Cluster and Cloud Computing
COMP90024

Assignment 2

Team 26

Name	**Student ID**
Ariel Kark	521428
Charu Smita Singh	1029083
Hina Bagai	1070601
Raksha Kammandore Ravi	1094298

Table of Contents

1. Abstract

This report outlines the work undertaken by Team 26 towards completion of Cluster and Cloud Computing - Assignment 2 wherein a Social Media Analytics system was constructed with the aid of National eResearch Collaboration Tools and Resources (Nectar) in conjunction with datasets from Twitter and AURIN to present interesting stories about life in Australian cities. It aims to provide valuable insights pertaining to different scenarios prevalent in Australian cities through detailed analysis and intelligible representations of the knowledge gained.

The proposed research mainly focuses on the scenario in each state in Australia, where the tweets are in a foreign language and draw a correlation to the English language proficiency of immigrants in that state. Thus the analysis provides us with adequate knowledge to decide which state in Australia has maximum foreign language tweets with respect to an immigrant's English language proficiency.

The harvester program is used to gather tweets from 6 states of Australia and uses analytics by comparing it with AURIN data to determine the state with maximum foreign language tweets based on non-proficient English language immigrants.

We mapped the correlation using MapReduce function and narrated the analytics story through spectacular visual charts on our website.

2. Introduction

2.1 Twitter

With 326 million users and up to half a billion tweets per day, Twitter has become an extremely valuable source of information for the analysis of any given society. Twitter is a global phenomenon which generates massive amounts of data related to ongoings in the world. Twitter users express their thoughts, emotions and opinions in the form of short posts known as tweets which is generally a combination of character strings, hashtags and emojis. Furthermore, Twitter provides a geolocation feature which spontaneously adds the tweeter's location to each tweet.

The informational richness of a tweet is not confined to the text itself, but bolstered by the metadata, which includes geotags, timestamps, user information and hashtags. This allows inferences to be made about certain areas, in certain timeframes and even about specific individuals. These elements tied together with the analysis of the tweet text, can provide invaluable insights into diverse social phenomenons.

We aim to study the relationship between the number of tweets in a foreign language in a given state to the level of english proficiency of the immigrant population in that Australian state.

2.2 Big Data

The sheer size of data generated by Twitter users presents researchers with a challenging issue, namely the design of a system to ingest Big Data and output insights that are useful and relevant. The definition of Big Data is not confined to the sheer volume of data being analysed alone, but also by a further 3 elements starting with V:

1. Velocity - The speed at which a system must ingest data to generate outputs.

2. Variety - The level of variability in a databases' schema, namely, its ability to adapt to different forms of data.
3. Veracity - The accuracy of the data that is being collected and used to generate outputs.

2.3 System Architecture Overview

Perhaps the main considerations when facing the challenges of dealing with the above challenges of Big Data involve the selection of the computing resources allocated to generate insights, as well as their system architecture and deployment. An effective solution is using a cluster based architecture, deployed on a number of critical machines hosted by an Infrastructure as a Service (IaaS) provider.

At a high level, the system produced has 4 central functions:

1. The harvesting of Twitter data using the Twitter Developer API.
2. The storage of data harvested from the above process in CouchDB.
3. The analysis of the data using MapReduce, and validation through comparison with data from AURIN.
4. The visualisation of findings through a web based user interface.

2.4 NeCTAR

To allow for the above deployment of resources, the task has required the utilization of the Nation eResearch Collaboration Tools and Resources research cloud, otherwise known as NeCTAR, which is predicated on the open stack protocol. Using the computer networking and storage resources offered by NeCTAR, the technology stack outlined above could be deployed as a cluster based system.

2.5 Problem Statement

This assignment presents the task of building a system which harnesses the power and efficiency of cloud computing in the form of multiple Virtual Machines, to both harvest and analyse Twitter data, in order to draw insights into communities within Australia. The findings generated by this analysis can then be validated using the AURIN platform.

