# COMP90024 Cluster and Cloud Computing

# City Analysis on the Cloud

Group Assignment 2 Report by Team 80

Qiyi Zhang qiyizhang@student.unimelb.edu.au

Xiangtian Zha xiangtian@student.unimelb.edu.au

Yifan Zhu zhuyz5@student.unimelb.edu.au

Yuzhe You yuzyou@student.unimelb.edu.au

Zhe Li lzl@student.unimelb.edu.au

May 25, 2021

## Contents

**1 Introduction** **2**

**2 Scenario Planning** **3**

2.1 Scenario 1 . . . 3

2.2 Scenario 2 . . . 3

2.3 Scenario 3 . . . 4

**3 Data Collection** **4**

3.1 Twitter Harvester . . . 4

3.1.1 Searching API . . . 4

3.1.2 Streaming API . . . 5

3.1.3 Twitter Corpus from Private Database . . . 6

3.2 Supplemental Data Collection . . . 7

**4 Data Pre-processing and Storing** **8**

4.1 Implementation of Pre-process . . . 8

4.2 CouchDB . . . 9

4.2.1 Four Databases on CouchDB . . . 10

4.2.2 MapReduce Views . . . 10

**5 System Design and Architecture** **11**

**6 Melbourne Research Cloud** **12**

6.1 Pros . . . 12

6.2 Cons . . . 13

**7 User Guide** **13**

7.1 System Deployment . . . . . 13

7.2 Data Upload . . . . . 14

7.3 Back-end Web Framework . . . . . 14

7.4 Front-end Application . . . . . 14

**8 Data visualisation and Analysis** **15**

8.1 Scenario 1 . . . . . 15

8.2 Scenario 2 . . . . . 19

8.3 Scenario 3 . . . . . 22

**9 Conclusion** **24**

**10 Appendix** **25**

# 1 Introduction

As social networking has become an inseparable part of everyday life, people are more than willing to express themselves on social media such as Twitter. On average, around 6000 tweets are posted per second worldwide on Twitter, which corresponds to 500 million tweets being generated per day. The huge amount of data provides opportunities for students and researchers to conduct various data analysis. The analyses assist them gain insights into social media and potentially make a significant impact on society. However, it is not trivial to develop a solution which can support big data analysis. Therefore, our solution is the cloud-based system which exploits four virtual machines (VMs) across the Unimelb Research Cloud with an associated CouchDB database. The utilized cloud service is the Melbourne Research Cloud (MRC), which provides an computing infrastructure to collect, store and analyze Twitter data. The key message of our research is not only a solution to big data analysis, but also a recipe to automatic deployment in cloud computing architecture using Ansible.

As Australia's population is rapidly growing and mainly gathers in major cities. The population in Sydney, Melbourne, Brisbane, Adelaide and Perth are forecast to double by mid-century. Using both the Stream and the Search API interfaces, our Twitter harvester has collected tweets located in these cities together with those in Australia's capital city Canberra. Besides, the tweet corpus from Richard Sinnott's private database are exhaustively utilized to make up for the restriction posed by the non-academic account in Twitter API. To understand Twitter data better, we integrate other data resources from the Australian Urban Research Infrastructure Network (AURIN) in our data analysis. Our research performs sentiment analysis on Twitter data and correlates it with AURIN data relevant to economic activity and socio-demographic trends. 3 scenarios will be introduced later, which reflects multiple aspects of people's life in Australia. Using spatial and statistical analytic tools, we model and test ideas related to the urban settings, development and well-being. The analytical result is displayed in a web application without authentication or

authorization of users.

The remainder of this paper is organized as follows. **Section 2** introduces the supported scenarios. **Section 3** describes how we collect essential data. **Section 4** gives the methods by which we pre-process the raw data and save them to couchDB. **Section 5** presents the system design and architecture, along with the reasons for adopting such a style. After experiencing the Melbourne Research Cloud, we discuss its advantages and disadvantages in **section 6**. **Section 7** provides a user guide to the system deployment, invocation and usage by end users. **Section 8** shows the detailed analytical scenarios and graphical results. Last but not the least, we conclude the paper by discussing the fault-tolerance in the software setup and the scalability of our application and infrastructure in **section 9**. The link to our GitHub repository and our demonstration video in YouTube are given in the **appendix**.

# 2 Scenario Planning

To fully show how to take advantage of Twitter big data in city-level macro analysis in a comprehensive way, we have formulated in total 3 scenarios for our city analysis. Each scenario is supported by a different topic and will use different tweet databases containing different sources of Twitter data (as specified in **section 4.2**).

## 2.1 Scenario 1

The first scenario aims to discover how Chinese-Australia relationship and Australian people's views on China are influenced by the breakout of COVID-19.

The global pandemic of COVID-19 has influenced the way we study, work, and interact. Due to the first sudden outbreak of corona virus last year in China, discrimination against the Chinese people and even Asians has become increasingly serious around the world, including Australia. Around 60% of Asian Australians surveyed in a recent report by the Scanlon Foundation suggested that during the pandemic, racism is a fairly big problem. Also, people are becoming divergent in views and distrusting of opposing opinions in the time of crisis. This scenario aims to discover how Australian people's views on China are influenced by the spread of COVID-19, where the sentiment score for the tweets is used as the indicator. This scenario will also explore how the sentiment score differs by location and how this difference is correlated with demographic characteristics and macro-economic indicators from other data sources.

## 2.2 Scenario 2

The second scenario aims to discover which suburb is considered as the most unsafe place in Melbourne.

There are hundreds of suburbs in Melbourne and people always have a preference when choosing

which suburb to live. Some suburbs are considered as comparably unsafe, which may be due to the frequency of criminal cases or population distribution. This scenario tries to find which SA2 area is considered as the unsafest place in Melbourne, where the proportion of tweets containing certain keywords is used as the indicator. This scenario will also explore how the result above is correlated with demographic characteristics and economic indicators of each SA2 area.

### 2.3 Scenario 3

The last scenario aims to discover which major city in Australia do non-English speaking immigrants favour most.

Australia is a country of immigrants and it is also the first choice of the destination country for many immigrants at the same time. Thousands of overseas travellers come to live in Australia every year in order to obtain Australian immigration qualifications, of whom a significant number comes from non-English speaking countries. This scenario aims to discover which major city in Australia has the most non-English speaker living there by calculating the proportion of non-English tweets made during the last month. This scenario will also explore how the result above is correlated with demographic characteristics and economic indicators of each city.

## 3 Data Collection

### 3.1 Twitter Harvester

Twitter provides two types of API to harvest tweets. To access the APIs, the python package `Tweepy` is used in this paper. To maximize the number of tweets harvested, we use both the Searching API and Streaming API. Both APIs are used for collecting the tweets made since this assignment was released. Historical tweets are collected from the Richard Sinnott's database. Since only 1% to 2% of all tweets via the available Twitter APIs are geo-tagged[1], the historical data with coordinates is a good supplementary data which allows spatial resolution of our analysis from the city level to a refiner level. Note that all raw twitter documents are pre-process before saving to CouchDB, which will be discussed in **section 4**.

#### 3.1.1 Searching API

The search API resource is retrospective and can collect tweets from the past. It requires authentication and controls the rate limit. We first sign up five student developer accounts and create Standard Projects. Each project contains an application where we can generate the required authentication credentials such as keys and tokens. Searching API is limited to 180 requests per 15-minute window for each user authentication. Since each request is up to a maximum of 100 tweets, this limitation results in 18,000 tweets per 15 minutes.

The searching API collects relevant tweets by matching the particular queries. Therefore, the parameter `q` (query) is required to search for tweets. To be noted, searching API cannot return

all tweets which match the searching criteria because not all tweets are indexed or available to the search interface. The search queries are created with a set of operators to match on tweets and user attributes, such as keywords, hashtags, and URLs. Since we do not constrain on specific topics or keywords at the data collection phrase, we set the parameter `place` equal to a specific city ID in the query for each round of search. To find city IDs, we use twitter geo-search API by querying, for example, `Melbourne` and set `granularity="city"`, and the city ID of Melbourne can then be returned.

There are two types of geo-tagged information associated with each tweet, which are attributes `coordinates` and `place`. If a Twitter user has enabled location based services, this feature of Twitter automatically adds the information of neighbourhood, city or state to each tweet the user has tagged. However, as we have discussed before, a place-tagged twitter cannot guarantee it is also coordinate-tagged (and non-academic Twitter account are prevented searching coordinate-tagged tweets directly). Nevertheless, using the above method still allows us to harvest as many tweets located at city level as possible. This is why our data harvester can support any scenario analysis by city level.

Once we have set up the query parameter and start receiving tweets, Searching API supports navigating the results by tweet ID and time. Since we run the search harvester on an hourly basis using Linux command *crontab*, our use case is a near real-time "listening" of tweets in major cities. We did not adopt the method of following the user's timeline to harvest tweets with coordinates because of the unreliability of individual users in adding coordinates or not.

To exclude duplicates during harvesting, we set the tweet ID as the document ID in our database and take the advantage of `max_id` and `since_id` request parameters in `tweepy`. Every-time a pre-scheduled harvest started up, it firstly found the largest document ID, which represents the newest tweet in our database. Then we set `since_id` to it plus 1 to ensure the oldest tweet to be harvested is still newer than the newest tweet in our database.

To exclude truncated tweets, we set the tweet mode as extended in the query. In 2017, Twitter expanded the character limit from 140 to 280. This measure retains the brevity of tweets and enables more expression in micro-blogs. Instead of the attribute `text` returned by the compatibility mode, the attribute `full_text` is returned using the extended mode. The completeness of tweet text leads to a more accurate result in later sentiment analysis.

