweet harvesting**

The goal of tweet harvesting is to download tweets by using twitter APIs.

Using package Tweepy, this task becomes easier. Tweepy provides two approaches to access tweets in real time. One is streaming and another one is search. Both methods are tried to complete this task, but streaming is the only method which is chosen to handle tweet harvesting in this program.

The reason why search is abandoned is that there are some limitations on search API. Search method returns tweets that match a specific query. The parameters in this method must include a search query string. However, our expectation is that we harvest tweets regardless of the content of text. Geographic information of each tweet is also important to our application, as the analytics scenarios are based on cities. But most tweets acquired by using search API do not contains geocode. In addition, over a period of time, the tweet obtained through this API is fixed, not only the quantity, but the content as well. Therefore, search API is not appropriate for our application.

Tweepy provides a class called StreamListener to download twitter message in real time. In this program, we create a class tweetListener inheriting from StreamListener and override the on\_data function.



**Figure 5.2.2** The structure of class tweetListener

Figure 5.2.2 illustrates the structure of class tweetListener. Function on\_status() and on\_error() are used to print out system messages. When a tweet has been converted to a specified format (a format for map reduce), savetodb() will be called so that this tweet will be saved into CouchDB.

All operations on tweets are done in function on\_data(), such as transferring tweets into a dictionary, analyzing the content of text and saving tweets into CouchDB.

In addition, StreamListener supports downloading tweets in accordance to regional information, so we ask our 4 instances to download tweets from 4 different cities in Australia, including Melbourne, Sydney, Perth and Brisbane. Figure 5.2.3 shows the geographic information of the 4 cities.



**Figure 5.2.3** Geographic information for 4 cities

#### **5.2.2.2 Text pre-processing**

In this procedure, we remove meaningless information from original tweet text. Regular expression and some basic functions are used. Special characters such as hashtag (#) and at (@) are eliminated, as well as website links in the text. After doing this, the whole text is transferred into lowercase.

An example of text pre-processing:

**Original text:** Prof. Hui Gan at @ONJCancerCentre talks of generous donation by @carriebickmore &amp; Beanies 4 Brain Cancer Foundation https://t.co/gpSwGvZUB4

**Pre-processed text:** prof hui gan at onjcancercentre talks of generous donation by carriebickmore amp beanies 4 brain cancer foundation

The significance of this step is to transform the text into a form that is easy to be analyzed.

#### **5.2.2.3 Text analysis**

**Sentiment analysis**

English is a natural language, and there are many studies on how to analyze human emotions from language. In our application, we achieved sentiment analysis, with the help of vaderSentiment python package. This package is used for natural language analysis.

The function polarity\_score() is used to get the sentiment strength of the given text. This function will return a float number which indicates the emotion of the tweet. Positive value means the user posts this twitter message with positive emotion, and negative value corresponds to negative mood. In addition, zero means neutral attitude.

Examples of sentimental analysis:

**Text:** what how interesting students will love it

**Result:** Compound: 0.8306. This tweet is considered with positive emotion.

**Text:** tough day at the footy day seriously dont know whats happening

**Result:** Compound: -0.296. A tweet with negative mood.

**Text:** starting yet another show

**Result:** Compound: 0. Neutral emotion.

**Words relevant to crime**

By testing whether there are words related to crime in tweets, we can guess the public order in a specific region. It is difficult to figure out all words related to crime, so we use WordNet to complete this analysis. The WordNet library is supported by NLTK python package.

WordNet is a large lexical database of English where synonyms are distributed in the same network. First, we find our several typical vocabularies which are related to crime, such as assault, robbery, arson and extortion. For each tweet, we check whether it contains synonyms of those typical crime vocabularies. Thus, tweets about the crime are screened out.

**Vulgarity detection**

To detect vulgarity in tweet text, we use the profanity package. This package provides a dictionary consisting of insulting words. In addition, by calling function contains\_profanity(), it will return whether a text contains offensive words.

**Talking about alcohol**

It is almost impossible to detect alcohol consumption by analyzing text. Therefore, we assume that there is a positive correlation between talking about alcohol and alcohol consumption.

We use keywords search to determine whether alcoholic drinks are mentioned in the text. We investigated the brands of beer and types of wine in Australia and a list of them is generated. For each name in the alcohol list, if it appears in a tweet text, we mark this tweet with True. Otherwise, a tweet will be considered as False.

#### **5.2.2.4 Integration**

All obtained tweets are in the Json format, so they will first be convert to a dictionary. After analyzing the content of text, a new key will be created in the dictionary for each kind of analysis, and the result will be written into that key. Finally, it will be converted back to Json and saved into CouchDB.

### 5.2.3 Interaction model



**Figure 5.2.4** The interaction model of tweet harvesting and analysis

Figure 5.2.4 shows data flow and the interaction between functions in the application.