To see the relevant source code discussed in this report, please see:
https://github.com/hbagai/Cluster-CloudComputing
To view the video of our system, please see:
https://www.youtube.com/watch?v=pE_Ccjfxw6A

3. System Architecture

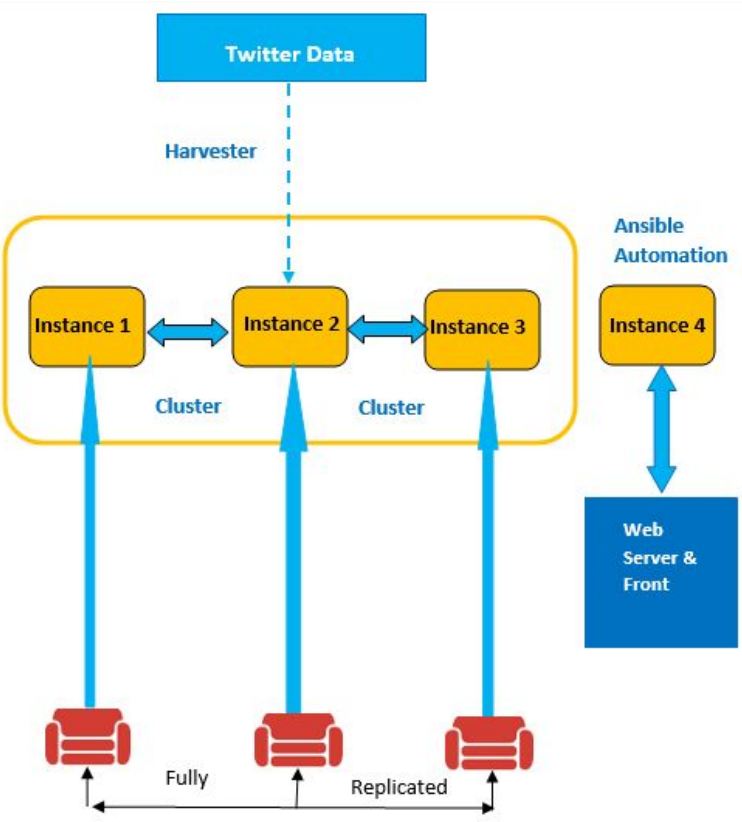
Four virtual machines/instances have been set up over NeCTAR, with all instances of the flavor uom.mse.2c9g, with Ubuntu 18.04 preinstalled, and have an attached volume of 250 Gb in total. Three instances hosting CouchDB have been assigned 70 Gb each, whereas 40 Gb has been assigned to the Website server. Basic security groups such as http, ssh, etc have also been assigned to each instance.

The automation of creation and deployment of the virtual machines has been done using the management tool, Ansible. The prerequisites for using Ansible are:

(1) Linux environment

(2) Python3 (since we have installed Ansible through Python package installer)

(3) Access to the instances created on Nectar through SSH connections using the private key file generated for the project



The four central functionalities of the system produced rely on the system architecture outlined in the adjacent diagram.

3.1 Tweet Harvesting

The dataset used for analysis was produced through access to Twitter's REST APIs

3.1.1 REST

REST (REpresentational State Transfer) design allows for efficient communication between systems over the internet. RESTful architecture is based on 4 principles as outlined by Pautasso et al. in "RESTful Web Services vs. Big Web Services: Marking the Right Architectural Decision":

1. Identification of resources through URIs

   URIs provide a universal addressing space for resources and services on the web, that allow them to be uniquely identified and easily discoverable.

2. Uniform Interface

   The above resources can be interacted with and manipulated using four main operations; PUT, GET, POST and DELETE.

3. Self-descriptive messages

   Resources are represented in a variety of formats, regardless of their actual representation at source. In this assignment, JSON format is used.

