

# **COMP90024: Cluster and Cloud Computing**

## **Assignment 2: Exploring the Deadly sins**



## **Twitter analysis on Immigration in Australia: The Truth**

### **Team 27**

**Maria Rashmi (1040446)**

**Avijit Chauhan (1012820)**

**Harshit Sethi (990198)**

**Karan Bansal (1007845)**

**Srijan Choudhary (1017194)**

## Abstract

A human behaving excessively or abusing its natural desires or habits is coined as sins. The purpose of the assignment is to explore one or more deadly sins, which are pride, greed, lust, envy, lust, gluttony, wrath and sloth and analyse these sins through social media data analysis. The sins analysed for our project and story are pride, greed, envy and wrath. We try to present these sins by building our story around immigration status in Australian cities. The movement of people into a non-native destination country with the purpose of employment and/or settlement is termed as immigration. In 2018, Australian Bureau of Statistics clocked a population of 25 million, with immigration constituting 62% in the last decade, which was 33 years ahead of the schedule [2].

Foreign residents can move to Australia to attain better employment, which can be associated to greed as Australia's employment rate increases to 0.8% annual with immigration as compared to 0.2% without immigration. Whereas due to the increasing immigration population, issues such as traffic, overpopulation and effects on housing prices can generate hatred amongst Australian natives due to lack of jobs, road congestion and affordability of houses, which can result in wrath and envy against the immigrants. Australia's unemployment rate rose from 4.5% to 5.6% in 2017, housing pricing boosted by 412%.

These aspects will be analysed by building a cloud-based solution on UniMelb's NeCTAR Research Cloud which harvests data from twitter and analysis is done between the sins and immigration status in Australian cities - Adelaide, Brisbane, Canberra, Melbourne, Perth and Sydney.

The story for the assignment is divided into scenarios under which we analyse the sins pertaining to the scenario. The scenarios are:

- Is immigration an issue in Australia and are overseas population travelling to Australia due to pride and/or greed?
- Is there wrath and/or envy among the Australian residents towards the immigrant population due to effects of immigration such as traffic, housing affordability and jobs.
- What is the general attitude of public, media and/or politicians towards immigration?

Scenarios are segregated into one of the following - data (out of the gathered data what amount is constituted towards the scenario), trend (out of the constituted data what is the nature and type of the tweet - positive or negative) and profile (politicians, media and/or public mentioning these scenarios).

## Table of Contents

1. Introduction.....	1
2. Architecture and Diagrams.....	2
3. System Components.....	4
3.1 Data Collection using Crawler.....	4
3.2. Front-End Web service.....	6
3.3 Error Handling.....	8
3.3.1 Removing Duplicate tweets.....	8
3.3.2 Fault Tolerance.....	8
4. Deployment and Set-up.....	9
4.1 NeCTAR Research Cloud.....	9
4.2 Ansible.....	10
4.3 CouchDB.....	13
5. Processing.....	15
5.1 Pre-processing.....	15
5.2 Sentimental Analysis.....	18
5.3 Data Analysis.....	20
6. Conclusion.....	29
7. Repository and Communication.....	30
8. References.....	31

# 1. Introduction

The report is submitted as part of COMP90024 Cluster and Cloud Computing Assignment 2. The aim of the assignment is to carry out social media analysis to explore the seven deadly sins. The activities, in order to achieve the same, is executed on University of Melbourne cloud system - NeCTAR Research Cloud []. For social media analysis, Twitter is used as the social media platform, from which tweets are harvested in order to analyse and drive conclusion. Above 4 million Australian population use twitter, which is 19% of the population using social media [4]. The conclusion derived from twitter data on the sins was then cross checked from the official data, made available by AURIN [4].

The assignment consists of building a Cloud-based solution that exploits a host of virtual machines (VMs) across the NeCTAR Research Cloud for streaming (using Streaming API interfaces) and searching (using Search API interfaces) tweets through the Twitter APIs for six Australian cities (Adelaide, Brisbane, Canberra, Melbourne, Perth and Sydney). The harvester runs on multiple instance of the cloud platform and CouchDB database is associated to this which stores the collection of tweets from the harvester applications in JSON form. The tweet collection is analysed, manually and automated, and associated with the sins in order to present the overall findings and conclusion from the data. The pre-analysis of the data showed that people on twitter talking about immigration, generally associated synonyms such as 'seekers', 'foreigner', 'migrant', 'permanent residence', 'overpopulation', etc. which major tweets showing negative behaviour to immigration in Australia.

The harvesters collected 7,123,658 tweets from the six cities out of which 448,557 tweets were talking about immigration or related topics, which 40.2% users having a positive behaviour towards immigration. These findings coupled with python processing are presented using a front-end web application to visualise the scenarios and the analysis from data sets, which is implemented using Django framework. The visualisation of the front end presents the trend, content and profile analysis of the collected data using graphs and maps with evidence from AURIN data. Data from the database views is send across to the front-end service in JSON form, which on picking it models the data to dictionary form. This dictionary is consumed by google apis, which are utilised for visual representations of the data.

The following sections of the report talk about the implementation of the cloud-based application, with each section providing insight on how components (such as crawler, CouchDB and front-end services) and deployment mechanics (ansible) were implemented and used. Fourth section of the report presents the discussion on how data processing and analytics

were carried out in order to explore the sins and relate them to the story. Also, the section discusses the issues faced while implementing that particular part.