Before accessing to Twitter Streaming API, consumer key and access token will be authenticated. After this, a connection will be established with API, and the instance starts to download tweets (each one is a Json file). Then, as mentioned above, the content will be analyzed and the results will be added into the original tweet Json file. Finally, the new tweet Json file will be saved into the CouchDB cluster.

### 5.2.4 Other design issues

**Duplicate data**

To ensure the accuracy of analysis results, duplicate data should be removed from database. However, removing duplicate tweets is a job which will cost much computation. Therefore, our solution is that, instead of checking repeated tweets manually, we transfer this task to CouchDB.

When a Json object is saved into a CouchDB, two new keys are added to the Json document, “\_id” and “\_rev”. Key “\_id” is the unique identifier for each Json document in the database. While for each tweet, it also has a unique identifier “id”, which is different from that of other tweets. Our approach is that we create key “\_id” manually and fill it with the “id” of tweet. In this case, if a duplicate tweet is going to be added into CouchDB, the database will through an exception, and as a result, this tweet will not be appended into the database.

**Volume of tweets**

Up to now, there are about 200 thousand tweets in our database, and this number is still growing slowly. Because we have specified the source (from 4 cities) of tweets, the speed of harvesting is considerable slow. However, all of obtained tweets contains geographic information, or even coordinates.

A small number of Twitter messages is likely to lead to inaccurate results, so we tried to access the tweet data provided by Richard Sinnott. Because of the slow speed of download, finally, we decided to use twitter data that is captured by ourselves.

### 5.2.5 Analyses and improvements

**Volume of data**

To a certain extent, the volume of tweets data will affect the results of analytics scenarios. The twitter API limits the download speed. Therefore, to get enough tweets, an improvement is obtaining tweet data from multiple data sources.

**Intended data**

Through the observation of twitter data, it is found out that a large number of tweets does not contain coordinates. This brings difficulty to analyses on scenarios. At present, the scenarios supported by our application are city wide. If the scope is reduced to suburb, there will be very little data that satisfies to the scenarios in each suburb. As a result, based on little data, there might be error on the analysis, or nothing can be concluded from the analysis. Therefore, how to get adequate tweets with coordinates is the research direction in the future.

## 5.3 Detailed Data Statistics:

### 5.3.1 CouchDB-Python

In this project, the final approach to solve the statistical problem is making use of the Map/Reduce mechanism embedded in the view of CouchDB. The default view definition language in CouchDB is JavaScript. Because using requests, the only Non-GMO HTTP library for Python, safe for human consumption (Requests: HTTP for Humans, 2017.) for http requests, which is much faster than the standard library, with the help of the document couchdb-python Documentation (Dirkjan Ochtman,2018), the project team decides to use Python to operate the CouchDB by installing the library called couchdb in the Python.

### 5.3.2 Key Points in Data Statistics

#### 5.3.2.1 Views

The index of the data in CouchDB is finished by creating a view. Once the view is created, the system will automatically generate and save the data index file defined by the view in the background. It is noted that the query and index of data in CouchDB are completed in the process of querying the view at the same time, and the query of data is also an indexing process. Views can show the relationship between data and make statistics. This mechanism guarantees the efficiency.

#### 5.3.2.2 Map/Reduce

The implementation of the Map method needs to determine whether to output the result based on the content of the document object. If users need to output, users can use emit to complete. The kind of keys and values in the emit method ​​to use should be determined based on the actual needs of the view. When sorting and filtering a field in a document, users should use the field as a key or part of the key; It is noted that the value of the result can be provided to the Reduce method, or it can appear in the final result.

The output from the Map method is called an intermediate result. Intermediate results can be further aggregated through the Reduce method. The aggregation operation is performed on the condition that the keys of data are the same. Then users can get an aggregation result.

#### 5.3.2.3 Design Document

The Map function and Reduce function are included in the design document. Although the design documents are created in the same way as creating normal database documents, the content and definition is different. The design document contains the detailed definition of views which are used to get information from CouchDB. Design Documents are named using an ID defined with the design document URL path, and this URL can then be used to access the database contents.

### 5.3.3 Implementation Details

1. The first step is getting connection with CouchDB deployed on the cloud. It means getting a Server object, representing a CouchDB server. By default, it assumes CouchDB is running on localhost:5984. However, the server in the project has authentication credentials, that is, when connecting the database server, users need to add username and secret with IP address.

2. Creating a database, if it does not exist, or using the database directly. Also, the project team needs to create another database to store the processed data.

3. Creating views and designing the CouchDB design documents. This is the most important step in the whole Map/Reduce process. Views in CouchDB are saved via JavaScript and Map /Reduce functions are written in Python code. In order to obtain the statistical data meeting 4 scenarios, the basic idea for Map/Reduce functions is assign 1 to each document id once the special scenario demands are met by the filtering conditions. For example, if users want to obtain the total number of tweets from Melbourne, the Map function would assign 1 to each document containing the location where the tweet was sent is Melbourne and the Reduce function is responsible to sum documents which have the same key.