### 3.1.2 Streaming API

The Twitter Streaming API allows us to stream tweets from the platform in real-time. We can also apply filtering criteria to harvest tweets in the major cities. Rather than the city IDs used in RESTful Searching API, a bounding box is placed for each city to filter tweets in Streaming API. The bounding box, written in the form of `[west_long south_lat east_long north_lat]`,

matches against the coordinates of a tweet when present or against a place polygon if the polygon is fully contained in the region. In the Streaming API, if a tweet text is no more than 140 characters, the harvester extracts a tweet message from the field `text`. If the tweet message is longer than 140 characters, a new `extended_tweet` field is added in the tweet object to hold the longer message and retain the completeness. The harvester instead uses the `full_text` field of the `extended_tweet` object to collect the complete and untruncated tweet message. Another comparison of the Streaming API to the Searching API is that once we establish a connection to the streaming endpoints, a long lived and persistent HTTP request is made. The Twitter server delivers real-time tweets through the connection. The script can run continuously, but sometimes the network setting will terminate the connection. To properly handle errors such as reaching the rate limit, the program would voluntarily disconnect from the server.

#### 3.1.3 Twitter Corpus from Private Database

Since Twitter's searching API only allows us to harvest the tweets created over the past one week without an fee-paid account and the date on which we initially successfully set up our harvester is already in the mid of April of 2021, it would be unable to collect all essential historical twitter data in our scenario analysis by using Twitter's searching API exclusively. Fortunately, Richard Sinnott has provided his private tweet corpus database to us, which contains all twitter data made in all Australia state capital cities since 2014. He has also provided a view-based data access approach, by which users are allowed to filter the corpus to download by city's name and date.

To ensure we have enough historical tweets for analysing scenario 1 and scenario 2, we have decided to collect all twitter data made in Sydney between 2019 and 2020 as well as all twitter data made in Melbourne since 2014. However, one challenge in accomplishing this is the scale of the data. On average, there are tens of thousands of tweets per day per city and the corresponding raw data file is around 300 megabytes (MB), which means the total size of all raw data we need will go beyond 500 gigabytes (GB). Three potential consequences may arise:

1. Our personal computers or cloud volumes may not have enough space to store the data.
2. CouchDB will take an unacceptable long time to bulk upload and generate view results at the stage of deployment, given only 2 chips and 4GB memory per each VM.
3. Download will take days (or weeks dependent on bandwidth) and any network instability will discontinue or even fail the download process.

Two methods are introduced here to address this issue. Firstly, we don't need to download all required raw data at once. Instead, we have created a shell script in the Twitter Analysis, which enables us to download the raw data day by day automatically in chronological order. Once the current download task ends, the raw data file is immediately handed over to a Python script to pre-process. The next shift of the download task will begin only if Python has finished processing the whole current file. We arbitrarily set the name of all downloaded files as the same name in the same folder, by which a new file will overwrite the old one and the space for only one raw

data file is needed (which is around 300MB). Secondly, the Python script does not upload each whole twitter data document to CouchDB. Instead, only relevant fields helpful for our scenario analysis will be saved and the rest fields are just discarded. Normally, a tweet document contains numerous irrelevant information for our scenario such as user profile or retweet status. By applying this process, the storage space required will be reduced to less than one hundredth of the original, which will greatly simplify the deployment. Details of the implementation will be introduced in **section 4.1**.

## 3.2 Supplemental Data Collection

The first supplemental data is from the Australian Urban Research Infrastructure Network (AURIN). To complement our urban knowledge in economic and socio-demographic aspects, AURIN provides us with the related data-sets. The source of AURIN data is from the Australian Bureau Statistics (ABS), the national statistical agency. The agency conducts the national Census every five year. The latest census information on income, economy, employment, etc. are 2016, and the next Census is August 2021. Limited by the out-dated nature of AURIN data, we will analyze its correlation rather than causation with Twitter data. Future studies can incorporate the Twitter data analysis with a newer version of Census.

To analyze data within regions across Australia, AURIN data is available for various geographical levels, such as SA2 and GCCSA. Based on the 2016 edition of the Australian Statistical Geography Standard (ASGS), while GCCSAs are designed to provide an accordant boundary of the functional extent for every Australia's capital city, AURIN data with a GCCSA level can be used in Scenario 3. While Statistical Areas Level 2 (SA2) are medium-sized areas representing communities that interact socially and economically, AURIN data with a SA2 level can apply in Scenario 1 & 2.

While we can access AURIN data in JSON, CSV files either through its open API or from its portal, we manually download the CSV version of multiple data-sets and then integrate them into two data-sets by joining tables based on the keys of `gccsa_code_2016` and `sa2_maincode_2016` respectively. Relevant attributes will be discussed in **section 8**.

The second and third complementary data-sets are two geoJSON files called *melb.json* and *sydney.json*. These are SA2 boundaries of greater Melbourne and greater Sydney, and the latter excludes 30 suburbs in Central Coast. Therefore, there are 309 and 697 entries in the respective geoJSON file. The two files are converted from the official *shapefile* for each Australia SA2 area, which is downloaded from the ABS official website under the file name of "Statistical Area Level 2 (SA2) ASGS Ed 2016 Digital Boundaries in ESRI *shapefile* Format".

# 4 Data Pre-processing and Storing

## 4.1 Implementation of Pre-process

Pre-process is not only applied to Twitter corpus collection but also to the tweets harvested by Twitter's API. The pre-process mainly consists of 3 steps: adding extra (set of) fields, removing irrelevant fields and deciding whether to upload or not.

1. Adding extra (set of) fields.

   (a) The first field is the sentiment score, which is calculated directly by Python package `afinn` [2]. The `afinn` package provides a well-defined sentiment lexicon or dictionary by including obscene words and Internet slang abbreviations. Due to the brevity and informal nature of tweet texts, incorporating these terms in the sentiment lexicon we employ reaches a better performance of semantic analysis.

   (b) The second set of fields calculates the number of occurrences of keywords corresponding to each pre-selected topic. We have adapted in total 5 topics to satisfy the scenario analysis requirement: alcohol, China, COVID, crime, and curse words. A list of keywords is arbitrarily selected for each topic, which contains 10 to 25 words depending on the broadness of the topic (e.g., China has only 10 keywords while alcohol has 25 keywords).

   (c) The last set of fields contains the Statistical Area 2 (SA2) code and name where the Twitter was created, if coordinate information is included in the raw data entry. If no coordinates are provided, those fields are set to null by default.

2. Removing irrelevant fields. Removing irrelevant fields. Only useful fields are kept in a document, which includes: ID, tweet text, language, city, date and time, sentiment score, numbers of keywords occurrence and SA2 code and name. A sample document is shown below:


```
{
  "_id": "1079892087171137536",
  "_rev": "1-4d2630b20cb1fcec43f4aecd86e2919c",
  "text": "Happy New Year! Wishing you all an incredible 2019 and hope
  all your dreams come true. I'm very grateful for my lif...
  https://t.co/9vQkUsA1Hx",
  "lang": "en",
  "location": "sydney",
  "year": "2019",
  "month": "Jan",
  "day": "01",
  "time": [
    "00",
    "08",
```

```
      "31"
    ],
    "zone": "+0000",
    "afinn": 12,
    "vulgar_words": 0,
    "crime_words": 0,
    "covid_words": 0,
    "alcohol_words": 0,
    "china_words": 0,
    "SA2_names": "Sydney - Haymarket - The Rocks",
    "SA2_codes_9_digits": "117031337",
    "SA2_codes_5_digits": "11337"
}
```


3. Deciding whether to upload or not. It is indeed a waste of storage to upload every refined tweet document. Instead, only those having coordinate information or having at least 1 occurrence of relevant keywords will be saved in CouchDB. For example, scenario 2 only requires geo-tagged tweets in Melbourne and scenario 1 only requires china-related tweets in Sydney.

## 4.2 CouchDB

The utilization of couchDB cluster makes the system feasible for big data storage and high computational performance. CouchDB is an open source database that stores data in JSON format and uses a document-oriented NoSQL database architecture. Unlike relational databases storing data and relationships in tables, couchDB provides a more flexible document model. While a document is the primary unit of data, each database of couchDB is a collection of independent documents.

CouchDB replication process makes the database fault tolerant. We start to harvest tweets on the single-node *localhost* and then initialize a three-node cluster later across three instances of Nectar cloud. CouchDB makes teamwork easier by the HTTP-based replication of data from a local database to a remote one. Within a cluster, data only needs to be uploaded to one of the nodes, say master node, as couchDB internally replicates and saves data in other nodes. Since a three-node clustered database can resist up to a two-node failure, a single point failure does not affect the system functionality such as data availability or storage capacity.

CouchDB built-in MapReduce functions are suited to Twitter analytical problems because rather than a single tweet, the aggregation result of tweets reflects more insight/trend. To support our research scenarios, several MapReduce functions are created with customized reduce functions.

### 4.2.1 Four Databases on CouchDB

To boost the view calculation efficiency in deployment, all tweet data are separated into 4 databases based on the usage for different scenarios:

1. *sydney-relative*: All tweets containing at least one occurrence of any keyword collected between 2019 and 2020 in Sydney. This database incorporates in total 2887199 tweets and will be used in Scenario 1.

2. *sydney-geo*: All geo-tagged tweets collected between 2019 and 2020 in Sydney. This database incorporates in total 104977 tweets and will be used in Scenario 1.

3. *melbourne-geo*: All geo-tagged tweets collected since 2014 in Melbourne. This database incorporates in total 622872 tweets and will be used in Scenario 2.

4. *all*: All tweets collected over the past 30 days (2021 Mid-April - 2021 Mid-May) in Sydney, Melbourne, Brisbane, Adelaide, Perth and Canberra. This database includes in total 288860 tweets and will be used in Scenario 3.

### 4.2.2 MapReduce Views

Generally, we have 2 types of views, say, **type A** and **type B**. Type A aims to calculate the average or total sentiment score of tweets containing specified keywords or filtered by other conditions (e.g., average / total sentiment score for all non-English tweets). Type B aims to calculate the proportion of tweets containing specified keywords or filtered by other conditions to the total tweets (e.g., the proportion of non-English tweets to total tweets). The two types of views can be aggregated at the level of different SA2 area, city or time period by adding the `?group=true` to the end of each view's URL, which depends on the scenario where it is used.