## 2. Architecture and Diagrams

### 2.1 Architecture

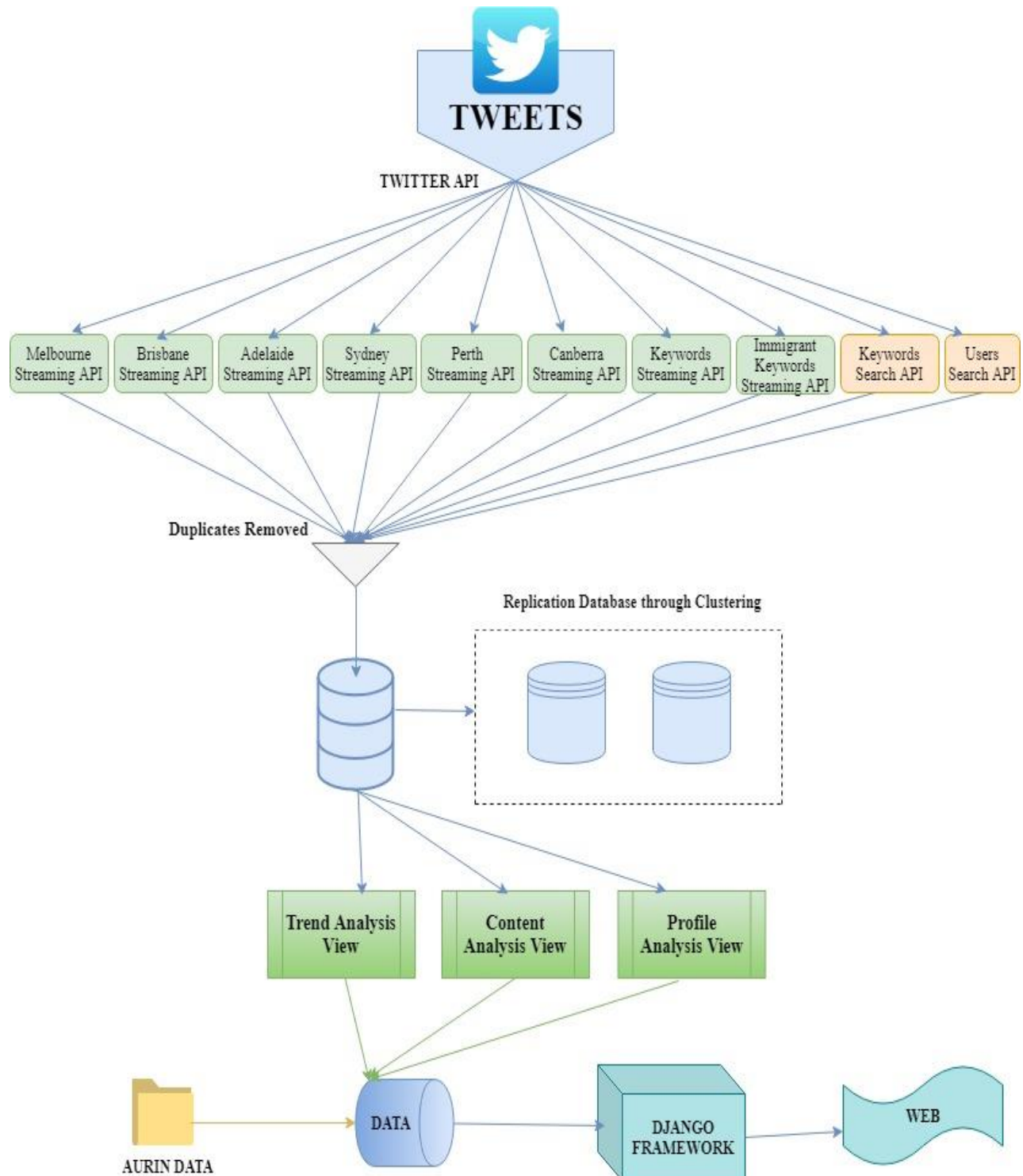


Figure: High level system Architecture

## 2.2 Diagrams

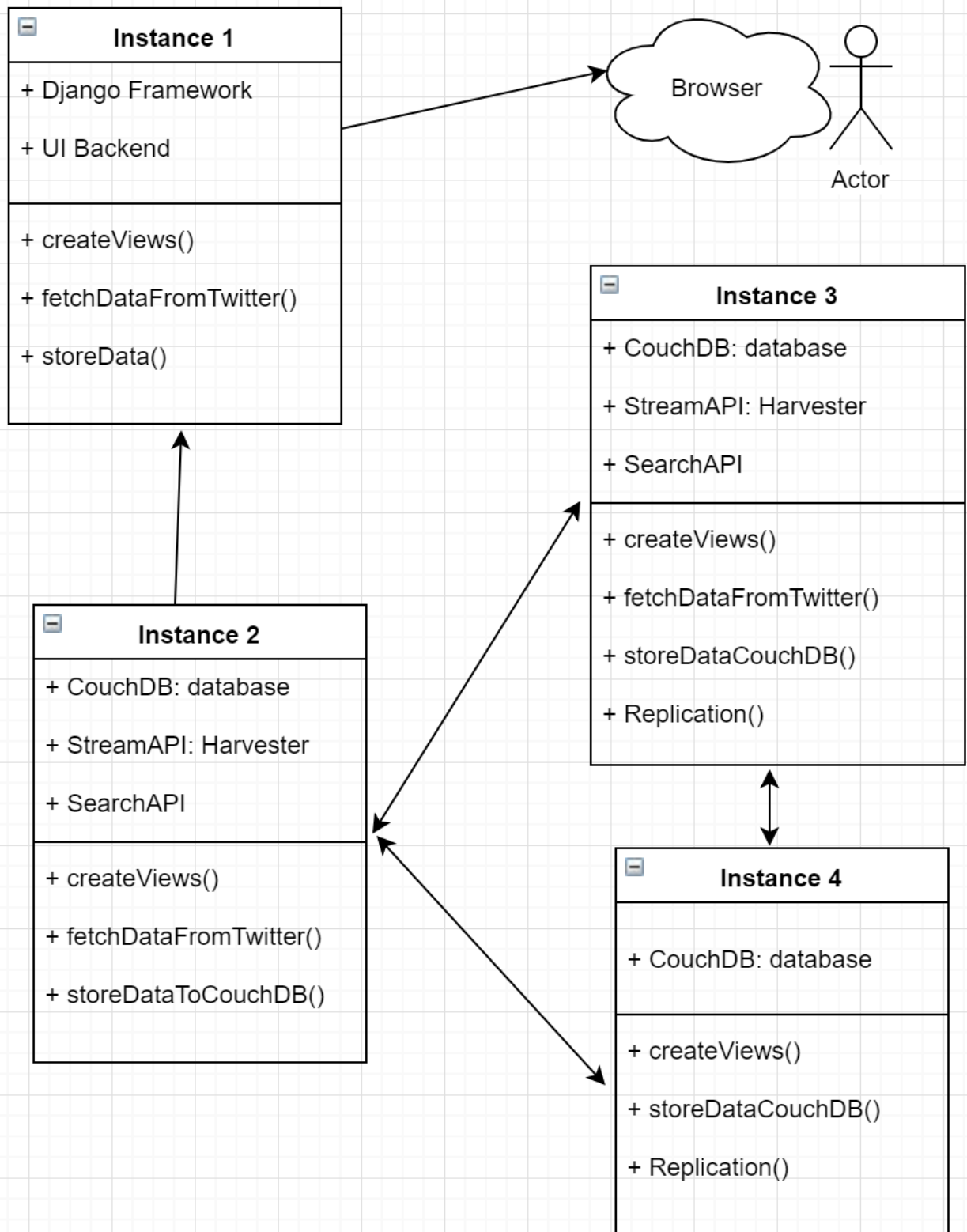


Figure: High level Class Diagram

### 3. System Components

#### 3.1 Data Collection using Crawler

For the purpose of performing upright data analytics for social media platform sufficient data was required. Twitter was used as the social media platform from where data was gathered in order to perform data analysis and base our scenarios. Twitter's Stream and Search API were consumed to construct the harvester in order to crawl data. Python's **tweepy** library was utilized to implement the harvesters. All data was gathered in JSON format.

Data was gathered city wise for the six cities included in this project - Adelaide, Brisbane, Canberra, Melbourne, Perth and Sydney. Box locations for the cities were used to collect data only for the specific locations. A total of 7,123,658 tweets were collected across the cities.

Box-locations city wise [5]:

Adelaide = [138.4421299, -35.3489699, 138.9634089955, -34.652564]

Brisbane = [152.6685, -27.7674, 153.5398, -26.9968]

Canberra = [148.7828, -35.9149, 149.4063, -35.1213]

Melbourne = [144.5937, -38.5047, 145.6299, -37.5113]

Perth = [115.5550415094, -32.7885221785, 116.0567258875, -31.4781991154]

Sydney = [150.2431, -34.4403, 151.5079, -33.3347]

Also, UniMelb Research Cloud platform provided data was gathered for the six cities. The data was accessed using the below command and saved into CouchDB:

```
curl
"http://45.113.232.90/couchdbro/twitter/_design/twitter/_view/
summary" -G --data-urlencode 'start_key=["perth",2016,1,1]'
--data-urlencode 'end_key=["perth",2016,12,31]' --data-
urlencode 'reduce=false' --data-urlencode 'include_docs=true'
--user "readonly:ween7ighai9gahR6" -o /tmp/twitter.json
```

The command required us to provide the city location and the start year to end year for which the data was to be collected. As in the above example command Perth was supplied as the location and 2016 as the year. This is repeated for all the six cities.

### 3.1.1 Why Tweepy?

As Python was decided as the initial language to implement twitter harvester, using already in-use libraries [1] or tweepy was the question. Tweepy was consumed since it provided the bounding box concept which allowed us to gather data specific to the location. Further, Tweepy allowed to filter tweet streaming by providing specific keywords/hashtags and/or user accounts. Also, tweepy allowed us to consume certain in-built functionalities such as error handling which is not provided by twitter-API implicitly. Addition to above, tweepy makes it easier to handle twitter API's authentication, creating and destroying session and reading incoming message, whereas other platform such as Twitter API itself requires additional work such as implementing Rest API which in order to pull the tweets would require further work of handling authentication and reading message.

Additionally, tweepy allows us to use **async** attribute which facilitates the work to be continued on another thread in case the working thread encounters any issue. This attribute comes into picture when we apply filter on tweepy, which in our case was filtering on basis of location grid.

### 3.1.2 Streaming API

Tweepy's stream function was used to gather data from twitter API. This allowed us to collect tweet in real-time. The harvester required us to provide authentication for user account of a twitter developer account. All the members of the group applied for a twitter developer account (<https://developer.twitter.com/en/apply-for-access.html>) and the authentication keys obtained by the same were applied to harvest data from different cities. The streaming harvester was utilised using two methods. Firstly, by filtering the tweets using bound box grid for the cities and secondly filtering the tweets using keywords and hashtags for the specified area.

The rate of tweets harvested by streaming API was 85-100 tweets per hour depending on the time of the day. This rate was comparatively slow due to two reasons. Firstly, we were harvesting tweet in bound box areas, hence tweets associated to the specific area might be limited. Secondly, in case of filtering tweets using specific keywords, it was the case that people actually tweeting about our related topic were not that much in number.

### Problem faced and solution

Although, twitter API imposes a rate limitation after which the user is blocked for a certain time period, generally 15 minutes, from harvesting tweets. This forces the harvester to stop, to overcome this the python harvester code was modified to listen of twitter's error code 420,



upon which the harvester was allowed to sleep for 15 minutes and start up again rather than die.

Another issue with streaming API was that large amount of data not being handled in program, we were getting *IncompleteRead* error which generally occurs when rate of incoming tweets and the rate at which the crawler is handling falls behind. This occurred since, twitter API was sending huge number of tweets for the list of keywords we were tracking. Due to this our crawler was lagging as twitter API stopped sending tweets after the error and the crawler interpreted that no more data is available and died. To overcome this the streaming crawler was modified to handle *IncompleteRead* exception, upon which the crawler would sleep for a certain amount of time and start connection with the twitter API again to stream tweets.

Though this work around resulted in loss of tweets, but it was simpler to implement it since the other work around required to utilise other thread to handle tweets, which was quite complicated. Also, the loss of the tweet won't harm the output with respect to the total amount of data collected.

### **3.1.3 Search API**

Streaming API only provided real-time tweets and the data we were collecting was not adequate for our scenarios, hence in order to back-track in time to collect old dated tweets search API was consumed. Search API allowed to back-track seven days to collect the data. Although the limitations for using Search API was that it only allowed us to search tweets from past using either using specific keywords in the tweet or user account. Thus, a manual analysis of the tweets gathered from streaming API was done in order to extract relevant keywords and hashtags to the scenarios. Keywords such as “immigration”, “jobs”, “traffic” etc. and user accounts who extensively tweeted regarding these keywords were picked.

Two separate search harvesters were run to collect seven days prior data, one being based on keywords and another based on user accounts. This data was saved onto CouchDB into two different schemas for keywords and users respectively. This was done to perform content analysis and profile-based analysis respectively, which is discussed in detail in section 4.

## **3.2 Front-End Web service**

Django framework was consumed in order to present our front-end application. It's a python based open-source framework based on Model, View and Control (MVC). Django allowed rapid development due to MVC. This was helpful since we were depicting data dynamically

from the cloud database. Hence, python scripts could be integrated on Django framework allowing better handling of data. So, controller could be utilised to fetch data from model as per view's requirements. Apart from rapid development feature of Django, scalability, maintainability and clean design which allowed us to focus on assignment specific development, rather than spend time on already available components.

In Django, when the application was created, this generates a *view.py* file which was utilised to direct the pages and data as per the task. Python scripts were written on server end to send data as per data analysis requirements to the Django views. Thus, we create a script within the *view.py*, since it gets executed each time, which is used to get and direct the data to html pages. HTML pages were constructed to view the data visualization. Bootstrap and CSS were consumed to provide beautification to a certain extend.

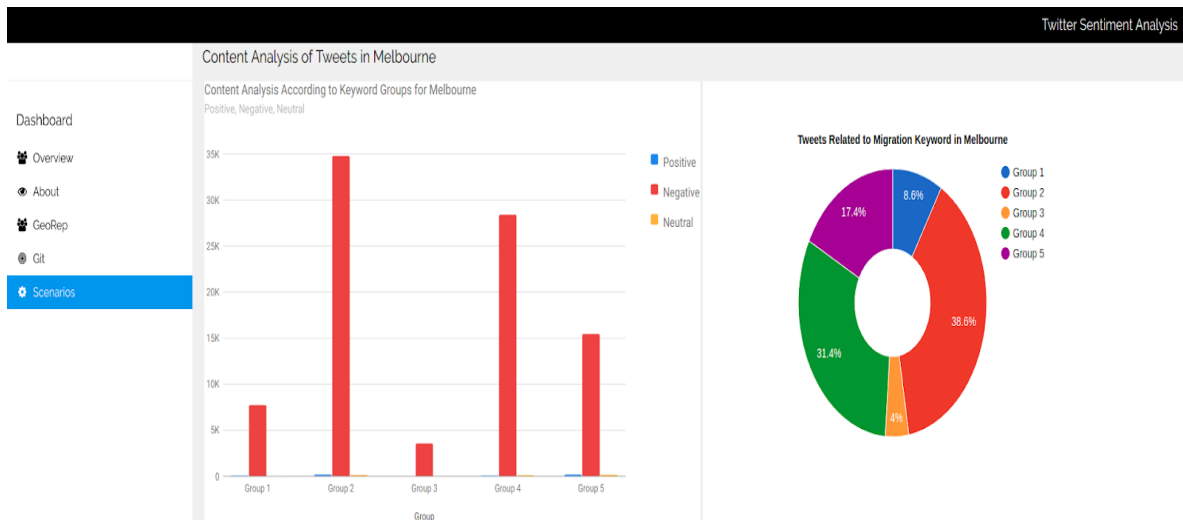


```
e.html views.py x trend.html about.html content.html georep.html profile.html
db_name = {'adelaide': ['immigrant_adelaide', 'all_keywords_adelaide', 'immigrant_adelaide', 'profile_analysis', 'trend_analysis'], 'brisbane': ['immigrant_brisbane', 'all_keywords_brisbane', 'immigrant_brisbane', 'profile_analysis', 'trend_analysis'], 'canberra': ['immigrant_canberra', 'all_keywords_canberra', 'immigrant_canberra', 'profile_analysis', 'trend_analysis'], 'melbourne': ['immigrant_melbourne', 'all_keywords_melbourne', 'immigrant_melbourne', 'profile_analysis', 'trend_analysis'], 'perth': ['immigrant_perth', 'all_keywords_perth', 'immigrant_perth', 'profile_analysis', 'trend_analysis'], 'sydney': ['immigrant_sydney', 'all_keywords_sydney', 'immigrant_sydney', 'profile_analysis', 'trend_analysis']}

#City Wise Data
trend_count = {'adelaide': [0,0], 'brisbane': [0,0], 'canberra': [0,0], 'melbourne': [0,0], 'perth': [0,0], 'sydney': [0,0]}
for dbname, design_name in db_name.items():
    db = couchserver[dbname]
    print(dbname)
    para = (design_name[0]+'/'+design_name[2])
    flag = True
    while flag:
        try:
            for item in db.view(para, group=True):
                trend_count[dbname][1] += item.value
            trend_count[dbname][0] = (db.info()['doc_count'])
            print(trend_count)
            flag = False
        except couchdb.http.ServerError:
            print("Retrying")
            time.sleep(3*60)
    trend_list['city'] = trend_count
```

*Figure: Python script on Django to get and model data*

The script within Django modelled the data received from database, which is in JSON form, to dictionary form. This dictionary is then directed to the HTML pages as per requirements. *Google.charts.api* and *google.maps.api* was utilised to show our data in a graphical representation form. Data specific to the scenarios is extracted for the Google apis, which interprets the dictionary and provides graphical visualization.