4. After creating design documents, it is easy to use the python method view() to get results and save these result into the new database created as mentioned above.

### 5.3.4 Eliminate the Choice of Spark

Apache Spark is a fast, general-purpose computing engine designed for large-scale data processing. One of significant features for Spark is that Spark can reuse related cached data when doing duplicate calculations, while Map/Reduce is awkwardly performing I/O streaming from the disk. Spark takes a code-parallel model and brings the code to the data, such as turning off serialization. This reduces network operations and local aggregation results can help reduce network IO operations. However, in this project, the data flow is not iterative, that is, the original statistical data does not become the input stream again in the system. This situation greatly constrains this advantage of Spark compared with Map/Reduce.

Also, there is another problem. Spark is a computing framework, not like Hadoop which has HDFS that can store data. The raw data is stored in the CouchDB, so if using Spark, an extra tool should be needed. After lots of searches, a tool called Bahir is found. Apache Bahir provides extensions to Spark and this tool is a library which can read data from CouchDB database with Spark SQL and Spark Streaming (Spark SQL and Spark Streaming, 2017). But the related documents are too few and it is really hard to deploy the Bahir in the cloud.

## 5.4 Web Application

### 5.4.1 Basic design

The main aim of this part is to make the result data visible. The details are as follows:

1. The data is stored in CouchDB, and the browser needs to send a request asking for the analysis results. The script is written in JavaScript, of course CouchDB has provided an API to deal with these requests, and by using this API, data in the database on the server can be easily obtained by the browser.

2. The returned data needs to be parsed by the script embedded in the html file. According to the keys in the returned JSON object, the corresponding values can be extracted easily.

3. Some data visualization tools are needed to show the maps or charts. Google Map JavaScript API is taken advantage of combining the exact coordinates of the chosen cities and the numbers from the database. And ECharts is used for charting in the data, which is an open source and powerful charting library.

4. Some basic stylesheets are designed and implemented, and buttons as well as selecting collections are added on the web pages to make viewing more conveniently.

### 5.4.2 Technical details

1. The jQery-CouchDB API (jquery.couch.js) provides methods to connect to the remote CouchDB. The request made here involves two concepts, the first one is the AJAX (Asynchronous JavaScript and XML) mechanism built in JavaScript and jQuery. To be exact, after a XMLHTTPRequest object is sent to the server, the browser will continue to run the rest scripts without waiting for this response. As soon as the response comes, the browser will go back to process it.

And here the browser is asking for the document which contains the analysis results from the CouchDB on the server, and there are two steps to realize this. First, by calling the login function in the API, the browser sends a request containing the username and password, then the server response with a hello message. Because we know exactly which database contains the result, there is no need to do some querying and choosing steps. Secondly, just calling the openDoc function and after a while the browser will receive the corresponding response. There are two callback functions dealing with this response, one for a success and the other for an error. It is worthwhile to mention again that because of the asynchronous mechanisms of AJAX, all operations with respect to this particular response have to be done in the callback function, or there would be some undefined object errors which seem to be mysterious, and this is because the browser just runs the script continuously without waiting.

The other related concept is CORS (Cross-Origin Resource Sharing) which means this request is asking resources from another domain, and it will certainly be prohibited by the security mechanisms. Fortunately, after setting Access-Control-Allow-Origin header in the server configuration and changing the CouchDB CORS option to “True”, the server can be accessed normally. As the server does not have a formal domain name, it is understandable that an issue like this may arise, but this is solved perfectly.

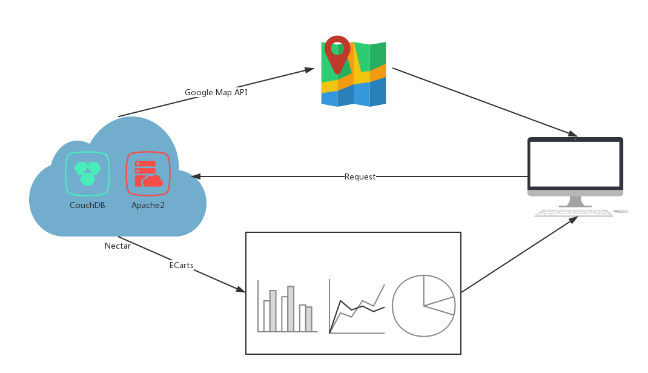
2. Google Map JavaScript API also has a multitude of features which can make the map more descriptive, and markers with labels as well as heat maps are used in this project. To illustrate, markers can be set at the exact coordinates on the map, and heat maps can demonstrate the variation of the numbers. The zooming rate and the center coordinates of the original map can be reset to show more details on particular regions. What is more, the styles of the heat maps and markers are redefined to realize better visual effects.