4. Stateful interaction through hyperlinks

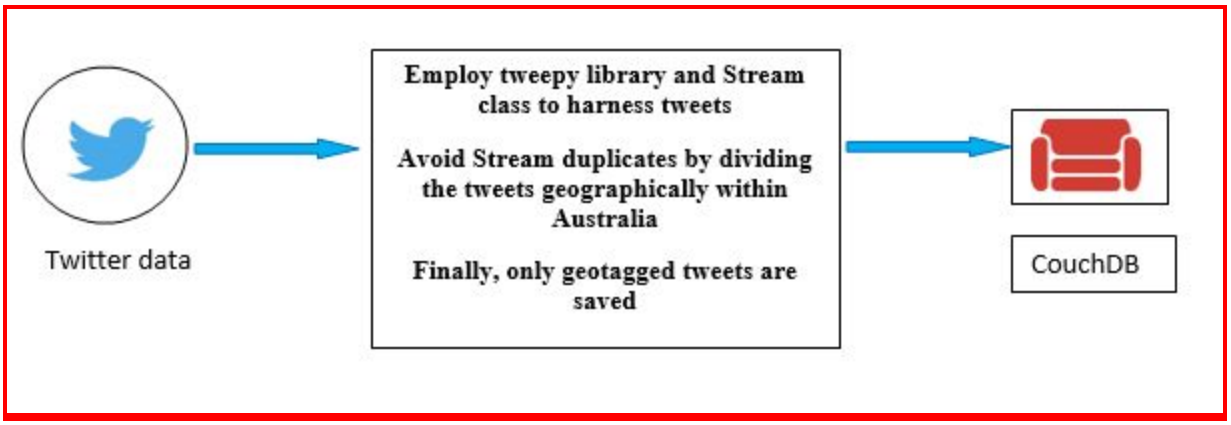
   A resource has no state, and therefore, a variety of techniques are provided to allow the exchange of states between systems, for example cookies and hidden form fields.

3.1.2 Tweepy

Tweepy is the Python library used to harvest tweets using the RESTful API supplied by Twitter. The Stream class of the Twitter API was used in the Python code to harvest tweets. By using the Stream class, it was possible to purley stream tags that are within a given bounding box. With

four members in the group, it was possible to divide the geographic area of Australia using four different Twitter Developer accounts to avoid the streaming of duplicate tweets. Further, only tweets that had been geotagged were stored. This ensures they could be mapped when analysed at a later stage.

Further preprocessing was required to ensure the persistence and consistency of the data being stored, namely implementing further preventive measures from duplicate tweets being stored, as per their tweet id.



3.2 Database - Apache CouchDB

Apache CouchDB, a NoSQL, documented oriented database, is responsible for the storage of the tweets harvested in the above process. Documents, in this case tweets, are recorded in JSON form, which is highly efficient as they are represented as this form when extracted using Tweepy. The main features of CouchDB is its ability to be deployed in a distributed fashion, while allowing data querying and manipulation through MapReduce functions and RESTful API/HTTP calls.

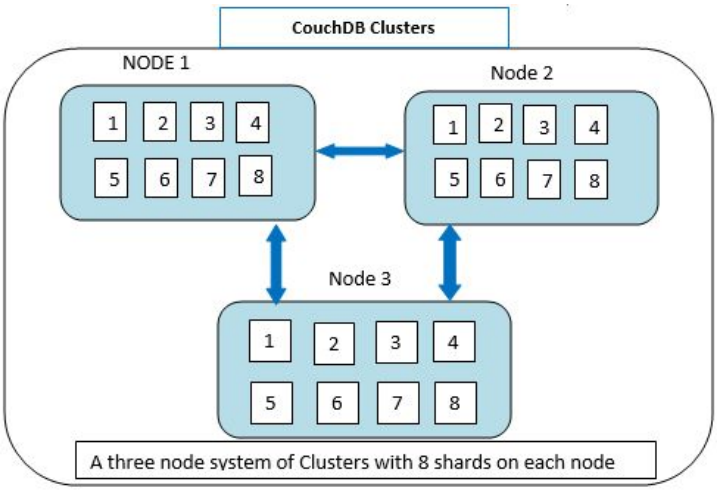
3.2.1 Clustered Architecture - Sharding and Replication

The distribution of data between nodes in CouchDB relies on two key parameters:

1. Sharding - data is split *horizontally* across nodes increasing the overall performance of the database.

2. Replication - data is replicated amongst nodes to ensure the availability and redundancy of data in the database.

Notably, these features are designed to remedy potential issues, including a lack of partition tolerance and availability in a database, as per Brewer's CAP Theorem. CouchDB prevents the impact of these potential issues through Multi-Version Concurrency Control (MVCC). MVCC ensures a database's availability and efficient recovery from partitioned nodes through a system that relies on *revisions*. Revisions create an environment where data is very rarely replaced or deleted, but rather provided with a new revision number upon being changed. In regards to the requirement for data consistency, when CouchDB receives multiple concurrent updates, instead of locking the database (as many database designs do), each transaction receives a revision number. The last transaction to be completed is then considered to be the *current value,* and the value presented in the database. It must be noted that CouchDB only ensures eventual consistency through *conflict resolution.* This is a process where data that has been replicated between nodes receive concurrent requests, only to be resolved by configuration at the user level. This is perhaps a key trade off when using data replication and a distributed system architecture.