*Figure: Front-end Web Application*

### 3.3 Error Handling

#### 3.3.1 Removing duplicate tweets

It is possible that the crawler would receive duplicated tweets. One of the criteria was to not allow CouchDB to contain duplicate tweets, since it would consume unnecessary space and hamper the analysis. This approach would allow us to improve the speed of retrieval of data and ease in indexing while requesting views.

Hence, to handle this issue tweet id was mapped to CouchDB's "\_id" attribute, which is unique to the database. Thus, when a duplicate tweet is diverted towards the database, CouchDB can easily identify if the tweet is already present, with quite low time complexity.

#### 3.3.2 Fault Tolerance

**Crawler:** While using streaming API, twitter API imposes a rate limitation after which the user is blocked for a certain time period, generally 15 minutes, from harvesting tweets. This forces the harvester to stop, to overcome this the python harvester code was modified to listen of twitter's error code 420, upon which the harvester was allowed to sleep for 15 minutes and start up again rather than die.

**Database:** CouchDB was installed on two instances, one being the master and the other being slave database. Crawlers feed data onto the master database, which are then replicated onto slave using the below command:

```
curl -H 'Content-Type: application/json' -X POST
http://localhost:5984_replicate -d '{"source":
```

```
"http://<IP>:5984/<db_name>", "target": "<db_name>",  
"create_target": true, "continuous": true} '
```

Where target dbname is the address for the slave database. Continuous is set to true to ensure that replication of data is undertaken continuously from master to slave.

Data is basically store on the instance volume. This was done in case all of the instances don't respond, we still have our data intact. This is done by changing the director path of the database to volume.

The changes made to the path in local.ini file to store data on volume, local.ini is on path /opt/couchdb/etc for our servers:

```
[couchdb]  
database_dir = /path/to/the/databases  
view_index_dir = /path/to/the/views
```

## 4. Deployment and set-up

### 4.1 NeCTAR Research Cloud

NeCTAR (National eResearch Collaboration Tools and Resources) provides an online infrastructure to create our cloud-based application. It's a infrastructure as a service (IaaS), based on OpenStack which is open source for cloud technologies, which provides virtualized resources over the internet. For the purpose of this assignment we were provided with four medium sized virtual machines (VMs) with 8 virtual CPUs, each of 4 GB memory, totalling to 36 GB of memory. Also, a volume of 250 GB was allocated which was distributed among the instances as per requirements of the components on the instances.

The four instances provided were allocated to different components in order to perform specific tasks. CouchDB was installed in three instances, in cluster mode. This allowed for each CouchDB to behave equally and perform all the requests and tasks similarly. Hence, each instance database will have access to all the data at all time and can be up in case the acting database fails due to any reason. Thus, in case of any database instance failing to connect due to any issues, other instances will remain intact to handle the requests and provide similar data as before as they will access to all the data at any time. One instance was dedicated for using the front-end web service application, which contains the installation of Django framework. The harvester was deployed on two instances, one dedicated instance which crawled data through search API and UniMelb Research Cloud platform. The streaming API was deployed in the instance with slave database.

Each instance has volume allocated to it out of the total 250GB volume allocated. The instances containing CouchDB were allocated 70GB each, whereas another instance for front end is allocated 40GB each. Also, security groups were associated with the instances. These security groups contain list of rules that constitutes the protocols and port, which facilitates the inter-instance communication and receiving outside requests.

## 4.2 Ansible

Deployment of instances, creating and attaching volume and security groups, downloading dependencies and installing CouchDB, is automated through the use of Ansible. Ansible, a management tool, provides forms to constitute scripts to automate the above-mentioned deployment task. Ansible scripts also aid in remote machine configuration. Ansible playbook is used to describe roles that are to be executed on remote machines. The playbook is written in YAML format, which provides a simple syntax for use.

Ansible's main advantage is its simplicity, as it limits itself to minimum requirements. It doesn't require any centralized management servers, is easy to install (command: `pip install pip`) and converting bash scripts to Ansible scripts is made easy, due to its SSH approach, used to connect to target machine.

### 4.2.1 Implementation

The playbook consists of roles defined to create instances, volume, security groups, volume snapshot, attaching volumes to the created instances, adding proxies to the created instances, installing OpenStack images available for NeCTAR Cloud platform, installing common utilities such as pip and python onto the instances, installing and restarting CouchDB and deploying harvesters.

Apart from the roles defined, variable file was defined, under 'host\_vars' which contains the variables such as instance name, volume name and size and security group names, protocol and ports. The roles are called from the main YAML file (*multi\_nectar.yaml* in our case). Ansible allows sequential execution, hence this allowed us to order the role in order to avoid any conflicts and/or errors.

### Roles

All roles are defined in YAML file under *roles/<role-definition>/tasks*.

a. openstack-common

This role is used to install and update pip and install openstacksdk.

b. openstack-images

Lists all the openstack images available on Nectar Cloud

c. openstack-volume

This role is used to create volume on nectar. Its uses *host\_vars/multi\_necatr.yaml* variable file

to attain the variable values used in creating the volume.

d. openstack-security-group

This role is used to create security groups for the instances, listing the names of the security groups and defining rules for the groups. It also utilises *host\_vars/multi\_necatr.yaml* variable file

to attain the variable values.

e. create\_multiple\_instances

This role creates multiple instances on the NeCTAR cloud. Looping through servers using *with\_items* is done in order to attain creation of multiple instances. Upon creating each instance is attached with respective volume and security groups. Also, flavors, availability zones, key and network associated with the instances are configured. It also utilises *host\_vars/multi\_necatr.yaml* variable file to attain the variable values.

### **Problem faced and solution**

Initially creating multiple instance was done by utilising the snippet for single instance four times, in order to create four instances. Since there was redundancy in code and was an inefficient way to perform creating of instance in case more than four were to be created. Hence, looping using *with\_items* on the servers was utilised to create multiple instances.

The instances were distinguished into two different groups based on the work that is to be carried on those instances. Two instances each were created under *Appservers* and *Webservers*, named for convenience. *Webservers* instances are hosting streaming harvesting and front-end web services. Whereas, *Appservers* host CouchDB, both master and slave.

Grouping is done to utilise the group to install instance specific requirements. For example, it was decided to institute CouchDB on two instances, to constitute for replication, hence group

the instances allowed us to only install CouchDB on the two instances under *Appservers*. IP address for the instances was collected and grouped accordingly. Below is the snippet for grouping the instances using Ansible's *add\_host* - which uses variables to create new hosts and groups in inventory for later usage in the same playbook:

```
add_host:
  name: "{{ item.openstack.accessIPv4 }}"
  groups: "{{ item['item']['meta']['group'] }}"
  with_items: "{{ os_instance.results }}"
```

#### f. Create Volume Snapshots

This role was implemented to create volume snapshot - which allows us to store the backup of a particular time. So, in case of any unavoidable circumstances we can recover data from that timestamp from the volume snapshot.

#### g. Adding proxy on instances

This role is defined to add proxies to the newly created instances. Proxy are utilised to communicate the cloud servers with external network. This was done since initially we were not able to install libraries or dependencies on the Linux servers such as pip. Hence proxy were added to instance to communicate with external network which allowed us the install these dependencies.

#### h. Attach Volumes to instances

This role was utilised to define file system (*xfs -Extended File System*), install file system dependencies (*xfsprogs*), create directory and mount the device. It utilises *host\_vars/m\_vol.yaml* variable file to attain the variable values.

#### i. Installing CouchDB

This role is utilised to install CouchDB on two instances.

#### j. deploying harvesters on instances

This role is divided into three components. First component installs harvester related dependencies such as TextBlob, spacy and tweepy. The second component copies the harvest files from a source location to the instance location. Lastly, the third component will deploy the harvester on the instances and run the same.

During the utilization of roles i and j certain roadblocks occurred due to command misplacement or incorrect use of command. This was resolved by doing a in-depth analysis and figuring the issue. Blackboard discussions and CouchDB doc were utilised for help (<https://docs.couchdb.org/en/stable/intro/why.html>).

#### k. installing web dependencies

This role installs Django framework and CouchDB's python libraries. This also, runs the web server.

Apart from the above-mentioned roles, roles to install dependency packages, editing host and binding address were created.

### 4.3 CouchDB

CouchDB database was used to collect and store the collection of tweets from the crawlers. CouchDB was implemented in cluster mode, which allowed to attain more efficiency as all requests won't be targeted onto a single node. If ways other than cluster mode was implemented than buffer full due to multiple requests targeted to a single node could occur. Thus, in order to handle requests, in or out, cluster mode was implemented. Also, requests can be handled by other instances, for example slave instance, in case master instance die due to any reason.