ECharts is also very simple to use with many customizable options from the kind to the detailed effects of a chart. It can turn the raw data to beautiful charts which demonstrate the features of the data quite well. In this project, pie/bar/line charts are all used, and settings are configured to make them more shining and colorful. Some important average values are also showed as marking lines.

3. After implementing and testing the web application, the deployment seems to be the last but most significant step. In this project Apache2 is used to make the web pages and data on the server accessible by others’ browsers. Apache2 has one directory explicitly used to deploy the web application (/var/www), after installing Apache2 on the instance, copying the scripts as well as resources to the root folder, and then adding CORS header settings, now the web pages can be visited by other people.

### 5.4.3 Application Design

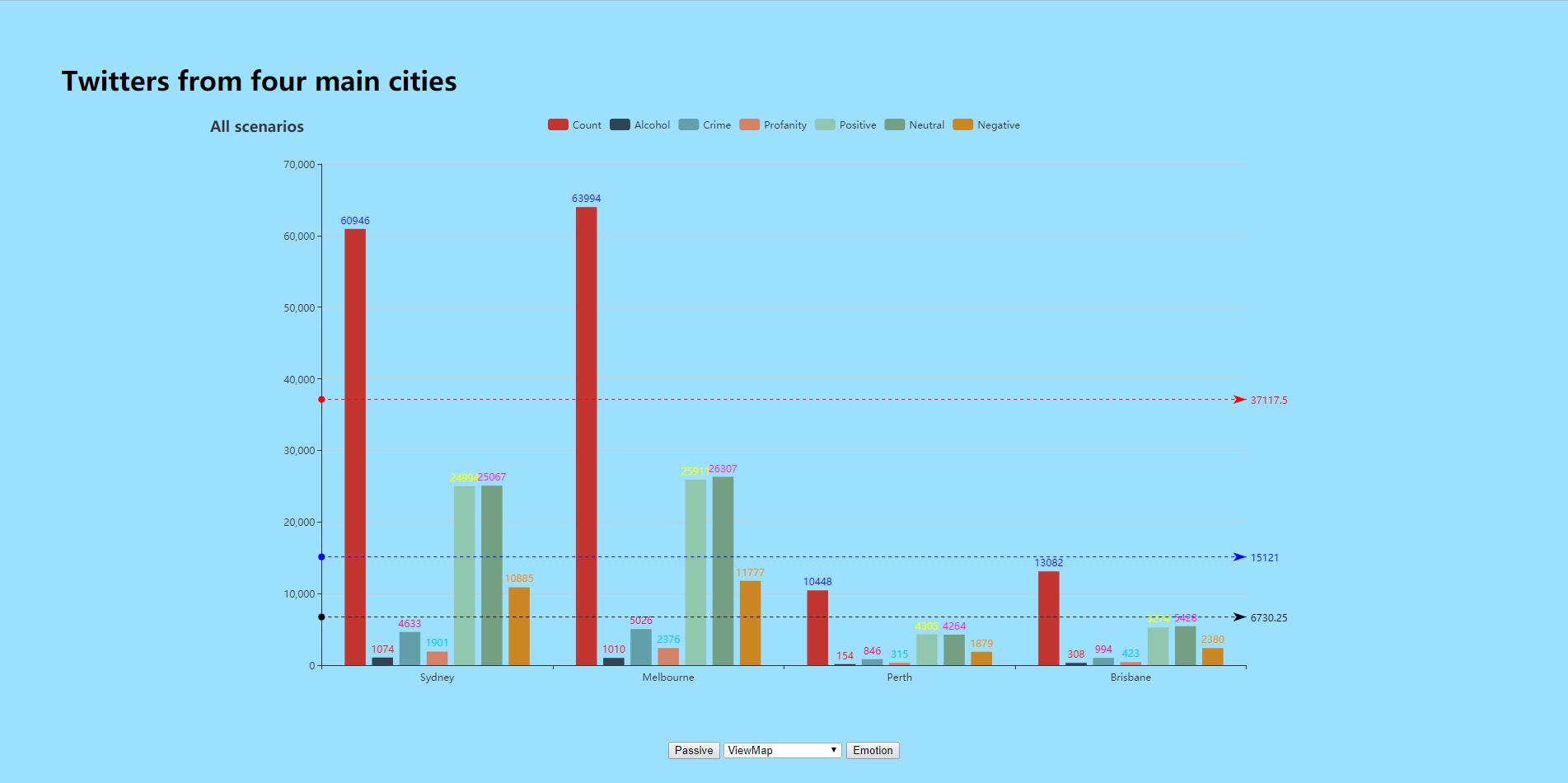
As the main duty of this web application is to show the correlated figures precisely and appropriately, there is no need to add too many visual effects which may make the charts look confusing and messy. Besides, based on the size if the charts generated, one page can at most contain two figures. So finally there are three chart pages, and in order to illustrate the geographical factors better, four pages containing maps are added. The architecture of this web application is like Figure 5.4.1.