The *Map* function part of the two types will emit different kinds of values. Type A maps a document to its sentiment score only if the tweet contains corresponding keywords (or satisfies other conditions). Type B maps a document to 1 if the tweet contains corresponding keywords (or satisfies other conditions) otherwise to 0. The keys emitted in the *Map* function are based on the scenario and aggregation requirement.

The *Reduce* function part of the two types are essentially the same in terms of coding structure. Both types will calculate the summation, mean and number of counts of the emitted values, which is output as a JavaScript object with keys in the form of `{sum, avg, count}`. However, the interpretations are different. For type A, `sum` and `avg` represent the total and average sentiment score respectively. `count` represents the number of tweets containing relative keywords. For type B, `sum` represents the number of tweets containing relative keywords. `avg` represents the proportion of tweets containing specified keywords to the total tweets. `count` represents the number of total tweets at a certain aggregation level.

The views used in each scenario are summarized in the table below.

| View Name | Database | Scenario | Type | Filter | Aggregation Level |
|---|---|---|---|---|---|
| `china_stats` | *sydney-geo* | 1 | A | Contains China-related keywords | SA2 Area |
| `china_stats` | *sydney-relative* | 1 | A | Contains China-related keywords | Month |
| `chin_covid_stats` | *sydney-relative* | 1 | A | Contains China-related and COVID-related keywords | Month |
| `china_vulgar_stats` | *sydney-relative* | 1 | A | Contains China-related and vulgar-related keywords | Month |
| `bad_words_stats` | *melbourne-geo* | 2 | B | Contains alcohol-related, crime-related and vulgar-related keywords | SA2 Area |
| `afinn_stats` | *melbourne-geo* | 2 | A | None | SA2 Area |
| `nonenglish_stats` | *all* | 3 | B | Text language is not English | City |
| `afinn\_stats` | *all* | 3 | A | None | City |

# 5 System Design and Architecture

An overview of our cloud architecture is demonstrated in the *figure 1* below. We deployed a total of four servers, including *instance0*, *instance1* and *instance2* as the carrier of data storage and web application. As a proxy service, *instance3* reversely proxy the user's request. Users can directly access the web application through port 80 of *instance3* to obtain the data in the cluster, which achieves the effect of load balancing to a certain extent.

The whole system has two layers of fault tolerance. At the database level, we set CouchDB on different nodes based on docker and cluster, which can be scaled up or reduced dynamically. Thus we ensure the reliability of data to a certain extent. When building a cluster, we were going to use *Docker Swarm* to build multiple services. However, considering the small scale of the project and the limited number of servers, we finally decided to adopt the cluster construction method that comes with the cluster. In addition, at the web application level, *Nginx* could be configured with multiple up streams. For example, if we have five instances running web services, the main reverse proxy server can have five different options to distribute clients' requests, and I choose the rule of IP hash. Still, to make sure the upstream is live, a health check module must be configured to poll status of all instances so that no request will be passed to a blocking instance. Generally if we have enough service providers, We should build a cluster of *Nginx*, which is a more secure solution.

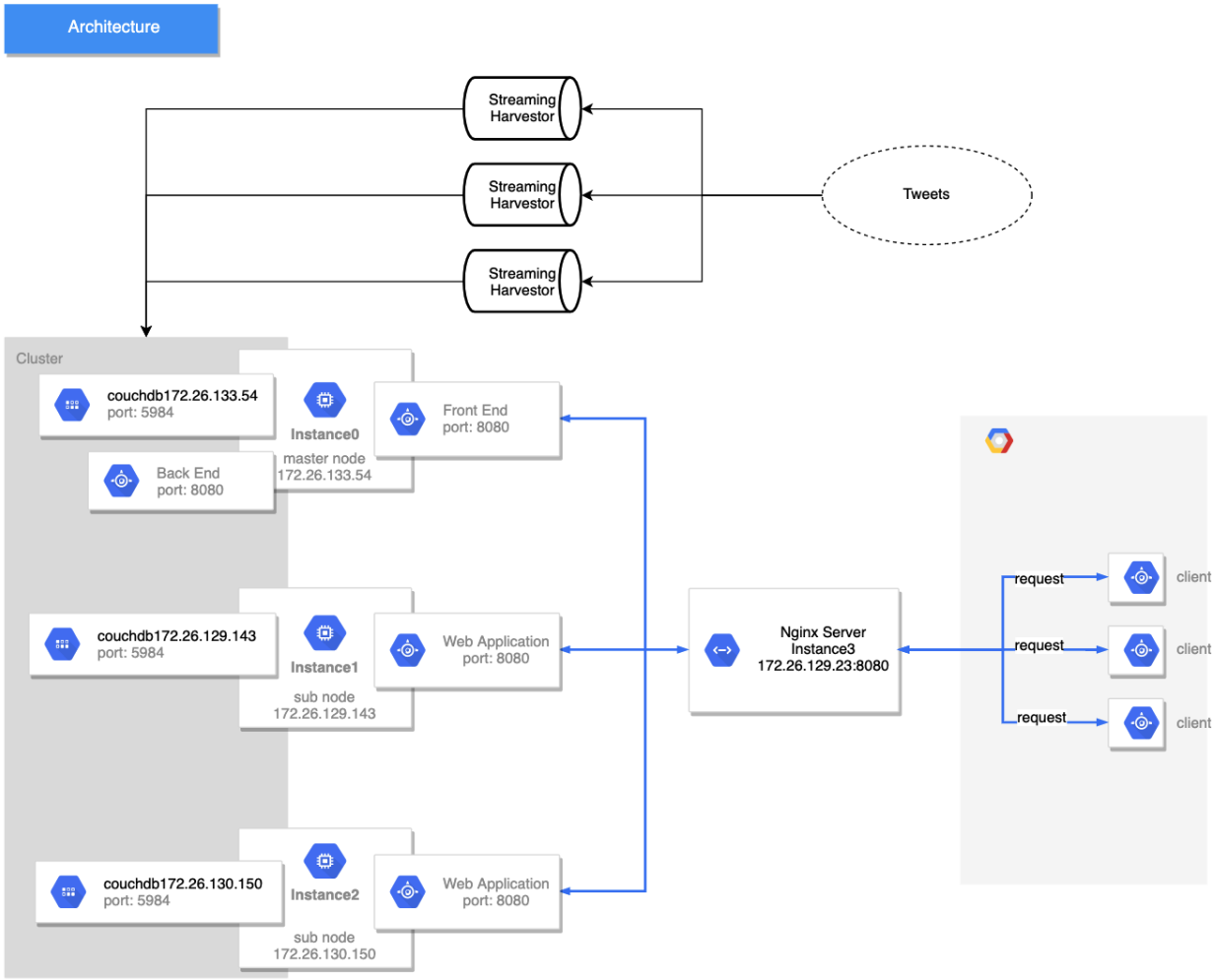


Figure 1: System Architecture

# 6 Melbourne Research Cloud

## 6.1 Pros

Some advantages of utilizing MRC are listed here:

1. Melbourne Research Cloud provides a scalable and configurable analysis platform for students and researchers of the University of Melbourne to access cloud computing.
2. Computing infrastructure makes teamwork easier and more flexible by remotely cooperating with each other.
3. Storing, accessing and analyzing our data can be done at any time by managing our own project on the cloud.
4. Multiple VMs can be exploited for a system with available allocated resources of Melbourne Research Cloud.
5. Cloud security is guaranteed.

### 6.2 Cons

Some disadvantages of utilizing MRC are listed here:

1. If any VM goes down, the team is responsible for backing up the MRC instances and data stored on MRC storage.
2. Connection is not stable when using VPN.
3. Compared with the traditional virtual machine, the user experience of MRC is not ideal in some ways. On the one hand, the technical documents are not perfect. On the other hand, it is difficult to repair the fault. If there is a problem with a component in the controller, locating and solving the problem requires a considerable technical level, and engineers need to be familiar with the corresponding knowledge field.

## 7 User Guide

### 7.1 System Deployment

When initializing the server, we use Ansible for automatic deployment. Ansible is a management tool which provides the ways of creating scripts for automating the deployment and configuration of remote virtual machines. One of the main advantages of using Ansible is definitely its simplicity. Given the increasing complexity of a management script, all this information is necessary in order to accomplish a coherent outcome.

The Ansible file includes different tasks to make a server ready to run our application. A brief description on the goals of several main modules:

- **Variables**: important parameters of four virtual nodes are configured.
- **Openstack security groups**: specific security groups are configured, and the most important function is to ensure that the port is opened or closed.
- **Openstack instance**: execute the deployment command.
- **Openstack images**: gets the specified image.
- **Openstack common**: install the necessary software on different nodes.
- **Docker configuration**: configure dockers on nodes and pull the required images.
- **CouchDB masternode**: create CouchDB container on instance0 and initialize a cluster.
- **Couchdb-subnode1**: create a CouchDB container on instance1 and add it to the cluster.
- **Couchdb-subnode2**: create a CouchDB container and cluster on instance2 and add it to the cluster.

To deploy automatically, it can be simply achieved by getting permission and running `/ run.sh.` directly on the command line. Some trouble shootings during deployment are summarized in *README.md* in the *Ansible* folder of our project GitHub repository.

## 7.2 Data Upload

It should be noted that all necessary pre-processed twitter data is saved beforehand in the form of JSON files, which means there is no need to setup our Twitter harvester after deployment. This is reasonable because our scenarios are based on historical twitter data and no real-time system is required. All documents in each of the 4 databases are dumped into a single JSON file separately and uploaded to one of our MRC servers. Note that the design document corresponding to its database is already included in each JSON file, thus we do not need to create the MapReduce function again in deployment.

When deploying our project, we only need to bulk upload each pre-processed data file to our master CouchDB node via CouchDB's `_bulk_docs` API. One challenge in this phrase is that `_bulk_docs` will take a long time or even be rejected by HTTP when the file is too large (e.g., more than 100MB). The reason is that `_bulk_docs` need to parse the whole raw text to a JavaScript object first before upload. Parsing time will exponentially increase as the size of the file increases and will eventually occupy all available memory. One solution to address this issue is to split each large JSON file into several smaller files and bulk load each file one by one. In our cases, each smaller file is arbitrarily configured to contain 10000 documents, which will take up to 5MB in hard disk space. As a result, it generates 392 small JSON files which takes approximately 2.2GB in hard disk space. Then, the whole bulk uploading process is accomplished by running a shell script, which will take roughly on average 1 hour in total, which is varied by MRC network condition.