When discussing the distributed architecture of CouchDB, there are two key parameters to outline; the number of 'shards' (q) and the number of replicas (n). In this context, the default CouchDB architecture is applied, namely, 24 shard replicas, derived from eight shards and three replicas, as displayed in the adjacent figure. In this case, a three node cluster has been deployed, leaving eight shard replicas to be hosted by each. This satisfies the recommendation by

CouchDB that shard replicas should be divided evenly amongst nodes, and reshuffled when the number of nodes is scaled.

3.2.2 Data Manipulation - MapReduce and API

API:

The code used for harvesting tweets relies on the CouchDB API to save tweets for future analysis. Further, the web UI also utilises the CouchDB API to pull data for the visual representation of all analysis conducted.

MapReduce:

Perhaps one of the most powerful features of CouchDB is the ability to manipulate data through MapReduce functions. MapReduce functions, as one may expect, can be divided into two parts, Map and Reduce. Initially, the Map function is run, which produces a list of key-value pairs, which is 'reduced' by the Reduce function to a desired output. The MapReduce functions used in this project are outlined below.

1. AllTweet:

   The allTweet MapReduce function applies a sanity check by filtering only Tweets originating from Australia. It then goes on to query the origin and language of each tweet stored in the database.

   ```
   function (doc) {
     if(doc.tweet.place.country_code == "AU"){
       emit(doc.tweet.place.full_name, doc.tweet.lang);
     }
   }
   ```

2. Language:

The language MapReduce function maps all tweets that originate from Australia and possess a language code that is not “en” (English). It then reduces the output to the language of the Tweet and its origin location.

```
function (doc) {
  if(doc.tweet.lang != "en" && doc.tweet.place.country_code == "AU"){
    emit(doc.tweet.lang, doc.tweet.place.full_name);
  }
}
```

Together with MapReduce functions, when conducting analysis on the data, CouchDB implements views as the primary mechanism for the production of MapReduce query outputs. In this case, there are two separate views used for analysis, which are generated by their relevant MapReduce functions:

1. Alltweets

   The alltweets view shows all the origins of a tweet, reduced by city, indicating the count of tweets originating from that city.

   /tweet/_design/newview/_view/alltweets?reduce=true&group_level=1

2. Languages

   The languages view reflects all non-english tweets and the cities they are from

   /tweet/_design/newview/_view/languages

3.3 Web Server - User Interface

CouchDB with Flask is an easy and powerful system. The installation steps are outlined below:

Installation:

We need CouchDB. Once the CouchDB database is installed, you can install Flask-CouchDB if you have pip installed then:

$ pip3 install Flask

To begin with the tweets are stored on CouchDB instances. One amongst the three instances have been connected to the Web Server. Here we have used FLASK. The view is stored on CouchDB accessed by the data analysis python file. Flask is a WSGI web app framework. It was created to make steps easy for web application designing and structuring. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

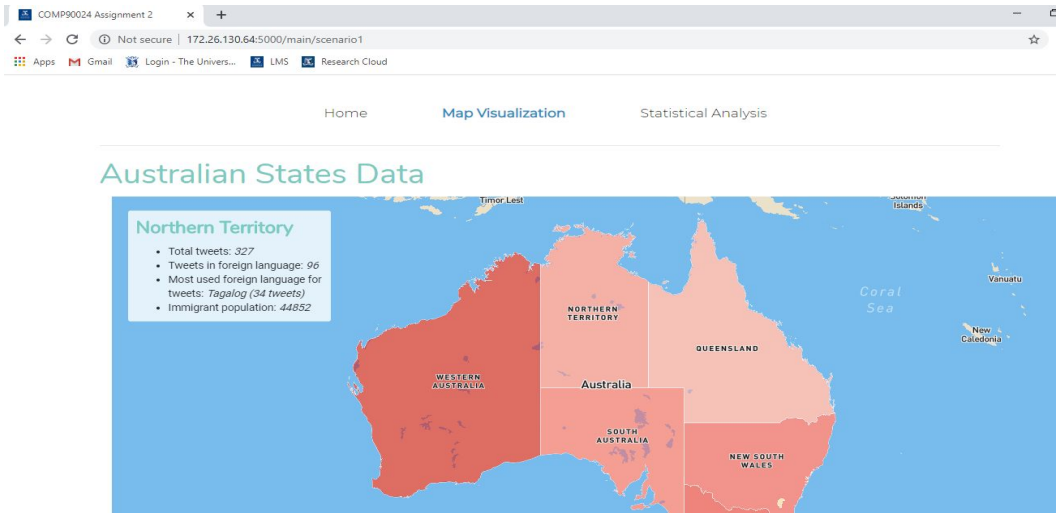
The json data retrieved from CouchDB is stored on a variable and then the data processing is done to extract the relevant features and are stored on a json file. Next the data is sent to a web app for visual analytics using Mapbox and Highcharts.