**Figure 5.4.1** The interaction model of tweet harvesting and analysis

# 6. Scenario supported

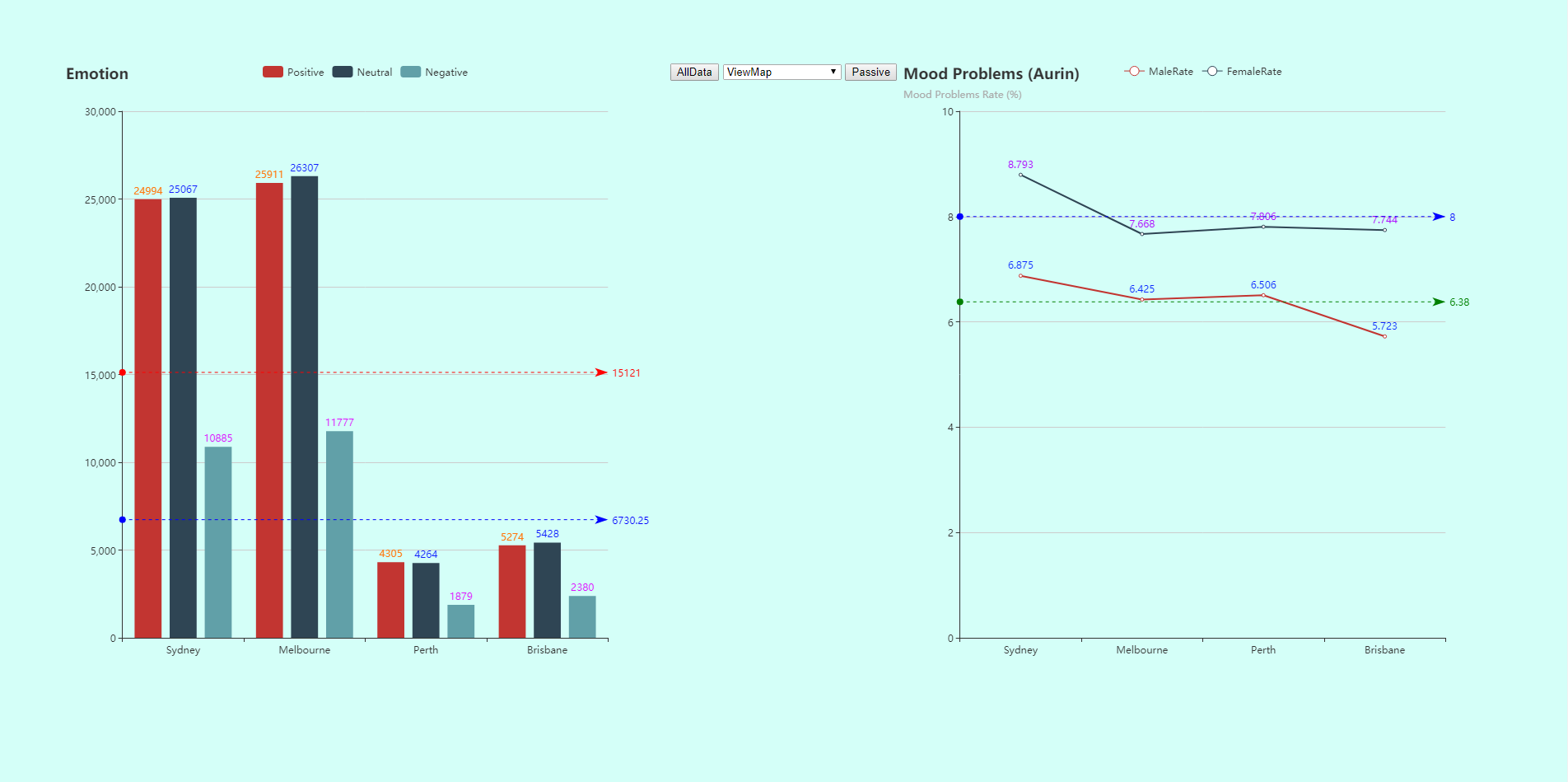
Figure 6.1 is the front page of our application, and all count results of analysis are show on this page. We analyzed the obtained tweets from 4 aspects, the emotion contained in the text, whether the content of the text is related to crime or alcohol, and whether the expression of language is vulgar. In addition, we have made a separate statistic for every city. By clicking the cutline, it is possible to hide and display the corresponding count result. The purpose of the front page is to give the audience an intuitive understanding on geographic distribution of tweets.



**Figure 6.1** Front Page

It is obvious that most tweets that we harvested are from Sydney and Melbourne, while only about 25 percent of total tweets are from Perth and Brisbane.