## 7.3 Back-end Web Framework

*Flask* is used as the back-end web framework for this project. It provides simplicity, flexibility and fine-grained control, which is suitable for our lightweight development. In order to be more portable and resource-efficient, the *flask* back-end is dockerized and distributed on the instance 0, 1 and 2 to achieve access load balancing. The main purpose of back-end is to access different views created on the CouchDB databases, and combine the aggregated statistics with the AURIN data to return the specific data that needs to be visualized on the front-end web.

## 7.4 Front-end Application

The front-end of the system is a single page application powered by the *Vue.js* framework with the assistance of *npm* and *vue-cli3*. It is easy to build the application in a few seconds. To interact with the back-end, we use AJAX to make the request. The user interface shows clarity in its

design. Users can switch between different scenarios and charts by clicking on the left navigation bar. Each chart is properly labeled with a title, an optional subtitle and labels of horizontal axis and vertical axis. To quickly identify the data series in a chart, data labels are added for each data point of the chart. Users can hover the point on the chart if they want to view the specific value of a data point. Also, by clicking on the legend buttons under the chart, users can select data they would like to display or choose displaying all the data. Considering the API request will take several seconds, we have added the loading animation to each page so as to tell users to wait for the loading of data, which makes our application more user-friendly.

# 8 Data visualisation and Analysis

## 8.1 Scenario 1

Based on historical Twitter data in Sydney, the outbreak and rapid spread of COVID-19 do affect the relationship between Australian and Chinese people in a negative way. Chart A and B's overall curve of China-related and China-Coronavirus-related tweets is decreasing since the commence of the pandemic, where these two curves are sharply falling to the lowest points respectively in February and May of 2020.

From charts A, B and D, as people living in Australia expressed more opinions about China on social media, their opinions tend to be more negative compared with those before the era of Coronavirus. Chart C's monthly new confirmed cases of COVID-19 assists us to recount the pandemic situation in Australia, In fact, it is difficult to distinguish people's negative views on all aspects or just about China at that time, because during the most severe period of COVID-19 last year, we may all have experienced anxiety, anger and other negative emotions.

Other than the previous time-series analysis, the functionality of our system enables correlation analysis between Twitter data and AURIN data.The database of *sydney-geo* is the historical Twitter data which all contains coordinates and enables the analysis to a refined level of SA2 suburbs. Mean income of individuals from 2016 Census data works as the economic indicator, which shows a truly weak positive correlation with average sentiment score of tweets containing keywords related to China from Chart G. An uncorrelated pattern is observed in Chart H, as hypothetically we believe there would be a positive association between the two variables.