MapBox is an open-source mapping platform. It is designed for custom designed maps. The APIs and SDKs are the basic blocks to integrate location into any mobile or web app. Highcharts is a software library for creating charting written in pure JavaScript, first released in 2009. It is used to build interactive maps to display metrics, election results or information related to geography. It is ideal for standalone applications or in dashboards.

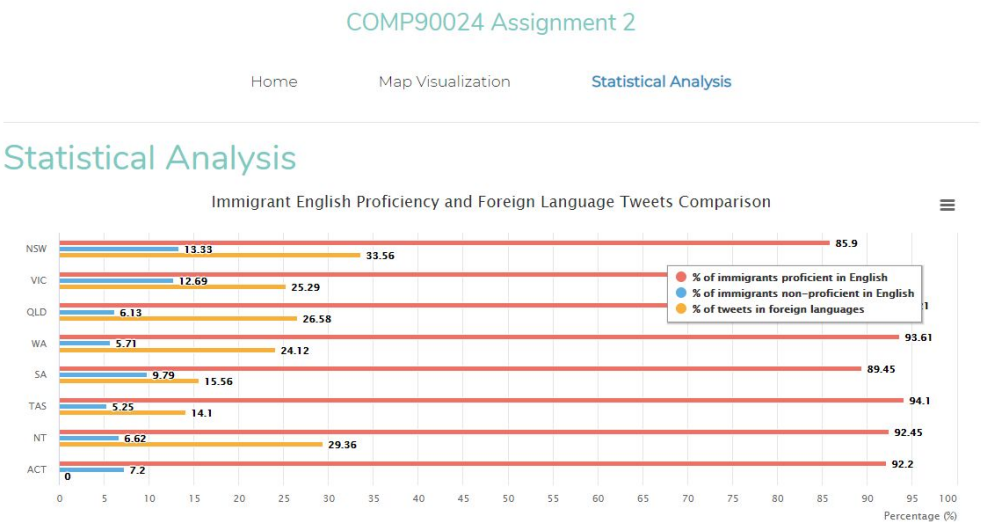
3.4 Web App Design

The front-end of the application was developed using HTML, CSS, JavaScript and jQuery technologies. No predefined template was used. The home page of web app is displayed below:

Mapbox Studio and Mapbox GL JS were used to render the map.



Highcharts was used to display the bar graph under the Statistical Analysis section.



3.4 Security and Fault Tolerance

Security and fault tolerance are essential attributes of any production level application. Security ensures that malicious outside entities do not affect the functioning of the system, while fault tolerance allows for its seamless operation for end users, despite the failure of nodes.

3.4.1 Security

At a high level, the system that has been developed depends on the security attributes and capabilities of the NeCTAR Research Cloud. Such attributes include the security from unwanted access by potential infiltrators caused by exposure to the internet, as all UME instances are only accessible through an IP address associated with The University of Melbourne. On top of this, instances are protected by login security groups, preventing any unwanted access to system infrastructure. With this in mind, small measures were implemented in order to ensure the security of the system. This includes limiting the instances hosting CouchDB to external traffic exclusively from port 5894. This is the default for CouchDB.

3.4.2 Fault Tolerance

As outlined above, the system architecture relies on a clustered formation, allowing fault tolerance in the case of a single node failing. This allows applications, such as CouchDB to continue running in the case of partial failure, with minimal effect on end users. In regards to the harvesting of tweets, the tweet harvester is deployed on each of the instances, with each iteration harvesting from a different geographic area. This allows for the continued harvesting of tweets even in the event of a failed node, while also ensuring that duplicate tweets are not stored in the database. Duplication of tweets can have a detrimental effect on storage space, as well as the consequent analysis conducted.