One of the scenarios that our application supports is that do people feel happy living in Sydney, Melbourne, Perth or Brisbane.

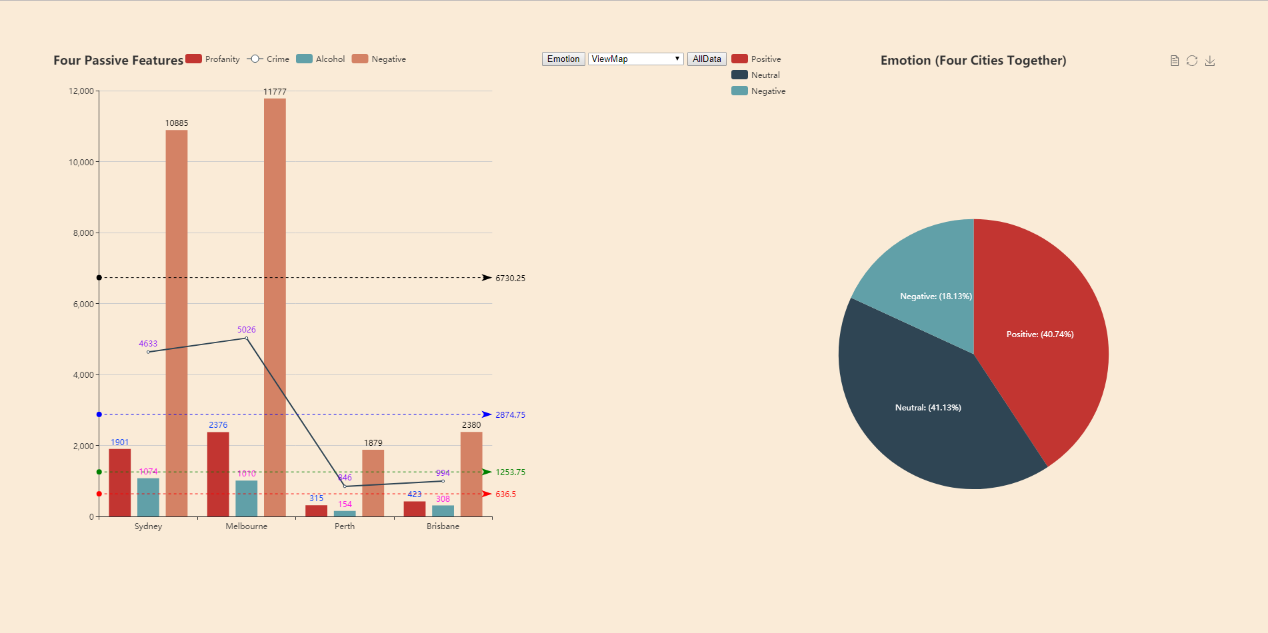


**Figure 6.2** Emotion analysis

As shown in Figure 6.2, no matter which city it is, the count of tweets which indicate positive mood is almost two times of that of negative emotion tweets. Although the total counts of tweets from the four cities are different, the trends are same, and the majority of people do not express negative emotion by positing tweets. This phenomenon indicates that most of people are happy in the four cities.