Below are the four steps employed to install CouchDB on each instance, one by one

#### a. Creating local repository on instances:

```
echo "deb https://apache.bintray.com/couchdb-deb bionic main" \  
| sudo tee -a /etc/apt/sources.list
```

#### b. Adding installation key:

```
curl -L https://couchdb.apache.org/repo/bintray-pubkey.asc \  
| sudo apt-key add -
```

#### c. Installing CouchDB:

```
sudo apt-get update && sudo apt-get install couchdb
```

#### d. To validate the CouchDB installation:

```
curl -X GET http://127.0.0.1:5984
```



id	key	value
947618371658334208	947618371658334208	{ "rev": "1-7c7e7d999e7f0d05cfb63c180481..." }
947618375420784640	947618375420784640	{ "rev": "1-a2dd934433846755a8732a959e..." }
947618375731175425	947618375731175425	{ "rev": "1-b0f85ffcc412b62e59e2065837b33..." }
947618375810760704	947618375810760704	{ "rev": "1-8891be894c518b0c47a4c4c0484f..." }
947618375861141504	947618375861141504	{ "rev": "1-670a33d185306595c37ad54e7d5..." }
947618376062504967	947618376062504967	{ "rev": "1-cd3c756e15f084192c2980e702d5..." }
947618377345908736	947618377345908736	{ "rev": "1-0b49fbfb24282861740eee2dd8dd..." }
947618381154287617	947618381154287617	{ "rev": "1-2d18b0a69a276a71bea71676266..." }
947618382219743232	947618382219743232	{ "rev": "1-f1c9507a87757dac8abbfde4013e..." }
947618385977786368	947618385977786368	{ "rev": "1-920aa12d2760d4503ad8d5e797..." }
947618387332554754	947618387332554754	{ "rev": "1-36a7be9c465c72cc46189b0fc763..." }
947618740656488449	947618740656488449	{ "rev": "1-3b385f9342de73031dd9866ed536..." }
947618745245151232	947618745245151232	{ "rev": "1-87b325b57f7113680a5cebf2e7c71..." }
947618749875490817	947618749875490817	{ "rev": "1-ccc35163b13d3474a4ba0f43d063..." }
947618752429924352	947618752429924352	{ "rev": "1-3a643219f349b33fe92509d49117..." }
947618755768614913	947618755768614913	{ "rev": "1-2334c47a20a25d9907d2d70715b..." }

Figure: CouchDB being accessed through browser

The data is basically stored on the instance volume. This was done in case all the instances don't respond, we still have our data intact. By bringing the data storage onto volume we achieve:

- fault tolerance (e.g. if current VM instance that the volume is attached to went wrong, we can re-mount volume to another VM instance) and
- scalability (e.g. volume can be expanded if our operational data store demands more)

## Installing CouchDB-Cluster

Initially CouchDB is deployed as a standalone application, that is all data is handled only by one database. Currently the node name for standalone CouchDB is *couchDB@127.0.0.0*. In order to form cluster of CouchDBs installed on multiple instances we use *dpkg-reconfig couchdb* command on our master instance. This will change the node name to *couchDB@<ip>* and configure cluster mode for the databases. Now, in cluster mode the couchDB itself shards the whole data into equal parts (usually 8) and distributes these parts to the cluster databases in the membership list. At this moment the membership list will contain the standalone node name, hence we require to remove it and add the IPs for the cluster CouchDBs.

Upon performing these tasks, the CouchDBs will be in cluster mode, with sharded data distributed among the databases. For our configuration we have deployed CouchDBs on three instances, hence in case any instance goes down, the other instance containing the CouchDB

will be up and perform the required tasks as usual and will have access to data on other databases even in case if the acting database is not responding.

```
ubuntu@unimelbcc-prod1:~$ sudo debconf-show couchdb
couchdb/adminpass_again: (password omitted)
couchdb/adminpass: (password omitted)
couchdb/mode: standalone
couchdb/have_1x_databases:
couchdb/error_setting_password:
couchdb/nodename: couchdb@localhost
couchdb/adminpass_mismatch:
couchdb/cookie: monster
couchdb/bindaddress: 127.0.0.1
ubuntu@unimelbcc-prod1:~$ sudo dpkg-reconfigure couchdb
ubuntu@unimelbcc-prod1:~$ curl -X GET "http://admin:admin@localhost:5984/_membership"
{"all_nodes":["couchdb@172.26.38.114","couchdb@172.26.38.150","couchdb@172.26.38.40"],"cluster_nodes":["couchdb@172.26.38.114","couchdb@172.26.38.150","couchdb@172.26.38.40"]}
```

*Figure: Cluster mode membership list*

## 5. Processing

### 5.1 Pre-Processing

On attaining initial data from streaming API, we did a naked eye check for the trend of the tweets in terms of how people were reacting to immigration in Australia. This allowed us to gather keywords from tweets associated with immigration. We observed the terms such as ‘overpopulation’, ‘economic impact’, ‘anti-muslim’, ‘housing crisis’, etc. were associated with tweets talking about immigration. With the current ongoing election scenarios in Australia and various political parties tweeting about pros and cons of immigration, with each party presenting its view point, “#auspol” was another hashtag quite often associated with immigration. This initial manual analysis was done on 540 tweets out of the 30643 streamed tweets. A total of 163 keywords were chosen that were associated with our scenarios to perform further processing.

Additionally, on manually analysing the tweets we were able to segregate certain accounts that were involved in discussion pertaining to immigration and issues related with it. Twitter accounts of politicians, media organisation and public were picked to perform further collect tweets made by them through search API. A total of 87 accounts were picked from media organisation, politicians and public. Also, people following these accounts and/or retweeting their tweets were also put into work to attain more data associated to our immigration story. On performing search upon this analysis, by using the keywords, hashtags and user accounts, the count of gathered tweet was pushed to 187,432.

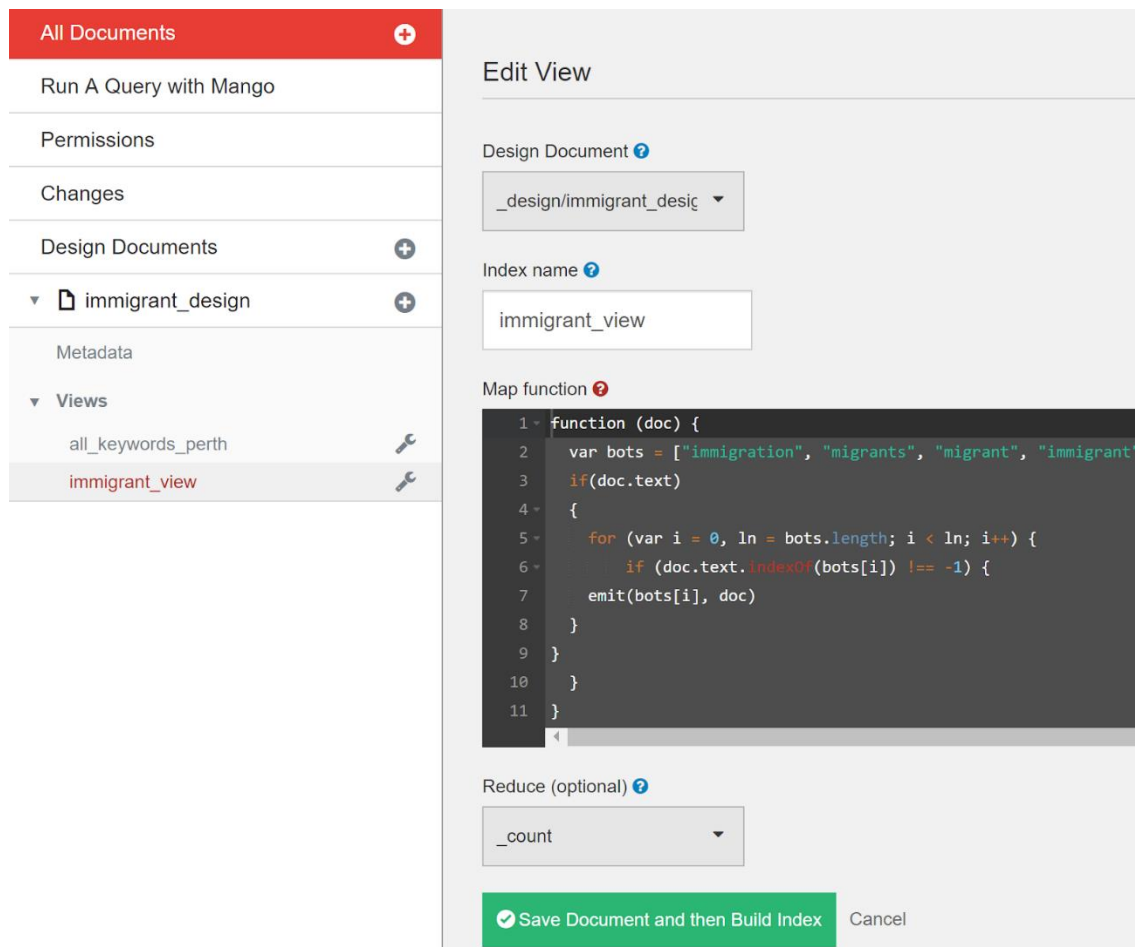
### **Problem faced and solution**

It was quite untidy to extract keywords manually as it was difficult to pick words frequently associated with immigration. Hence, we distributed the workload among all the members of the team. Each of us picked 5-6 user accounts who were frequently associating their tweets to immigration and followed their tweets and other users who were retweeting these tweets. Also, picked user account's followers and following were looked upon to attain base on gathering more related twitter accounts and keywords.

### **Using CouchDB: MapReduce**

For the data collected from UniMelb Research Cloud platform, which constituted generic data for all the six cities, we needed to segregate tweet as per our story requirements. The keywords and hashtags gathered from the initial analysis were put into work to filter out tweets which contained these words and were associated to our story. CouchDB's MapReduce was used applied on this dataset. The concept of MapReduce solves problems by applying a two-step process, firstly map phase and secondly the reduce phase. To filter the generic data map phase was applied to look at all tweets in CouchDB separately one by one and create a map result. The map result is an ordered list of key/value pairs. The ordered list was formed by applying the analysed keywords on the tweet data to execute the map phase.

Map phase resulted in formation of view in CouchDB which were formed by filtering out the full set of documents in the database based upon certain keywords. Two separate views were created for each database present i.e. for each city. One view contains the filtering on the tweets based upon how many people were actually talking about immigration in Australian cities, positive and negative sentiments were also captured from them. Second view contains the filtering on the tweets based upon keywords associated with immigration. These keywords were picked from the initial pre-processing of the streamed tweets.



*Figure: Creating view using Map function in database of immigration keywords*

Count attribute was applied on Reduce, the second phase, to get the count of tweets per keywords. This helped us with analysing how tweets were related to our scenarios and how they pertained to the sins.

## MapReduce Views

Three views were created for the databases. The views were for trend analysis, content analysis and profile analysis.

**Trend analysis view:** This view revolved around immigration as its centre point. 'Immigration' and its synonyms, that were observed in manual analysis of the tweets, were used to filter out and get the count of the tweets that were actually talking about immigration from the dataset.