4. Automated Deployment - Ansible

Ansible is a highly efficient way of deploying system infrastructure through code. It gives the power to deploy multi-tier applications reliably and consistently, all from one common framework.

4.1 Advantages of Ansible

- **Reliable and Repeatable**

  Ansible allows us to write 'Playbooks' that are descriptions of the system's desired state under source code management. Ansible does the hard work of getting the systems to that state no matter what state they are at. Playbooks make installations, upgrades, and day-to-day management repeatable and reliable.

- **The process to write and maintain is easy**

  Playbooks are simple to handle. Users become more efficient with Ansible after only a few hours. Ansible uses the same tools likely already used daily, and playbooks written in YAML, a human readable language, so they are easy to comprehend.

- **Agentless design**

  Ansible introduced into the environment without any bootstrapping of remote systems or additional opening ports. Eliminates the management, but system resource utilization is also dramatically improved.

- **Downtime is rare**

  Ansible dictates zero downtime rolling updates trivially, ensuring the production of applications without the users knowledge.

4.2 Ansible Deployment

At a high level, all system infrastructure for the project is instantiated through running the below shell command:

```
1  #!/bin/bash
2
3  . ./unimelb-comp90024-2020-grp-26-openrc.sh; ansible-playbook --ask-become-pass playbook.yml -i inventory/hosts.ini
```

By running this script on the command line interface, the playbook.yaml (detailed below) playbook is called, which allows Ansible to deploy the predefined instances on the NeCTAR IaaS platform.

```
1  - hosts: localhost
2    vars_files:
3      - variables/vars.yml
4    gather_facts: true
5    roles:
6      - role: common
7      - role: show-images
8      - role: create-volumes
9      - role: create-security-groups
10     - role: create-instances
```

Ansible works on the premise of predefined roles, which are located in the 'roles' directory.
For this project, five roles were defined as outlined in the above screenshot. The first role, 'common' installs dependencies on the local machine (localhost), including pip and openstacksdk (to allow for programmatic interaction with NeCTAR). Once this is complete, the second, third and fourth ranking roles, 'show-images', 'create-volumes', and 'create-security-groups', are responsible for retrieving instance images, mounting volumes and initiating security groups. The final role, 'create-instances' then instantiates the instances, *db-server1, db-server2, analysis-server* and *web server*. Once deployed, the servers are able to be accessed via ssh using their respective IP addresses.

The open-ports.yml playbook is then called to open certain ports on each server to allow for intercommunication, as is required for a clustered deployment of CouchDB and correspondence with the web server. Security groups and their respective rules are also created, and then applied to each instance. Similarly, the cluster.yml playbook is called to initiate the CouchDB cluster.

```
1
2  - hosts: all
3    gather_facts: true
4
5    roles:
6      - role: common
7      - role: mount-volume
8      - role: create-couchdb
```

As above, common is called initially to add the proxy settings. ‘Mount-volume’ then ensures that all couch-db servers are mounted with a file system, upon which CouchDB and its dependencies will be installed. The ‘create-couchdb’ role then installs CouchDB, and initiates a cluster of three nodes, using a binding address, port, username and password on the *db-server1*, *db-server2* and *analysis-server*.

```
29     - name: Install Libraries
30       become: true
31       command: pip3 install {{ item }}
32       with_items:
33         - tweepy
34         - couchdb
35
36     - name: Run Harvester
37       become: true
38       raw: nohup python3 tweet_harvester.py 1 > /dev/null 2 > /dev/null &
```

Harvester playbook once called, adds all the relevant dependencies required to run the Twitter Harvester code and then runs the Twitter Harvester code as a background process for all the four instances so they can harvest tweets simultaneously. Similarly, WebServer playbook installs proxy settings and all the relevant dependencies, like flask, on our *web-server* instance to then run our application as a background process. This eliminates our need to manually connect to the machines using SSH and running the codes. Also the code will keep running in the background even if we disconnect from the *web-server* instance.