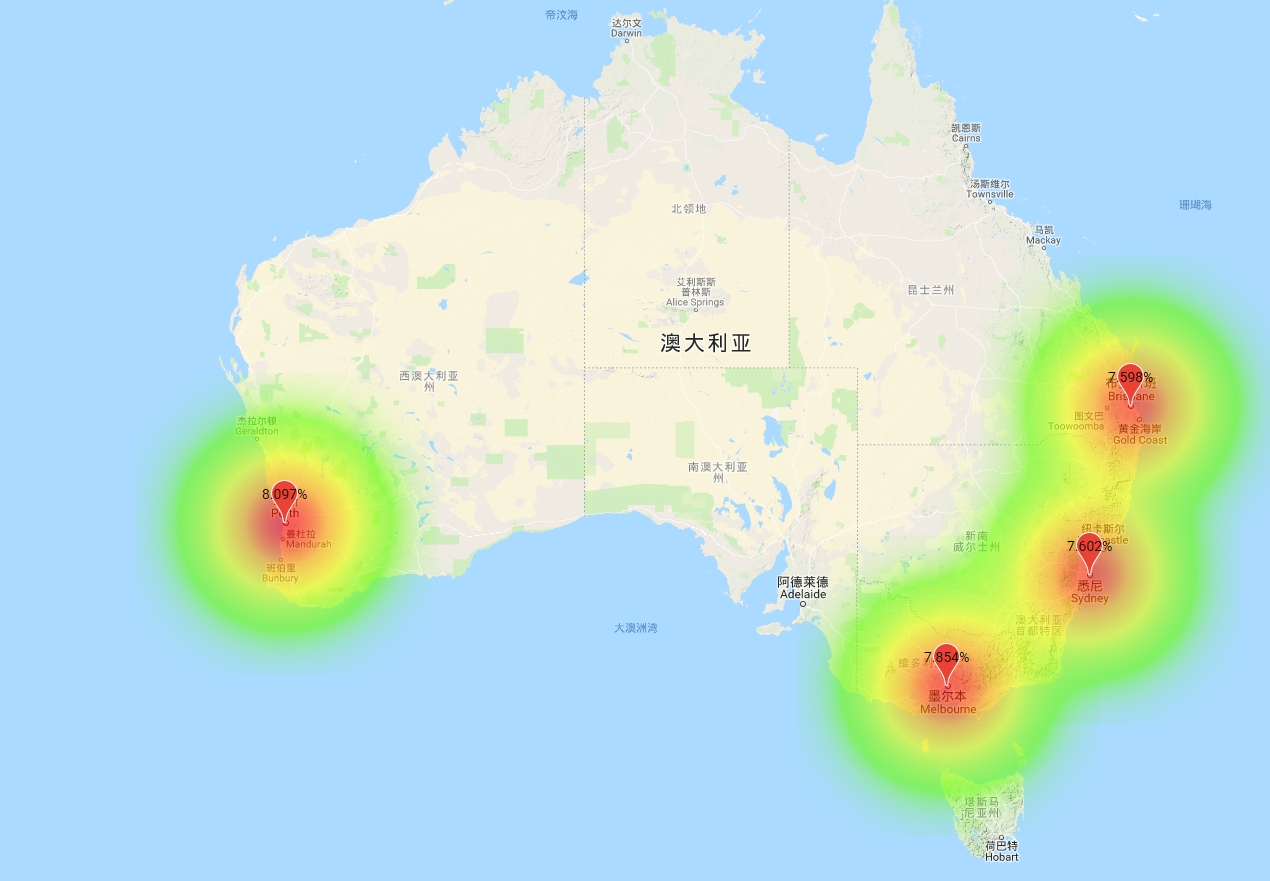
The line chart on the right shows the mood problem rate in four countries, which is generated with Aurin data. It is discovered that the mood problem rate in these four cities is about 7% on average, and it is lower than the analysis result (about 18%) from tweet. One possible reason might be that the sentiment analysis based on tweet text is inaccurate. Another explanation to this is that people are more likely to post tweets when they feel bad. Interestingly, female have a higher possibility to meet mood problem than male.

Another scenario is which topic does people talk about much, crime or alcohol. From the bar chart (Figure 6.3), it is clear that there are more tweets which contain vocabularies related to crime. According to the figure, we consider that people pay much attention to crime issues, rather than alcohol consumption.