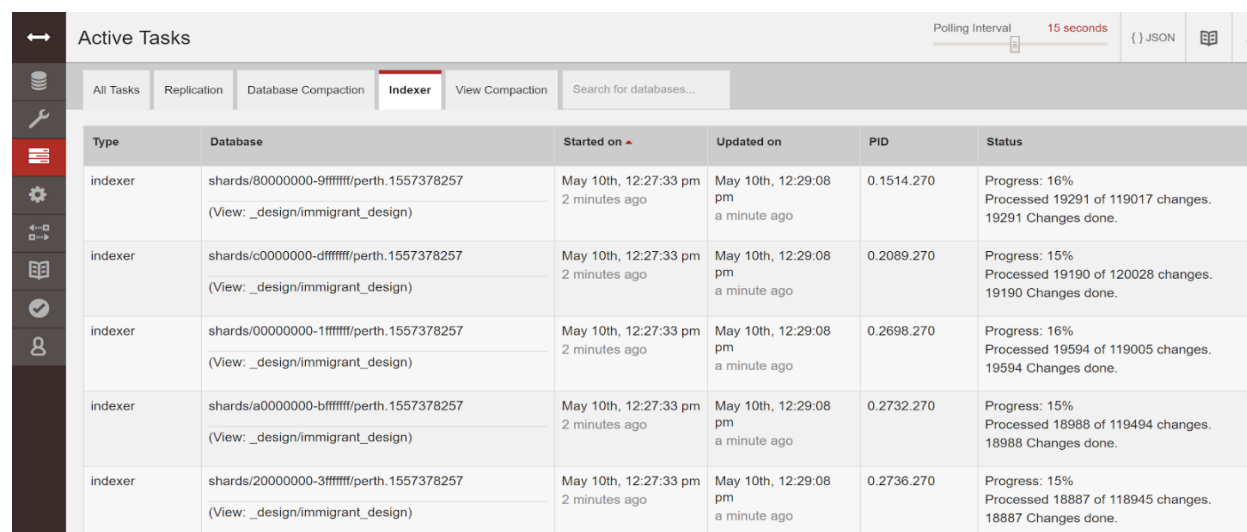
**Content analysis view:** This view was used to filter out and get the count of the tweets which mentioned and/or used keywords regarding our scenarios and story. This view was utilised to observe what were the main topic, such as issues of jobs, housing, migration, infrastructure,

that were being talked out. Sentimental analysis for the tweets was attached to the view to analysis if users were grudging regarding the scenario or were on the other side.

**Profile analysis view:** This view generates that count of the tweets with respect to users. This allowed us to analysis what category of users, either public or media or politicians, were tweeting and what was their attitude, positive, negative or neutral, towards immigration.

### Problem faced and solution

Database consisted of large number of tweets, hence the time consumed for converting the actually documents in database to view was on a higher side, approximately six minutes were being consumed to map 0.3 million tweets to view. Although, CouchDB supports sharding, which shards the whole dataset into equal parts to work on them parallely.



The screenshot shows the CouchDB 'Active Tasks' interface. The 'Indexer' tab is selected, displaying a table of active indexing tasks. The table has columns for Type, Database, Started on, Updated on, PID, and Status. There are five tasks listed, each representing a shard of the database. Each task shows progress and the number of changes processed.

Type	Database	Started on	Updated on	PID	Status
indexer	shards/80000000-9ffffff/perth.1557378257 (View: _design/immigrant_design)	May 10th, 12:27:33 pm 2 minutes ago	May 10th, 12:29:08 pm a minute ago	0.1514.270	Progress: 16% Processed 19291 of 119017 changes. 19291 Changes done.
indexer	shards/c0000000-dffffff/perth.1557378257 (View: _design/immigrant_design)	May 10th, 12:27:33 pm 2 minutes ago	May 10th, 12:29:08 pm a minute ago	0.2089.270	Progress: 15% Processed 19190 of 120028 changes. 19190 Changes done.
indexer	shards/00000000-1ffffff/perth.1557378257 (View: _design/immigrant_design)	May 10th, 12:27:33 pm 2 minutes ago	May 10th, 12:29:08 pm a minute ago	0.2698.270	Progress: 16% Processed 19594 of 119005 changes. 19594 Changes done.
indexer	shards/a0000000-bffffff/perth.1557378257 (View: _design/immigrant_design)	May 10th, 12:27:33 pm 2 minutes ago	May 10th, 12:29:08 pm a minute ago	0.2732.270	Progress: 15% Processed 18988 of 119494 changes. 18988 Changes done.
indexer	shards/20000000-3ffffff/perth.1557378257 (View: _design/immigrant_design)	May 10th, 12:27:33 pm 2 minutes ago	May 10th, 12:29:08 pm a minute ago	0.2736.270	Progress: 15% Processed 18887 of 118945 changes. 18887 Changes done.

*Figure: Sharding of documents while creating view using Map phase*

Option of using Python was there to overcome the high time consumption, by applying filtering in the initial phase when data was being saved onto the database. Due to time constraint of the assignment, we stucked with the initial method of applying map phase using CouchDB. Also, using Python to filter out tweet data initial wouldn't leave much work for the map-reduce functionality.

## 5.2 Sentimental Analysis

Dealing with immigration and related issues, it was necessary to perform sentiment analysis in order to analysis the general attitude of people tweeting regarding the issues. Sentiment

analysis was undertaken to judge if the tweets gathered are in general against, neutral or positive with the topic. Initially, Google Cloud platform's natural language API which provides sentimental analysis was implemented which gave neutral result when the score from sentiment analysis was between -0.25 to 0.25, negative result for score below -0.25 and positive result for score above 0.25. The limitations with using Google's natural language API is that it works by using credits which are paid, on exhausting the allocated credits we required to pay for continuing the analysis, which estimated manual would result to approximately \$500 for performing sentimental analysis on our data.

TextBlob was employed to overpower this deficit. TextBlob scores the tweet data by designating neutral if the score is 0, negative if the score is below 0 and positive if the score is above 0. TextBlob generates the score of the full tweet text by computing an average score for the words constituting the tweet. Each word has a polarity which TextBlob uses to generate an average score. Words such as 'hate' has negative polarity, 'love' has positive polarity and words such as 'I' has neutral polarity.

Before sentiment tagging the text of the tweet, pre-processing of the tweets was done. Pre-processing of tweets particularly means to refine the tweet text for further processing. The tasks in this pre-processing were:

1. Deleting URL and links present in the tweet text.
2. Removing special character (such as #, \$, etc.) and numbers from tweet text.
3. Removing username and/or account (tagged using @)

Pre-processing generated a cleaned text of the tweet which is then used for sentiment tagging. The purpose of sentiment analysis was to determine which tweet resulted to the sins. For example, if an user tweeting about immigration, was the tweet pro or against immigration was to determined. The sins explored for the assignment were - Pride, Envy, Greed and Wrath. Hence, depending on the scenarios the sins included within the scenarios were analysed to be positive, negative or neutral to the scenario.

Label	_id	text	user	created_at
NEUTRAL	1126092184170532864	@ozloman 3	{ "follow_request_sent": null, "profile...	Wed May 08 11:51:13 +0000 2019
NEUTRAL	1126092375405649921	Saaammee shiit	{ "follow_request_sent": null, "profile...	Wed May 08 11:51:58 +0000 2019
POSITIVE	1126092398382026752	You have good taste?	{ "follow_request_sent": null, "profile...	Wed May 08 11:52:04 +0000 2019
POSITIVE	1126092406372257794	Surprised to learn both Morrison an...	{ "follow_request_sent": null, "profile...	Wed May 08 11:52:06 +0000 2019
NEGATIVE	1126092489209798657	@Latte_Bogan @MCavoodle ...and ...	{ "follow_request_sent": null, "profile...	Wed May 08 11:52:25 +0000 2019
NEGATIVE	1126092514199461888	@RitaPanahi He's a nasty piece of ...	{ "follow_request_sent": null, "profile...	Wed May 08 11:52:31 +0000 2019
NEGATIVE	1126092660291264512	Grilled Chicken with Fried Basil and ...	{ "follow_request_sent": null, "profile...	Wed May 08 11:53:06 +0000 2019
NEUTRAL	1126092692797026304	@jack13maloney @instagram No si...	{ "follow_request_sent": null, "profile...	Wed May 08 11:53:14 +0000 2019
POSITIVE	1126092726489890816	@ErniMarie Surely has to be Spen...	{ "follow_request_sent": null, "profile...	Wed May 08 11:53:22 +0000 2019
POSITIVE	112609275625936385	@SkyNewsAust @ScottMorrisonMP...	{ "follow_request_sent": null, "profile...	Wed May 08 11:53:29 +0000 2019

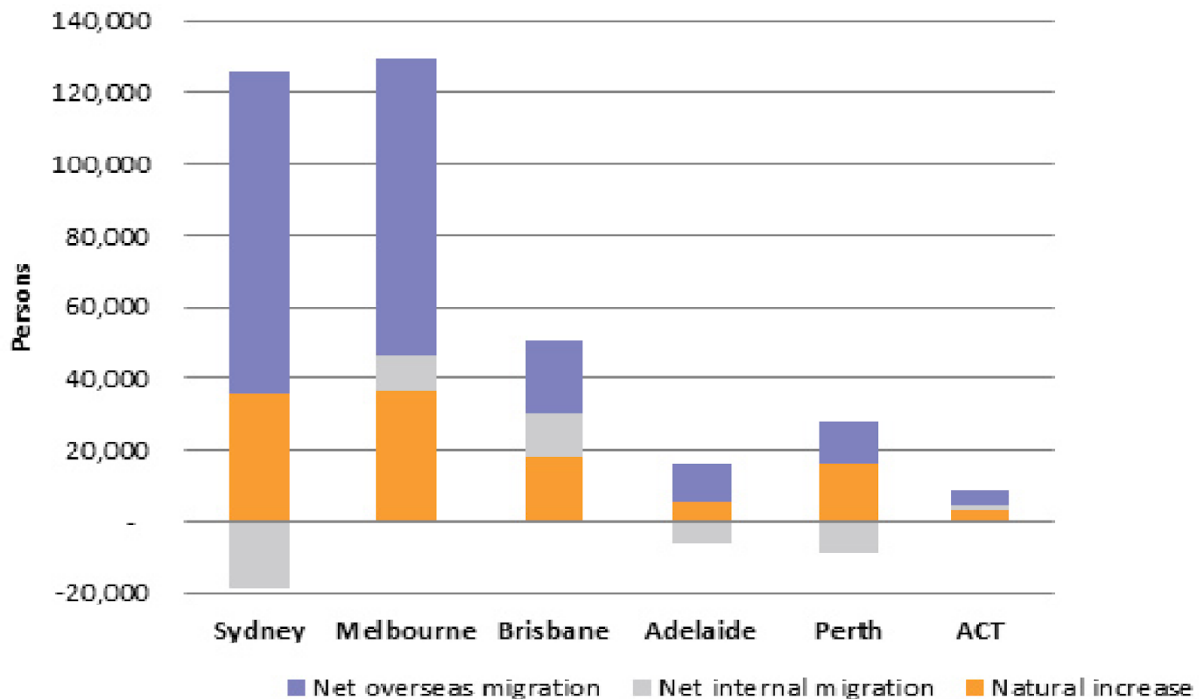
Figure: Sentimental analysis into positive, negative or neutral of tweets in Melbourne area

### 5.3 Data Analysis

Our group being overseas born studying in Australia we decided to go with the immigration topic, since we found a large population of non-Australian around us and ourselves thought was immigration has brought to Australia. The idea was to depict our main story on immigration via creating scenarios. The data collected from twitter and its analysis will mount the scenarios which we verified and related to AURIN data. Each sin explored consumes one or more than one scenario analysis. Upon understanding the kind of data, we had gathered, we had decided to term our scenarios as trend, content and profile analysis. This was done in order to provide a better insight on what the scenarios tend to achieve.