5. Data Analysis

Team 26 decided to focus on the relationship between english proficiency in immigrant populations and the language used by twitter users in different geographic areas. Australia boasts a highly multicultural society, lending to a variety of languages and dialects spoken by its immigrant population. The proposed research focuses on the occurrence in each Australian state, where tweets are in a foreign language and allow one to draw a correlation to the English language proficiency of immigrants in that state. Thus the analysis provides us with adequate knowledge to decide which state in Australia has maximum foreign language tweets with respect to an immigrant's English language proficiency. It would be expected that prominence of immigrants with low english proficiency will lend to higher levels of tweets in a foreign language in a given state.

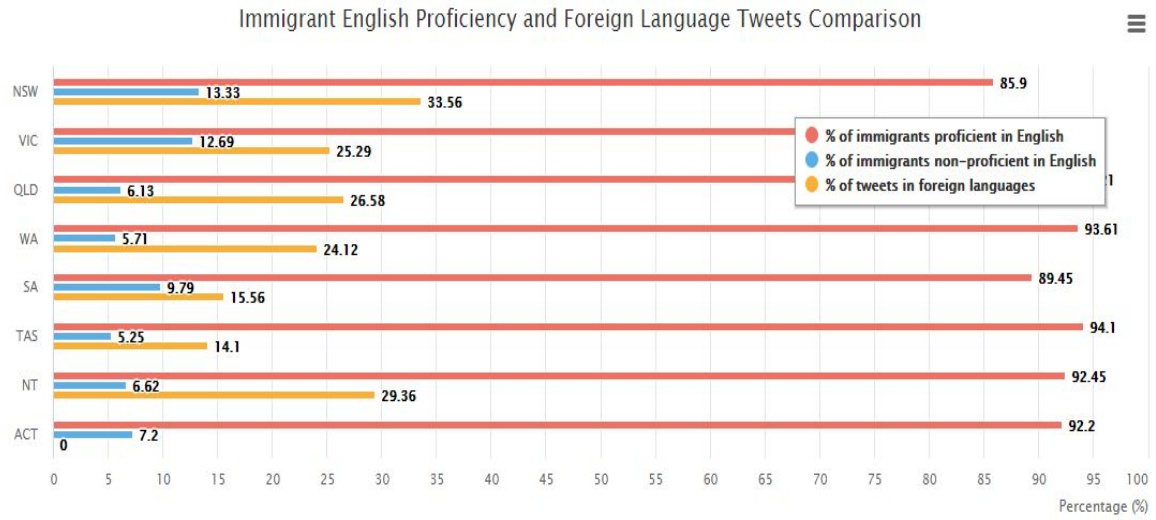
5.1 Findings in ABS - Data by Region - Persons Born Overseas (GCCSA) 2011-2016

The ABS dataset offers statistics on the english proficiency of the immigrant population of each state, recorded between 2011 and 2016. Generally, the english proficiency of immigrants in Australia appears to be high, with the highest level of low english proficiency amongst an immigrant population being just 13.33% in NSW. This would lead to an assumption that, overall, the number of tweets in a foreign language in Australia would be low.

5.2 Findings in Tweet Analysis

As displayed in the figure below, there is a slight correlation between the english proficiency of an immigrant population in a given state, and the number of tweets in a foreign language. As seen in NSW, 33.56% of tweets were in a foreign language, the highest rate from any state, which reflects the low english proficiency of its immigrant population which is only 85.9% proficient.

Statistical Analysis



This relationship is echoed in Victoria, although the same is not reflected in Queensland. Queensland has a relatively high recorded rate of 26.58% tweets in a foreign language, yet its immigrant population's proficiency in english is also high at nearly 94%. This appears to show a weak correlation between the two variables. The relevant statistics for each state are detailed below:

New South Wales

- Total tweets: *1326*
- Tweets in foreign language: *445*
- Most used foreign language for tweets: *Indonesian (201 tweets)*
- Immigrant population: *2064898*

Victoria

- Total tweets: *3152*
- Tweets in foreign language: *797*
- Most used foreign language for tweets: *French (124 tweets)*
- Immigrant population: *1673222*

South Australia

- Total tweets: *1800*
- Tweets in foreign language: *280*
- Most used foreign language for tweets: *Indonesian (71 tweets)*
- Immigrant population: *382682*