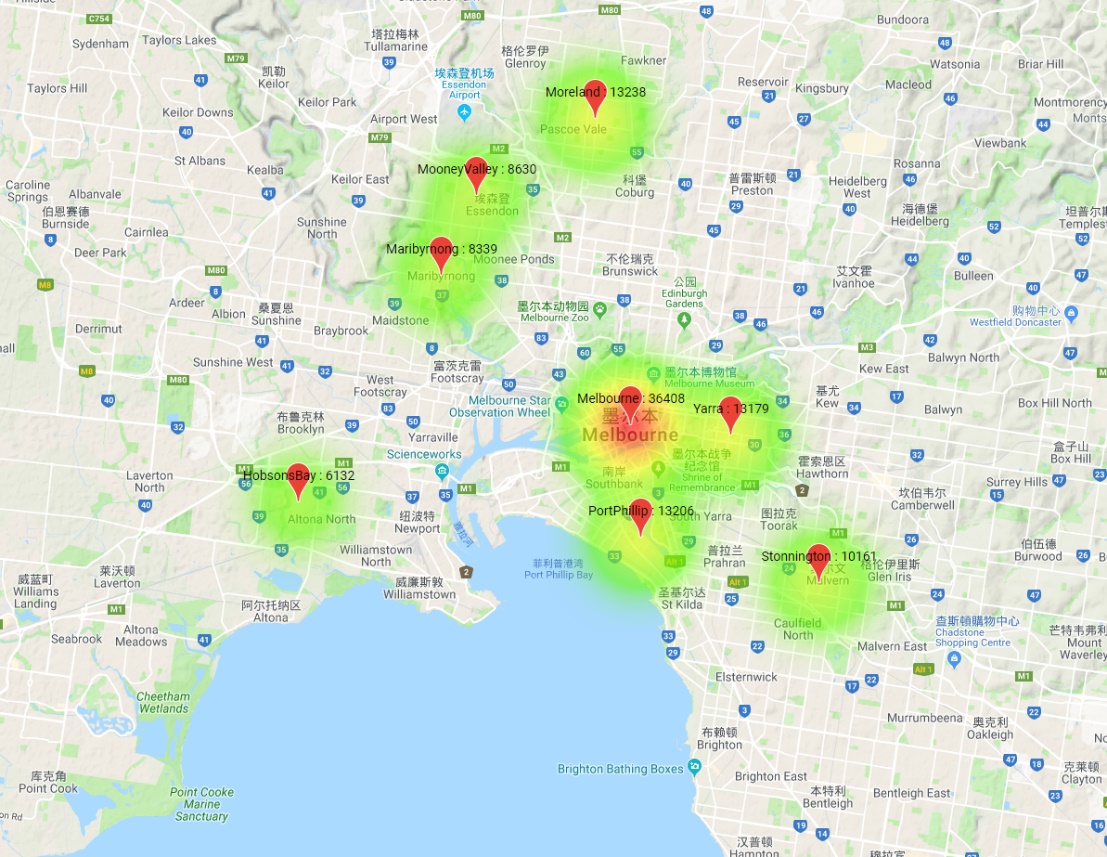
**Figure 6.3** Four passive features

In addition, the bar chart shows a relation between vulgarity and negative emotion. The increasing of people with bad mood causes more appearance of vulgarity.



**Figure 6.4** The proportion of tweets of talking about crime.

Our application also supports showing the result on google map. Figure 6.4 illustrates the proportion of tweets of talking about crime in the four cities.



**Figure 6.5** Crimes in Melbourne (Aurin)

The original idea of us is to mark tweets about crime activities on google map. However, due to limited data downloaded from twitter API, we collect crime data form Aurin and print them on map.

# 7. NeCTAR Pro & Cons

## 7.1 Advantages of Nectar

### 7.1.1 Pre-built Linux image

Nectar allows setup an instance from selected image which save a lot of time and avoid potential problems when system being installed. This improves the usability of cloud service and efficiency. There are also various OS images for choosing, which suits users with different needs.

### 7.1.2 Well-designed API

As a cloud platform, large-scale deployment is significant. Nectar offers different API to various language. The Boto tool in python uses popular Object-Oriented programming design, which is easy to use for programmer from different area. It has precise documentation of the API, which is friendly to users who are first to nectar and even new to cloud service.

### 7.1.3 Well-designed Web GUI and QA document in detail

As a national research infrastructure for Australia, users who use nectar may come from different areas may be lack of programming skills. At such situation, a well-designed Web GUI can help them manage cloud resource and facilities better. With reading some lucid tutorial and follow instructions can help them easily get into nectar while the good QA documentation.

### 7.1.4 Security strategy

Nectar uses security group as default security strategy rather than built-in firewall, which protects users properly. For users who have less computing skills, the can finish their work with in VM rather than protecting their information by their selves. For developer, using protection from cloud platform release them from heavy work of configuration of firewalls. Centralization management means it is easy to keep all patches up-to-date and to save the computing resource from being exhausted in protection attacks, like DDoS.

## 7.2 Disadvantages

### 7.2.1 Unreliability

Nectar sometimes responses timeout to the requests which has a negative impact on using. Web server of nectar may consider more concurrent visiting requests from users and website. Availability is a necessary requirement of web service.

### 7.2.2 Long response time

Nectar sometimes is not able to update the information in real-time. For instance, when an instance is terminated, it may still occupy the position in limit summery and need minutes to become consistent. This kind of problems seldom occur but may result in bad experience.

# 8. Source Code

All Source Code can be download from <https://github.com/guanziyue/too>.

# Reference

AJAX examples. (2018). Retrieved from <https://www.w3schools.com/js/js_ajax_intro.asp>

Ben, F. (2013). Profanity. Retrieved from https://github.com/ben174/profanity

Boto cloudhackers documentation (2018)

Retrieved from <http://boto.cloudhackers.com/en/latest/ref/ec2.html#module-boto.ec2>

Ansible documentation (2018)

Retrieved from http://docs.ansible.com/ansible/latest/index.html

Echarts documentation. (2018).

Retrieved from https://ecomfe.github.io/echarts-doc/public/en/index.html

Enable cross-origin resource sharing. (2018). Retrieved from <https://enable-cors.org/>

Google Map JavaScript API documentation. (2018).

Retrieved from <https://developers.google.com/maps/documentation/javascript/?hl=zh-cn>

Hutto, CJ. (2018). VADER Sentiment Analysis.

Retrieved from https://github.com/cjhutto/vaderSentiment

jQuery-CouchDB-API. (2014). Retrieved from <https://github.com/apache/couchdb-jquery-couch>

Requests: HTTP for Humans. (2017). Retrieved from http://docs.python-requests.org/en/latest/

Spark SQL and Spark Streaming. (2017).

Retrieved from http://bahir.apache.org/docs/spark/current/spark-sql-cloudant/

WordNet. (2018). Retrieved from https://wordnet.princeton.edu/