Immigration being one of the hot topics with ongoing elections in Australia (we followed #auspol) people have being talking about the pros and cons immigration on twitter, including politicians and citizens. Around 48,392 tweets were sourced which talked or mentioned 'immigration'. With immigration affect the population growth in Australia, resulting into numerous factors such as 'overburden', 'unemployment', 'housing crisis', 'traffic volume' has increased hatred amongst the locals with 44.1% tweets being made in negative sense.





*Figure: Impact of Immigration on Australia's overall population growth*

### 5.3.1 AURIN Data

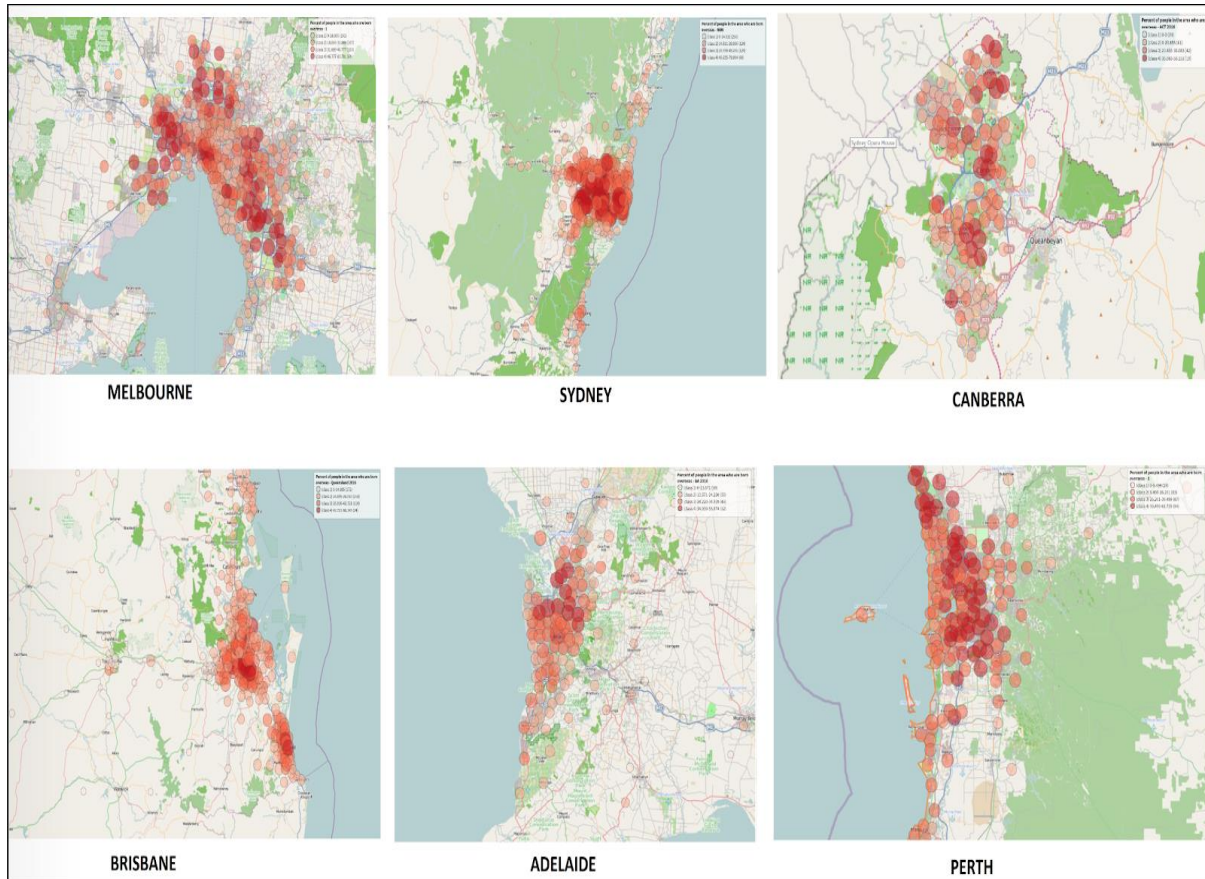
The Australian Urban Research Infrastructure Network (ARUIN) is a crucial platform which provides access to data and tools across Australia for researchers. The data provided allowed us to make evidence-based decisions confidently and aided in cross verifying our scenarios.

For each city the below mentioned data was sourced from AURIN in order to analyse trend and base our conclusions:

- [SA2 OECD Indicator: Migration Rate 2011](#): Provides the number of people migrated from overseas and the total population of that area for the year 2011.
- [NATSEM - Social and Economic Indicators - Migration Rate SA2 2016](#): Provides the number of people migrated from overseas and the total population of that area for the year 2016.
- [SA4 Estimating Homelessness 2011](#): Provides the number of homeless people in a particular area for the year 2011.
- [SA4 Estimating Homelessness 2016](#): Provides the number of homeless people in a particular area for the year 2016.
- [SA4-W08e Industry of employment by year of arrival in Australia by Proficiency in Spoken English - Census 2016](#): Provides the overseas population employed in different field of work.



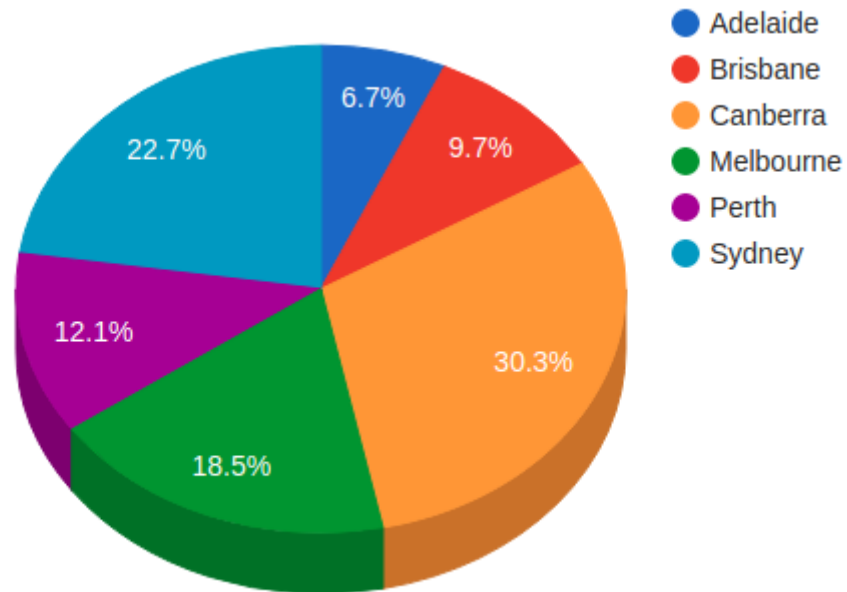
- **NATSEM - Social and Economic Indicators - Unemployment Rate SA2 2016:**  
Provides the percentage of people unemployment out of the total population for a particular area.



*Figure: Overseas born population percentage for all the cities (AURIN)*

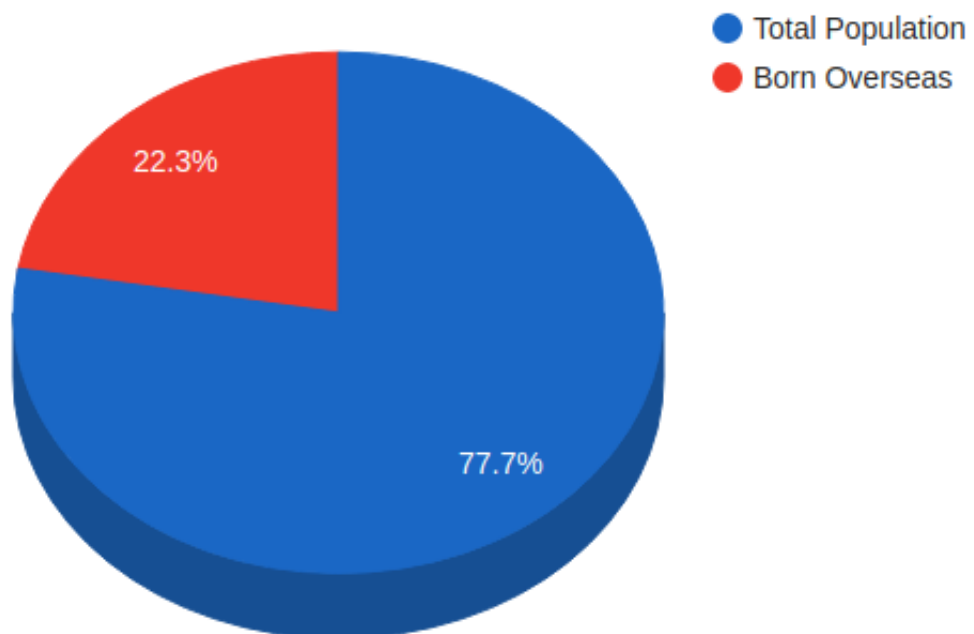
### 5.3.2 Trend Analysis

Revolving around immigration as our centre point, we carried out analysis to observe how many twitter users were actually talking about immigration, or synonyms words such as migrant, seeker, refugees. Search API carried on keywords consisting of ‘immigration’ and related synonyms, fetched 77,531 tweets from a period of 12 days. Manual analysis of these tweets was done, and further keywords such as ‘settling in Australia’, ‘moving to Australia’, ‘jobs in Australia’ were extracted and map-view was run on the database of 77,531 tweets. This generated a resultant database of 39,478 tweets which shows that people are choosing Australia as destination for their [greed](#), may it be due to jobs, or better quality of life. AURIN statistics shows that a total of 3,056,940 overseas born population were employed in the six cities under study.



*Figure: Percentage of tweets made city-wise related to immigration*

With overseas born population constituting 22.3% of the six cities under study, it can be seen that people are preferring to move to Australia maybe to due [greed](#), with Australia ranking 4 amongst the most immigrant friendly country [2].