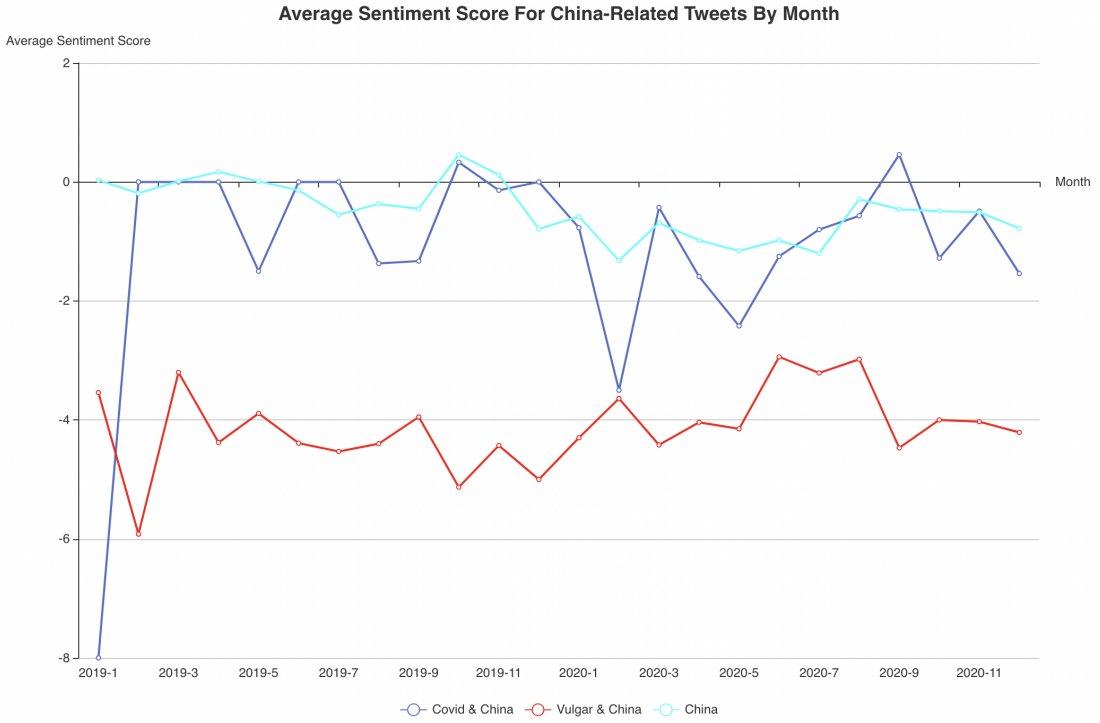


Figure 2: Scenario 1 Chart A

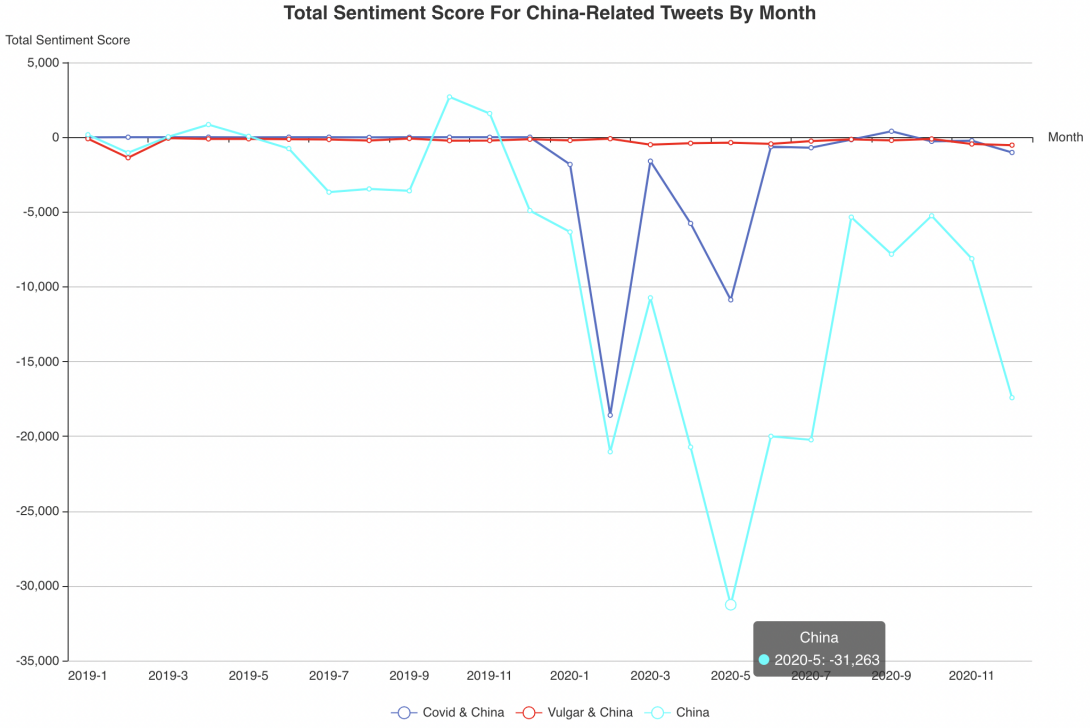


Figure 3: Scenario 1 Chart B

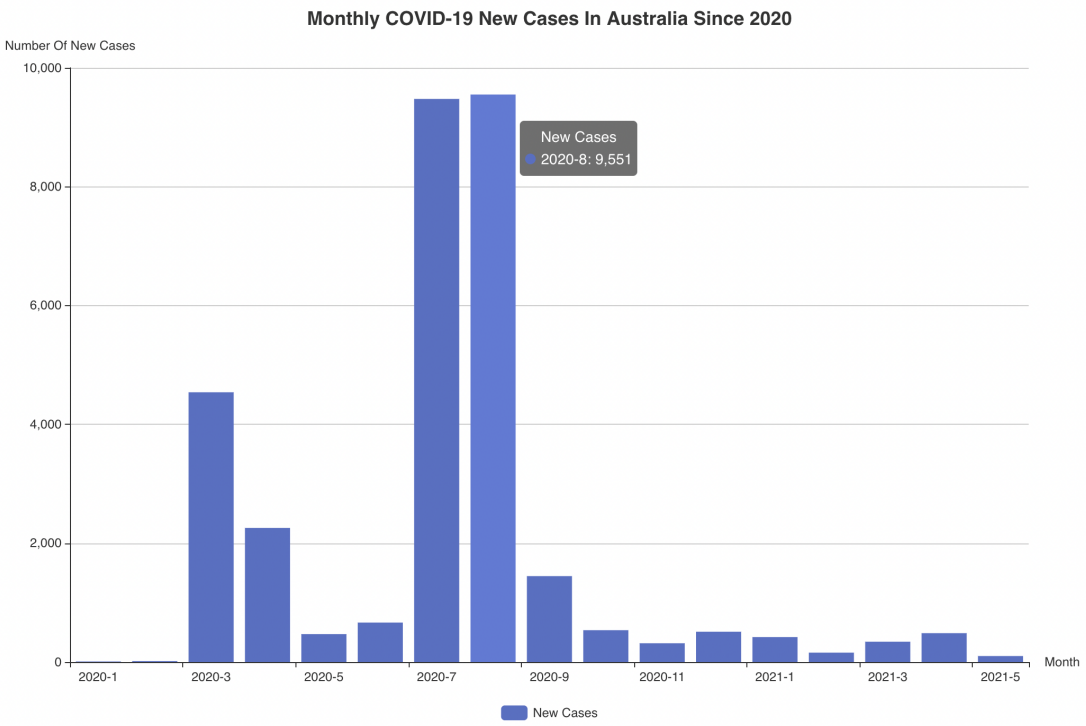


Figure 4: Scenario 1 Chart C

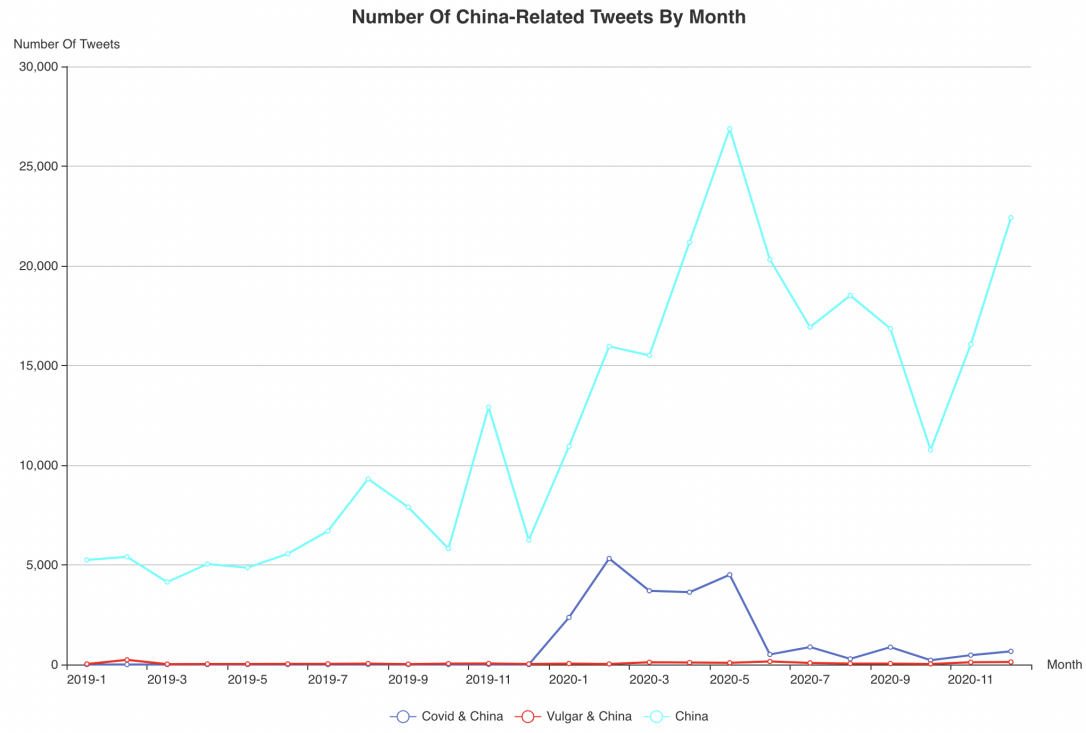


Figure 5: Scenario 1 Chart D

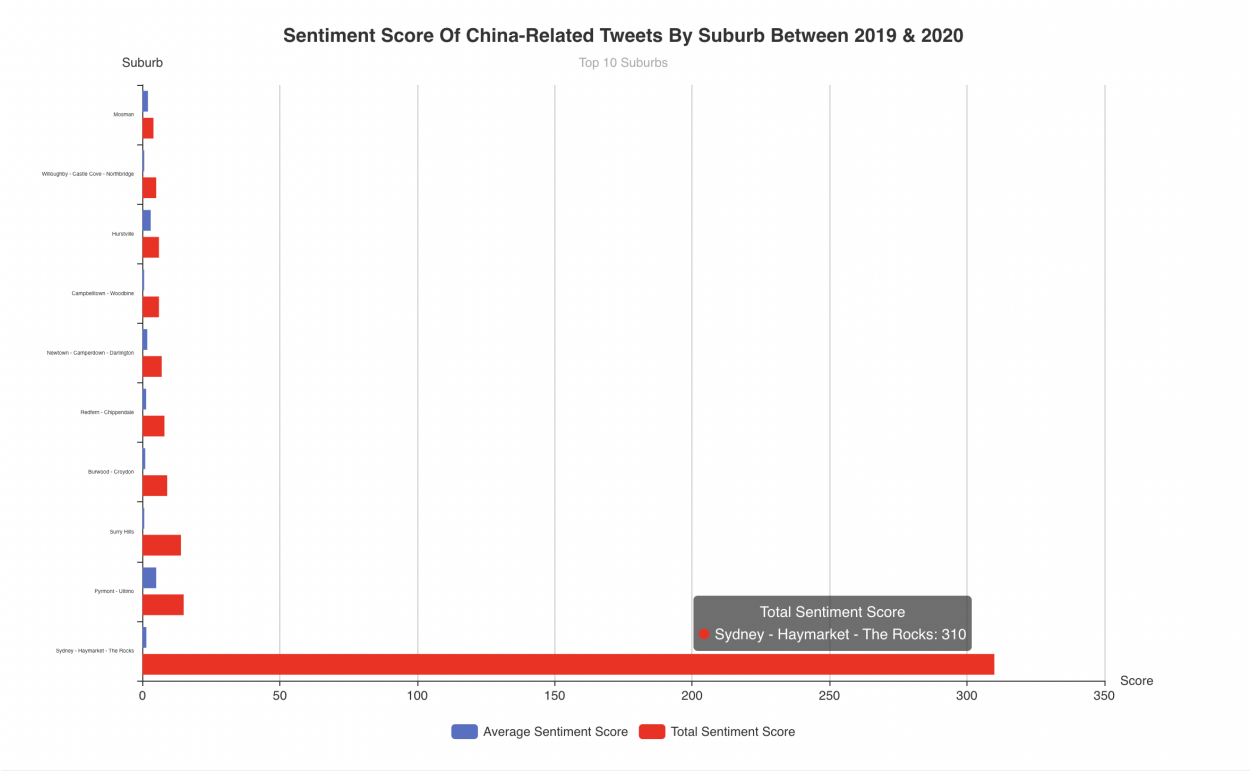


Figure 6: Scenario 1 Chart E

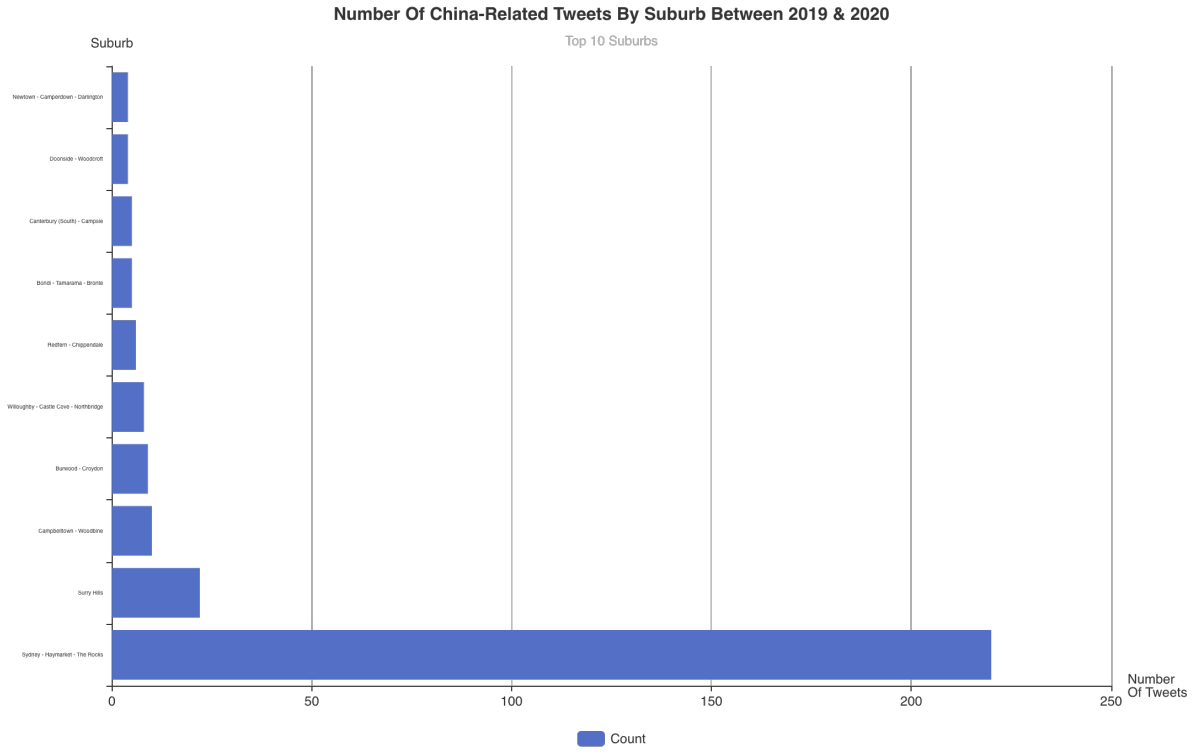


Figure 7: Scenario 1 Chart F

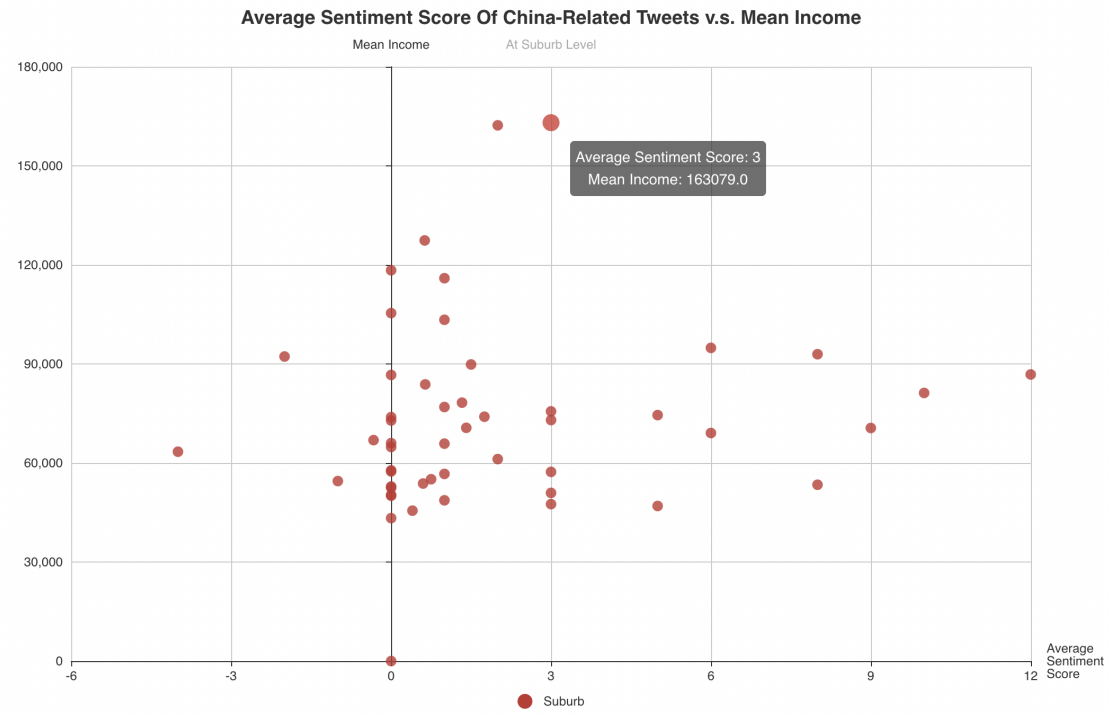


Figure 8: Scenario 1 Chart G

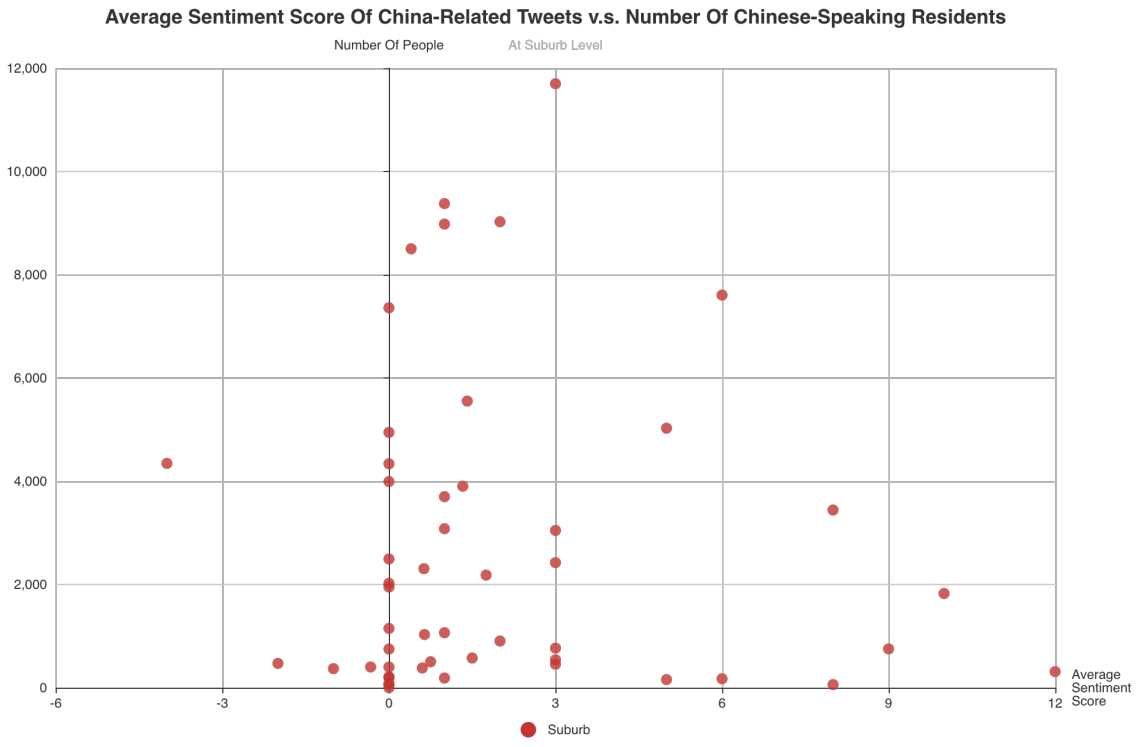


Figure 9: Scenario 1 Chart H

## 8.2 Scenario 2

Using vulgar, criminal or alcoholic keywords and the sentiment scores of related tweets as indicators, we have ranked the suburbs considered as the most unsafe ones in Melbourne based on

its historical Twitter data. However, the top ten suburbs with the highest proportion of related tweets do not align with those suburbs with the highest total number of related tweets from Chart A and B. Since no overlap of suburbs is detected when further including suburbs with the lowest average or total sentiment scores in Chart C, we conclude these suburbs all have potential to be the unsafe suburbs.

The suburbs with the lowest average sentiment scores are clustered in the bottom left corner of Chart D. Other outliers are located either on the horizontal axis, or on the right side of the cluster, being much further away from the cluster than other data points are. Excluding these outliers, we can observe a negative correlation between the proportion of vulgar, criminal or alcoholic keywords in tweets and their sentiment scores. A lower average sentiment score is observed with a higher proportion of tweets including keywords, but this result does not indicate a negative cause-and-effect relationship.

The demographic and economic characteristics used in Scenario 2 are gini-index and unemployment rate at suburb level. The former measures the income distribution of the population in each suburb. The higher ones indicate greater income inequalities, with the high-income individuals receiving much larger percentages of the total income of that suburb. Data points are clustered together and show no obvious linear patterns in Chart E and F.