Queensland

- Total tweets: *316*
- Tweets in foreign language: *84*
- Most used foreign language for tweets: *Spanish (32 tweets)*
- Immigrant population: *1010596*

Western Australia

- Total tweets: *5016*
- Tweets in foreign language: *1210*
- Most used foreign language for tweets: *Nepali (115 tweets)*
- Immigrant population: *794686*

Northern Territory

- Total tweets: *327*
- Tweets in foreign language: *96*
- Most used foreign language for tweets: *Tagalog (34 tweets)*
- Immigrant population: *44852*

It can be concluded however, that with Australia's high level of english proficiency amongst its immigrant population, the preferred language when it comes to Twitter is English. This would lend to the low number of harvested tweets that were authored in a foreign language.

6. Project Challenges and Team Outline

6.1 Project Challenges

- **Tweet Harvesting:** The project faced issues concerning collecting the vast number of Tweets required for the analysis. The tweets collection was indeed time consuming and elaborate. Moreover, streaming down to only tweets within Australia and Geotagged tweets an uphill task.
- **Ansible:** There was a steep learning curve when having to write the Ansible Playbook. No member of the team had strong familiarity with Ansible, let alone YAML. This was overcome, with a successful final product.
- **Analysis:** The other major issue was related to the hypothesis that a Twitter user may tweet in a foreign language not because they are not proficient in English and are an immigrant, but it is that the user may prefer their native language. This type of filtering is hard to achieve. Yet, with the help of AURIN data and Map Reduce function we were able to draw correlations due to features like English language proficiency of an immigrant.
- **Virtual Group Work:** A major challenge was the difficulty in working in a group using only virtual mechanisms. There was an attempt to implement a Trello board to manage workflow, but the challenge was only slightly mitigated by this solution. The primary

mechanism for communication was Facebook Messenger, where the team would have 'standup' before the start of a session of work.

- **Dividing Work:** The group was faced with the initial challenge of how to split work between the team. It was decided that part of the team focus on the system infrastructure, the other on data analysis and tweet collection, and the remaining on the web user interface.

6.2 Team Contribution

Each team member worked diligently towards the realization of the system and the tasks undertaken by each member is summarized below.

Team members	**Tasks**
Ariel Kark	Data Analysis Historical Tweet Ingestion UI Backend Report Group Management
Charu Smita Singh	Ansible CouchDB Implementation Web App Design and Development UI Backend Part of report
Hina Bagai	Ansible CouchDB and Cluster Implementation Map Reduce Error Handling System Architecture

	Data Analysis UI Backend Part of report
Raksha Ravi	Tweet Harvester Part of report Presentation

7. Conclusion

The completion of this project allowed for the implementation of a functioning and scalable cloud-based application, dedicated to providing data driven insight to Australian society through the analysis of Twitter data.

The system is scalable as nodes can be added to the CouchDB cluster with ease and on a needs basis, should more data be required to be manipulated or stored. Together with this, more tweet harvesters can be instantiated on additional nodes to increase the number of tweets collected. The demonstrated system architecture is highly efficient, as infrastructure is deployed on a needs basis, preventing the impact associated with the high cost of cloud infrastructure.

As outlined fault tolerance and the availability of a system are essential attributes of any production level application. The clustered architecture of the CouchDB nodes assured this requirement, preventing the impact of failed nodes for end users.

Finally, the system infrastructure was insatinated through code, a requirement that enables the benefits of automated system deployment through Ansible. Infrastructure as code allows for fast and efficient deployment of production level applications.

Overall, Group 26 grew an appreciation for cloud and cluster computing, while understanding its endless applications in the space of Big Data and analysis.

8. References

[1] Mapbox Studio, https://www.mapbox.com/mapbox-studio/

[2] Mapbox Chloropeth Studio https://docs.mapbox.com/help/tutorials/choropleth-studio-gl-pt-1/

[3] Anon., n.d. Government of the Commonwealth of Australia - Australian Bureau of Statistics, (2019): ABS - Data by Region - Persons Born Overseas (GCCSA) 2011-2016; accessed from AURIN on 05/2020.. S.l.:s.n.

[4] Cesare Pautasso, O. Z. F. L., 2008. RESTful Web Services vs. “Big” Web Services:Making the Right Architectural Decision. Proceedings of the 17th International Conference on World Wide Web.