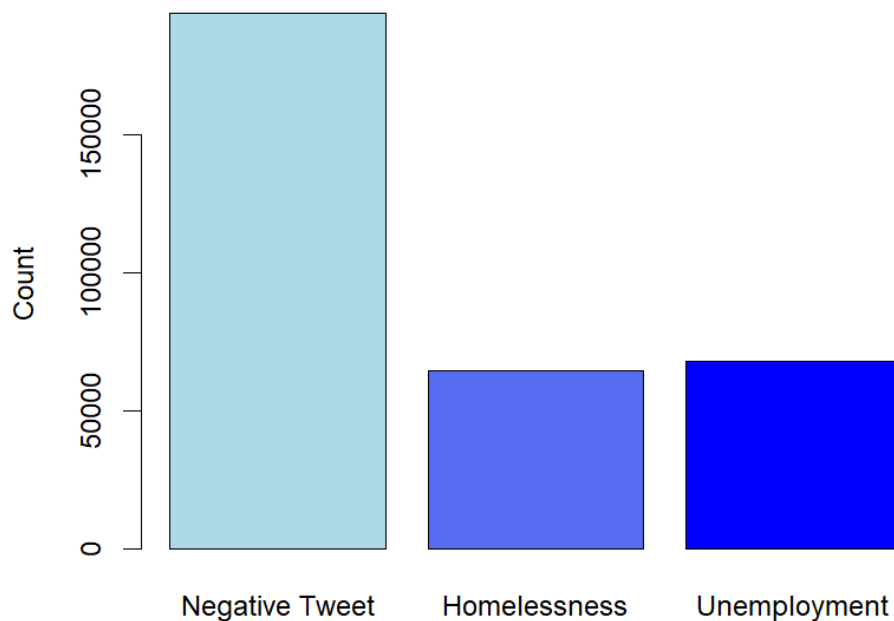
*Figure: Percentage of overseas population in Australia's six cities (2016)*

### 5.3.3 Content Analysis

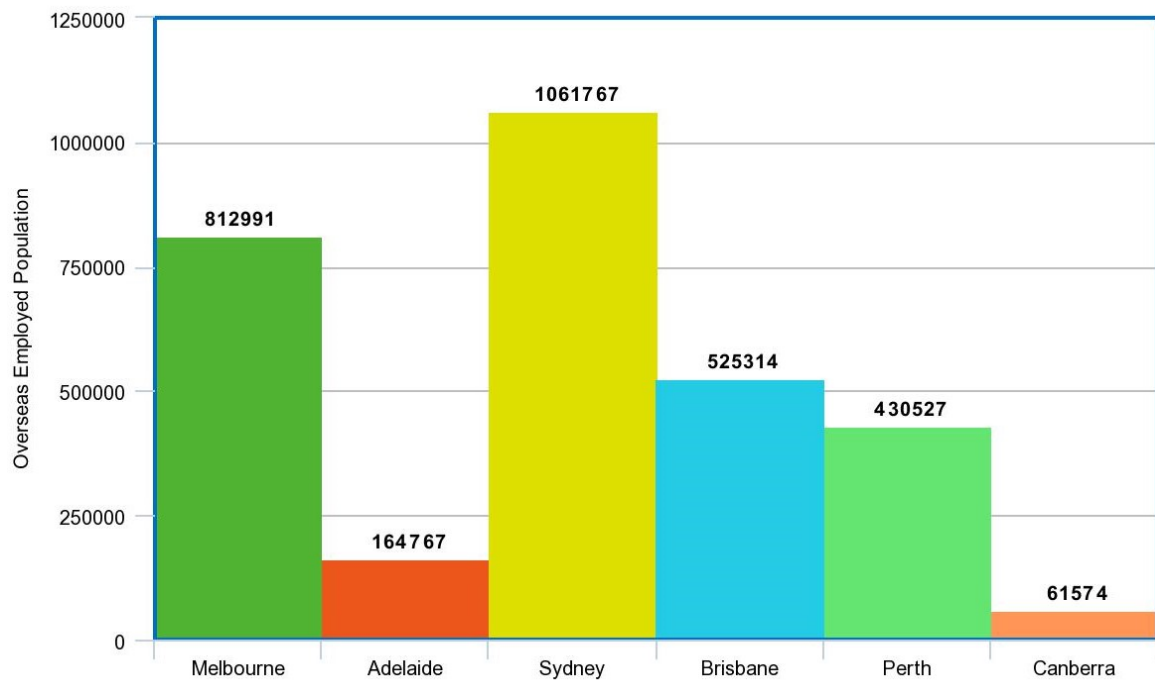
On performing pre-processing on the data collected we extracted a list of keywords that were associated to the topic of immigration. A total of 163 keywords were chosen to analysis the contents related to our story. A total of 448,557 tweets were collected for the list of keywords. Out of which 194,189 tweets showed negative sentiments as compared to 180,338 positive tweets.

These negative sentiment tweets show that:

- There **envy** and **wrath** against the migration population in Australia, as issue such as unemployment and homelessness are rising parallel. The six cities recorded 64647 homelessness count and 67970 unemployment count, with people actively talking about these issues on twitter, as 180,000 tweets were harvested which talked about issues related to immigration.
- **Envy** can also be judged as we see a large number of overseas born population being employed in Australia. The below graph depicts the number for each city under study.



*Figure: Tweets talking about migration related issues (Negative Tweets) with count of homelessness and unemployment in six cities from AURIN*



*Figure: Number of overseas born employed in Australia, city-wise*

To perform a better analysis on the actual topic related to immigration we segregated the keywords into five groups, which were as follows:

- Group 1 - Immigration and synonyms keywords (48,800 Tweets)
- Group 2 - Factors being talked about related to immigration (180,00 Tweets)
- Group 3 - Issues with immigration (13,600 Tweets)
- Group 4 - Immigration related hashtags (94,700 Tweets)
- Group 5 - Other Keywords (192,00 Tweets)

### City-wise analysis:

**Melbourne:** Major tweets sourced from this city had negative sentiments towards immigration, with majority of the tweets (35,229) talking about factors related to immigration. Out of these tweets 99.3 % tweets were in negative sense. This is usual since Melbourne recorded the second highest migration rate of 26.35% among the Australian cities. Also, Melbourne had an unemployment rate 6.3, which is above the Australian average of 5.6.

**Brisbane:** The tweets sourced from this city had a positive attitude towards immigration and related issues, which major tweets (32,886) talking about other topics related to immigration. Out of there 16,100 tweets were in positive sense. Although the homelessness count rose for Brisbane from 2011 to 2012 and unemployment rate was 7.87.

**Perth:** The tweets sourced from this city had a neutral attitude towards immigration and related issues, which major tweets (21,013) talking about other topics related to immigration. Out of these 10,200 tweets were talked in neutral sense, even though Perth recorded the highest immigration rate of 27.93%. But Perth has seen a slight decrease in homelessness count from 2011 to 2016.

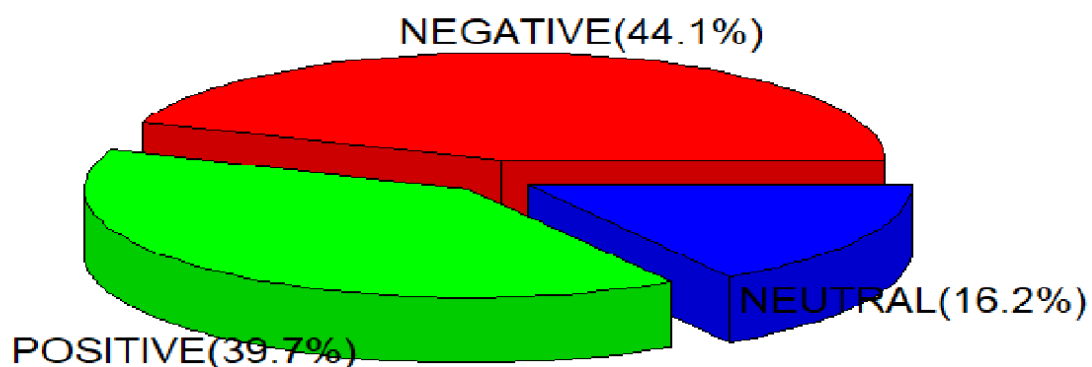
**Adelaide:** Major tweets sourced from this area had negative sentiments towards immigration, with majority of the tweets (38,266) talking about factors related to immigration. Out of these 99.7% of the tweets had negative sentiment. This is usual since Adelaide recorded an unemployment rate 6.96, which is above the Australian average of 5.6 and homelessness increasing by 6%.

**Sydney:** The tweets sourced from this city had a positive plus neutral attitude towards immigration and related issues, which major tweets (63,261) talking about other topics related to immigration. With 26,200 tweets having neutral and 25,900 tweets having positive sentiment.

**Canberra:** The tweets sourced from this city had a neutral attitude towards immigration and related issues, which major tweets (46,591) talking hashtags related to immigration and elections such as 'auspol'. This is usual behaviour since unemployment rate for Canberra is recorded at 3.67, lower than the average, and the lowest homelessness rate among the six cities.

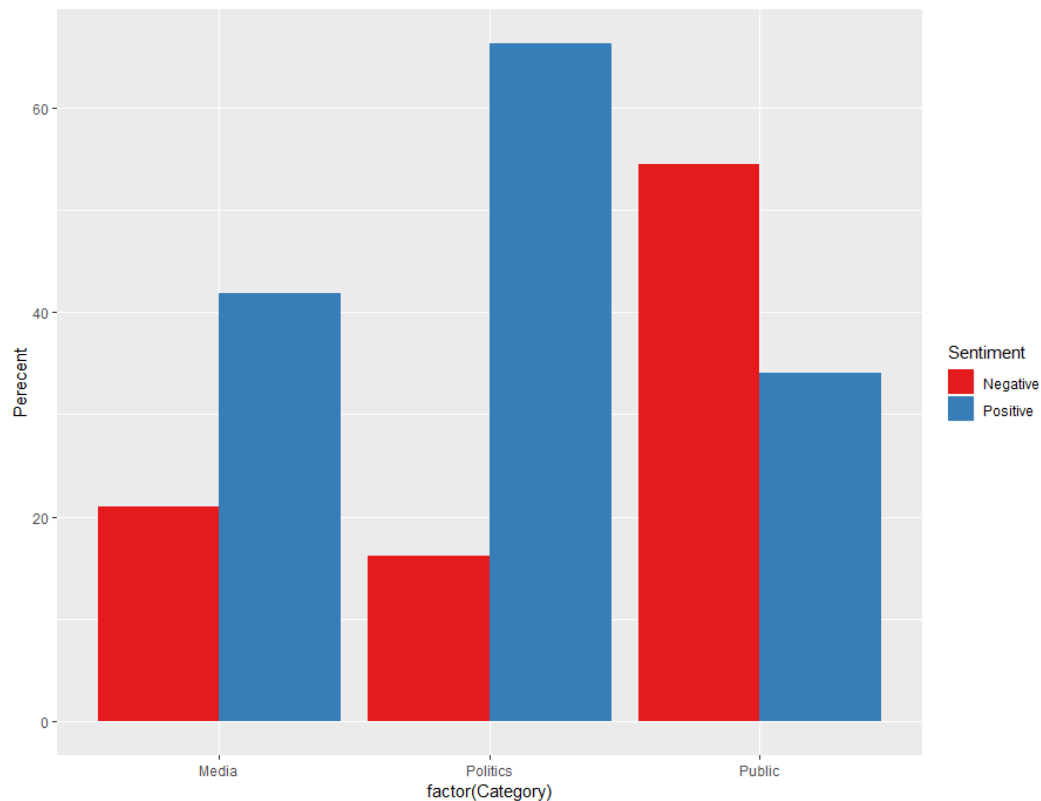
### 5.3.4 Profile Analysis

On manual analysis of tweets 86 profiles were picked from public, politics and media/organisation users who were actively involved in immigration related topic. It was observed that the overall sentiment towards immigration was negative, constituting 44.1% of the overall data.