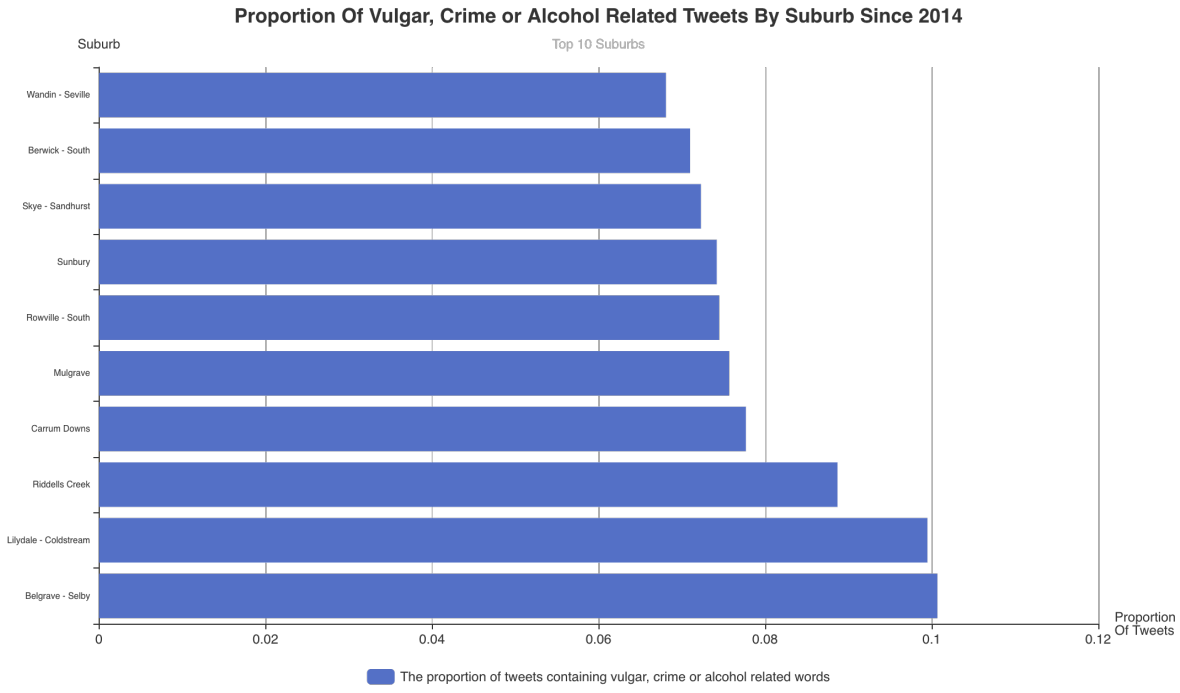


Figure 10: Scenario 2 Chart A

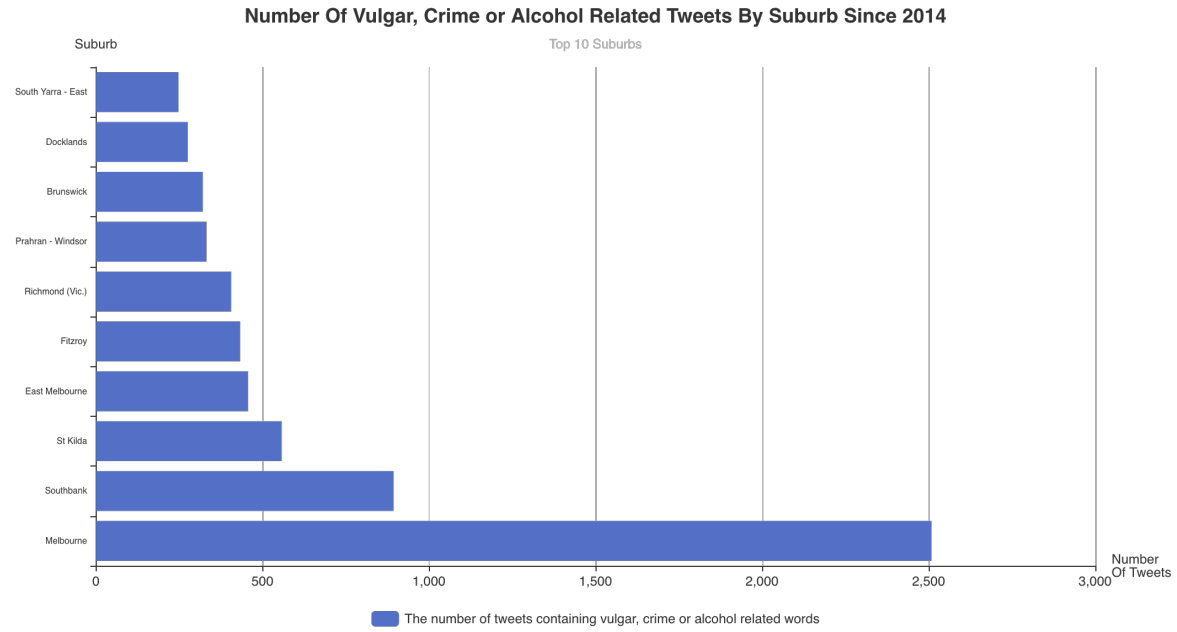


Figure 11: Scenario 2 Chart B

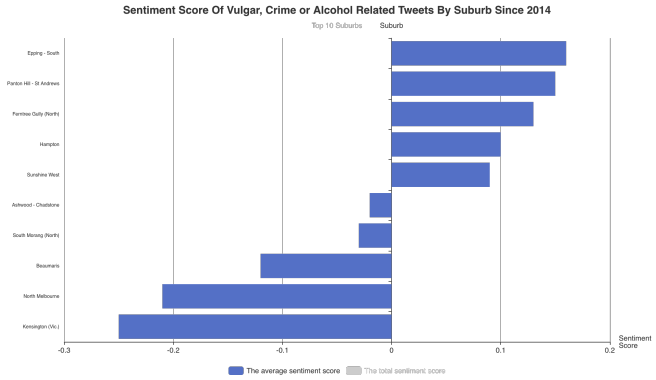


Figure 12: Scenario 2 Chart C-1

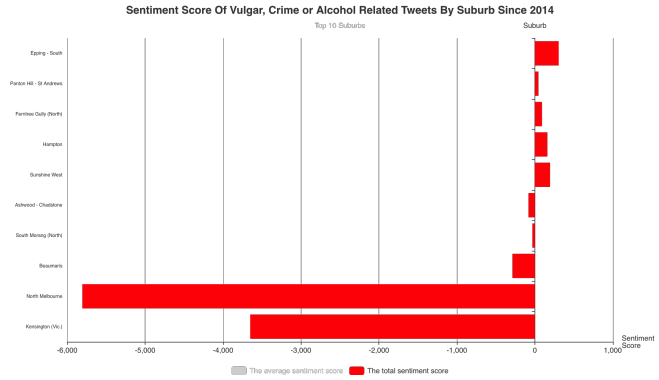


Figure 13: Scenario 2 Chart C-2

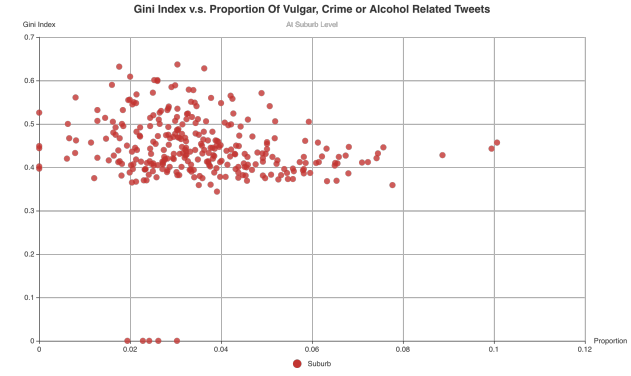


Figure 14: Scenario 2 Chart E

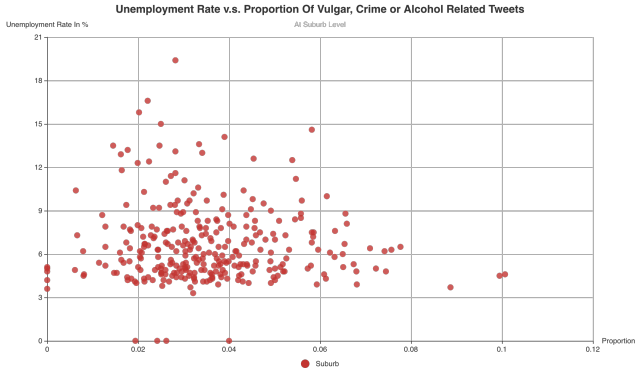


Figure 15: Scenario 2 Chart F

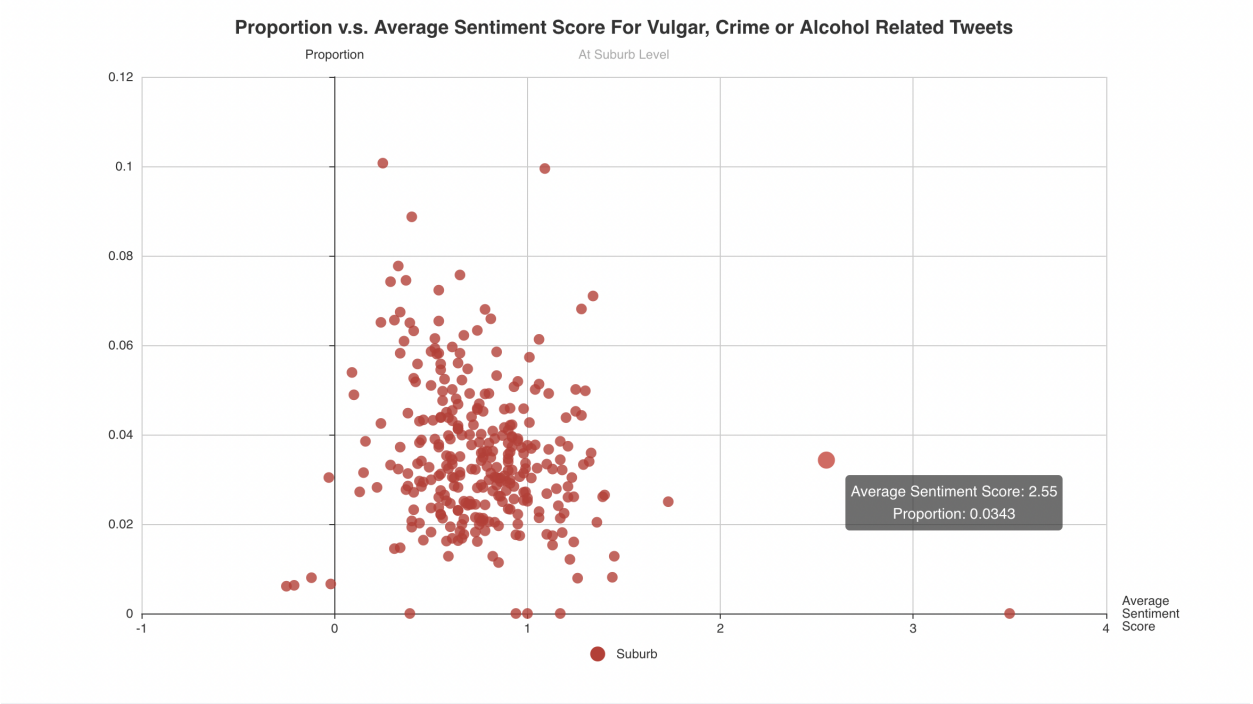


Figure 16: Scenario 2 Chart D

## 8.3 Scenario 3

Based on the Twitter data collected in six major cities of Australia since last month, we infer that Sydney has the most non-English speakers and possibly the most immigrants, and it is also the most favorable city for immigrants. The difference in the percentages of non-English tweets among these cities is not too wide, ranging from 7.07% to 12.82% in Chart A. Although English is still the main language of Australia, the multi-linguistic environment makes any of these cities a favorable choice of immigrants when they first come to the country.

In chart D, AURIN's demographic data is the proportion of Australian residents who do not speak English at home in each city, which shows a positive correlation with the percentage of non-English tweets. The proportions in Sydney and Melbourne (at 35.8% and 32.3%) are much larger than those (averaging at 19.43%) in the four other cities, and this result narrows down to the two former cities to become the most favorable city.

When people decide to migrate to a new city, another important factor is the city's vitality and growth of economic activities. Using the total business number as the indicator in Chart C, we show another strong positive correlation between the total business number and the percentage of non-English tweets. This result shows that immigrants and people from non-English speaking countries prefer cities with better economies and more job opportunities, such as Sydney and Melbourne.

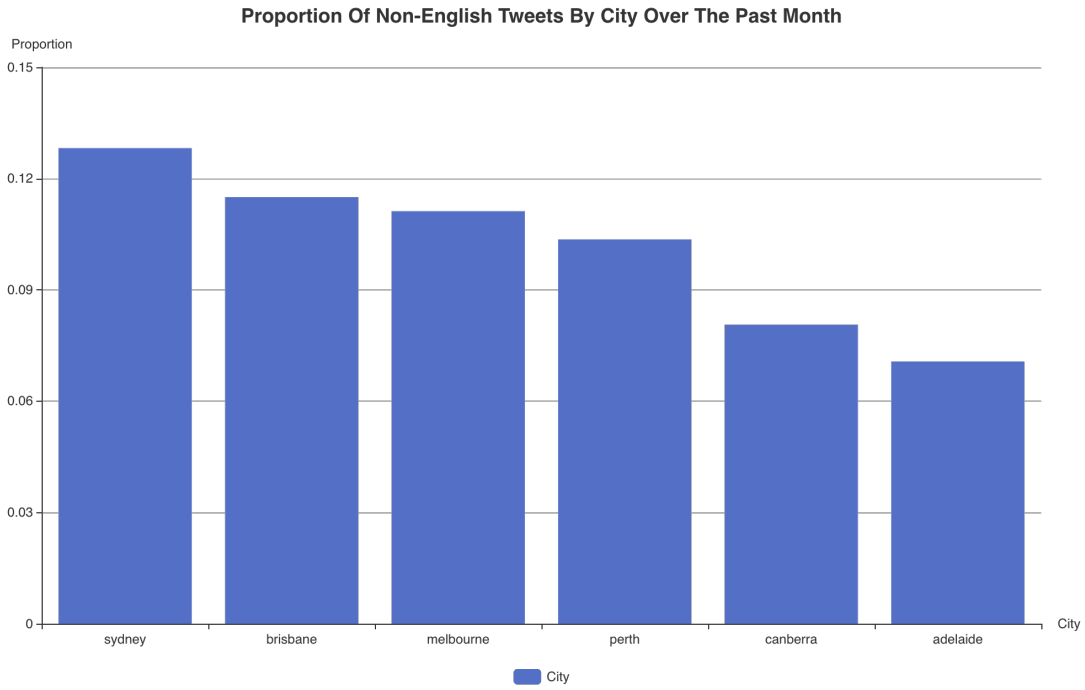


Figure 17: Scenario 3 Chart A

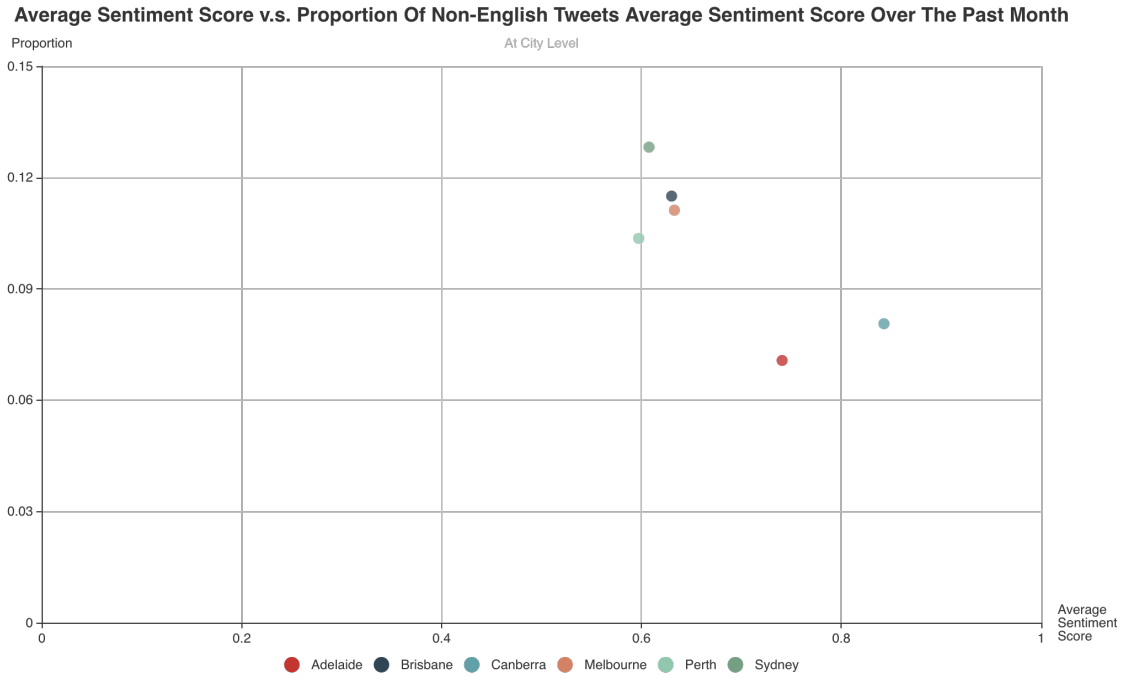


Figure 18: Scenario 3 Chart B

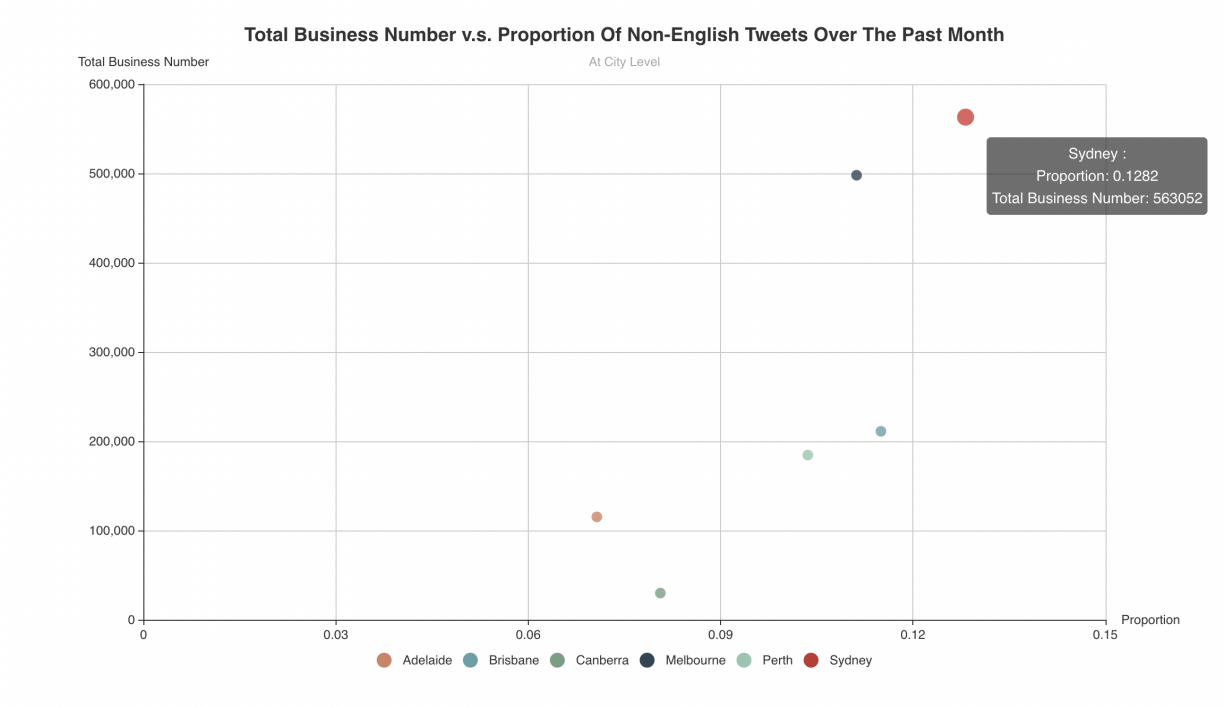


Figure 19: Scenario 3 Chart C

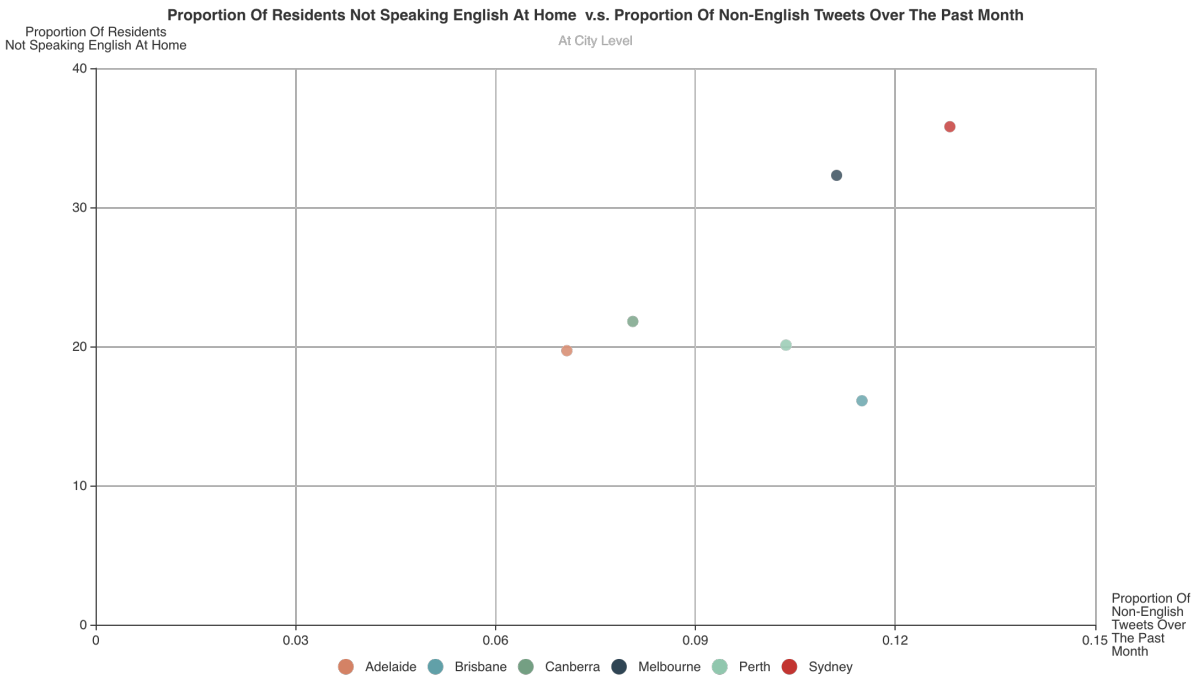


Figure 20: Scenario 3 Chart D

# 9 Conclusion

The cloud-based software system possesses the functionalities of harvesting Twitter data, pre-processing data to retain useful information and generate sentiment score, storing data in CouchDB, aggregating data by MapReduce views and visualizing data at the web application. The system also has scripted deployment capabilities using Ansible. The system is fault-tolerant at the database

level where the three-node CouchDB cluster is set up on different VMs and at the web application level where *Nginx* could be configured with multiple up-streams if more resources were available. The former point also makes the system dynamically scale up to meet demand.

# 10 Appendix

- Our GitHub repository: https://github.com/Mrsquuuid/cccTeam
- A demonstration video of our system has been uploaded on YouTube: jphFhttps
- The link to our app: http://172.26.129.23:8080/

# References


[1] Stuart E. Middleton, Giorgos Kordopatis-Zilos, Symeon Papadopoulos, and Yiannis Kompatsiaris. Location extraction from social media: Geoparsing, location disambiguation, and geotagging. *ACM Trans. Inf. Syst.*, 36(4), June 2018.

[2] Finn Årup Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*, 2011.