*Figure: Users' (picked by manual analysis) tweet sentiment analysis*

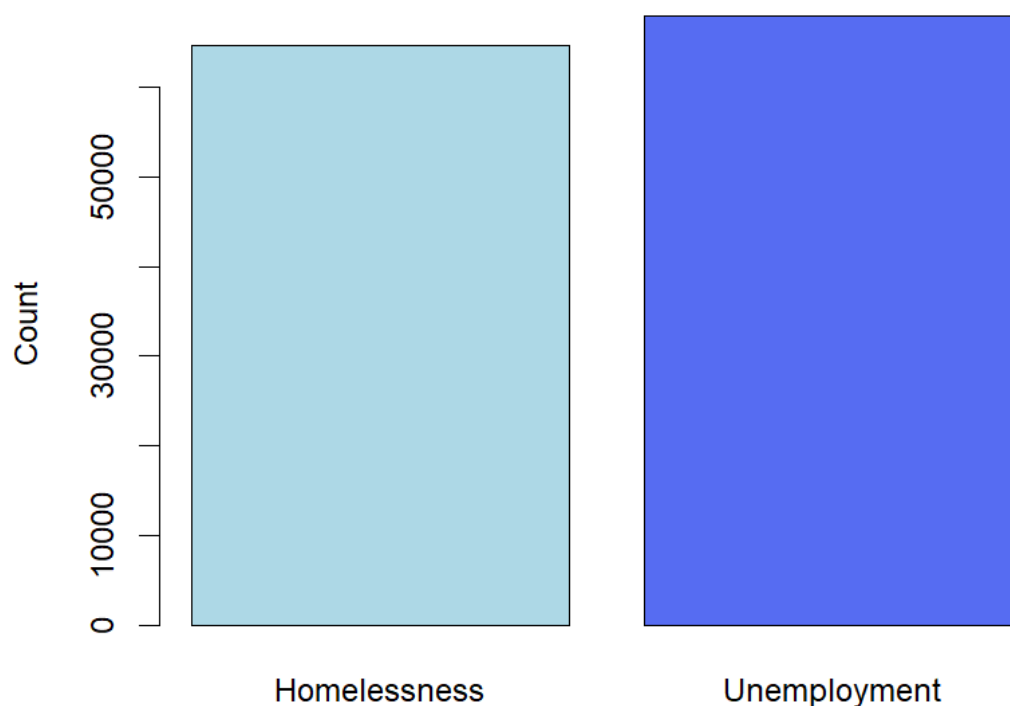
Observing each profile separately it was observed that negative-ness towards immigration was majorly among general public users, whereas politicians and media organisations showed a positive attitude towards the issue.



*Figure: Out of the total tweets made by each - percentage of negative and positive sentiment for Media, Politics and Public profiles*

54.5% tweets from public profile were targeted in a negative sense, showing that:

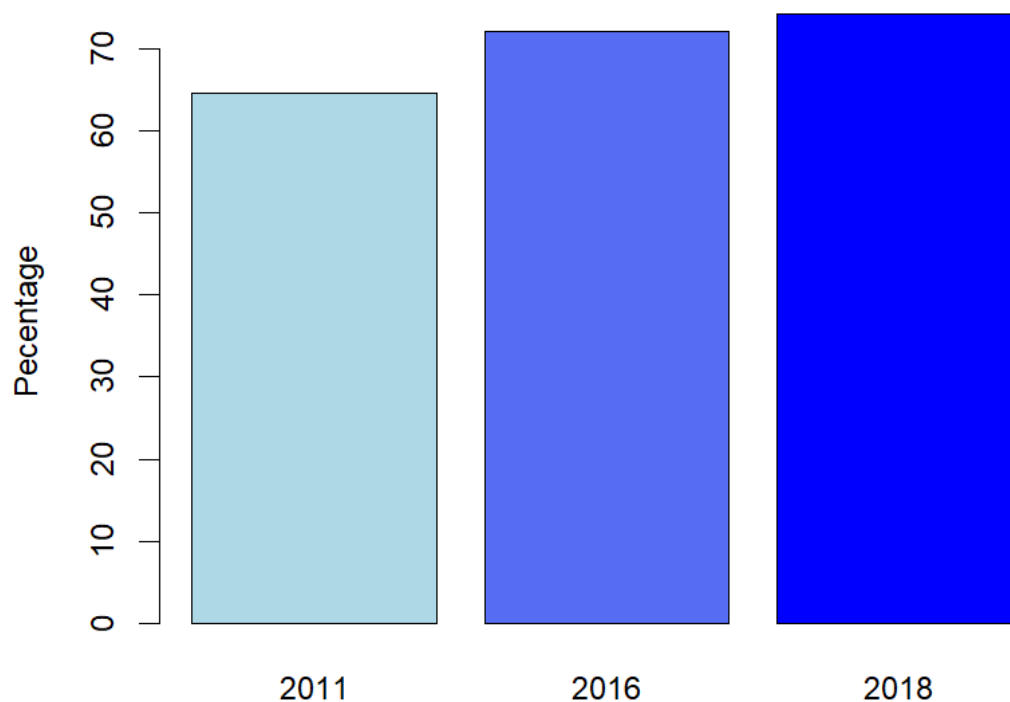
- The local Australians have **hatred (wrath)** against overseas population as people are relating issues such as overpopulation, traffic, unemployment, housing stress, etc. to immigration. Out of the 80,880 tweets, specific to the public profile users, 58,870 tweets were related to immigration issues.
- **Envy** can also be associated here as suggested by AURIN the total overseas born population rising to 6,042,112 out of the total population of 21,061,180. Also with people talking about unemployment, homelessness and housing stress, with immigration population employed in the six Australian cities considered here is 3,056,940.
- As result of wrath local Australians spoke out for taking actions against immigration rise. Tweets related to ‘skilledvisa’, ‘anti-black’, ‘anti-muslim’, ‘anti-asian’ etc. mounted to 30,200 for the public profile.



*Figure: Number of Homeless people and Number of Unemployed people in the ^ cities of Australia (AURIN)*

Whereas, users from media and politics profile tend to have a positive attitude towards immigration, with 41.8% and 66.3% positive sentiment tweets respectively, showing that:

- These user profiles tend to talk positively about immigration with an intention of greed. Keywords such as ‘day without immigrants’, ‘noban’, ‘multiculturalism’, ‘migration labour’, ‘colonization’, ‘economy’, etc. were used. Out of the 146,120 tweets for these two profiles, these keywords tweets constituted 69,536.
- Which shows the greed to flaunt the statistics of the country in terms of employment, average age reduction, per capita income and housing and taking pride for the same. As with migration the employment rate increase by 0.8% for the six cities. Also, 85% of the immigrants are below the age of 45, as compared to 45% population of Australia [6].



*Figure: Employment Trend For 6 Australia cities (AURIN)*

## 6. Conclusion

The increasing migration in Australia as brought in about benefits as well as cons with it. Increasing per capita income, decrease in average population age and increase in employment rate are the some of the boons from immigration. On the other hand, increasing homelessness, spike in housing prices and overpopulation are some of the banes from high immigration rate. We constructed our story for the assignment around immigration and built scenarios in order to relate the data being harvested to the seven deadly sins. We performed trend, content and profile analysis on the data and included one or more sins within the analysis. We observed that with Australia being a hot destination for migrants to move, there has been a high rise in overseas population and people have been talking about it over social media. Each individual holds the right to his/her opinion may it be negative or positive. The sentiment analysis carried out as part of this assignment we observed that not significant (44.1% negative and 39.7% positive) but general trend is of negativity towards immigration and related issue. People



actively took part in tweeting about immigration related issues may it be due to cause to envy or wrath.

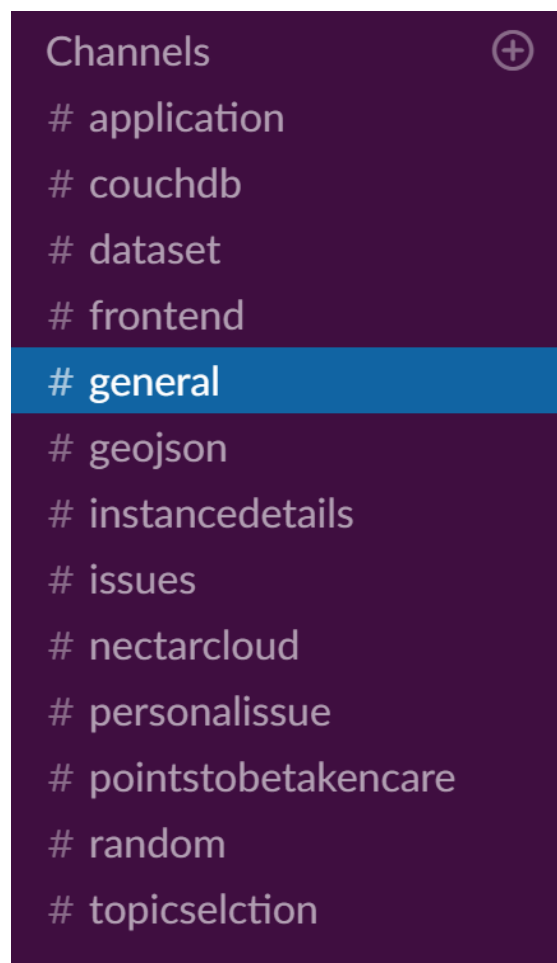
## 7. Repository and Communication

GitHub: <https://github.com/pallsac/CloudAssignment2>

YouTube: <https://youtu.be/UvZFuy2bBo0>

### 7.1 Communication

All the members were involved extensively from the initial discussion of the scenarios till constructing the video for the complete assignment deployment. Tasks were distributed among the team members equally by segregating the work into instance set-up, CouchDB, harvesters, ansible scripts and front-end application. Although each member was actively involved in every component of the assignment We met regularly (2-3 time a week) to discuss progress and integrate our works. Elsewise, slack (<https://slack.com/intl/en-au/>) group was created for communication purpose.



*Figure: Communication channels on slack for the team*

## 8 References

- [1] Keita Kurita, 2016, 'Twitter-past-crawler' <https://github.com/keitakurita/twitter-past-crawler>
- [2] Immigration in Australia, [https://en.wikipedia.org/wiki/Immigration\\_to\\_Australia](https://en.wikipedia.org/wiki/Immigration_to_Australia)
- [3] [https://en.wikipedia.org/wiki/Seven\\_deadly\\_sins](https://en.wikipedia.org/wiki/Seven_deadly_sins)
- [4] <https://aurin.org.au/resources/aurin-apis/>
- [5] <https://boundingbox.klokantech.com/>
- [6] <https://www.afr.com/news/politics/migrants-deliver-positive-benefits-to-the-economy-new-treasury-paper-finds-20180416-h0yuc3>