COMP90024-Cluster and Cloud Computing
Assignment 2 Report

Australian Political Tweets Analysis Prior to 2019 Federal Election

Group Number: 2
City: Melbourne

Group Members:

Kazi Abir Adnan <Student Id: 940406>
Ahmed Fahmin <Student Id: 926184>
Mohammad Nafis Ul Islam <Student Id: 926190>
Daniel Gil <Student Id: 905923>
Kun Chen <Student Id: 965513>

Abstract

This report describes the development of a cloud-based solution to analyze political tweets regarding the 2019 federal election of Australia, exploiting the multitude of virtual machines hosted on cloud. This system specifically performs data analysis of tweets of a fixed time period generated from Australia about political parties and their leaders, validating the facts using the last federal election result of 2016 in Australia. Alongside, this emphasizes automated deployment capability, orchestrating the set up of all required and necessary softwares on virtual machines supported by Nectar to make a ready-to-run system without any manual involvement. The whole project is an end-to-end system which starts from the scratch of setting up the cloud system from providers followed by collecting data and visualizing the analysis results on a web application.

Contents

1 Introduction

Twitter is a social network where people come to discover what is happening around the world and exchange their opinion in their own networks. Over 500 million tweets are being sent each along with various meta-data. These tweets cover various number of topics and it is a good source of tracking people's opinions regarding current affairs. This project focuses on the tweets mentioning the political parties and their leaders with regards to the 2019 Federal election of Australia. We have collected tweets of last 3 weeks, starting from April 27, 2019; to keep our scope of analysis to the closest possible time of the election. Our aim is to collect the data about what people of Australia feel about the parties, leaders and the election; and how it correlates with the result of last federal election. This research provides insight into the usage of twitter data and its applicability in analyzing facts, and also validating those with Aurin data. This project work develops an end-to-end solution of a cloud-based system, starting from setting up the architecture to the application layer. The volume of the dataset is fairly big to process with a lot of variety and data is coming with a high streaming velocity. So, accessing, manipulating or searching this amount of data makes it a Big Data problem which has to be handled appropriately with an efficient cloud solution.

The rest of this report is organized as follows. In Section: 2, we give information about some related works regarding social network data analysis. In Section: 3, we define our problem scope and give a summarized overview of our approach to the solution. Next, in Section: 4, we define our system architecture and its design. After that, in Section: 5, we give a very brief description of the technologies and tools, we used for developing our system. Later, in Section: 6, we discuss in detail, about how we have harvested the tweets. Next, in Section: 7, we discuss about different functionalities offered by our system. After that, in Section: 8, we give our data analysis report on various scenarios, which also covers some of the deadly sins criteria and its validation using AURIN dataset (required as per the project specification). Next, in Section: 9 and Section: 10, we give discussed about the UniMelb Research Cloud and our error handling approaches of the system respectively. Later in Section: 11, we provide a user guideline about how to use the system. Finally, we discuss about the limitations and future scope of the system in Section: 12 and following that, give some concluding remarks in Section: 13.

2 Related Work

Now a days, social media platform is a massive source of various data, which are being used for different researches, related to real life scenarios. Moreover, as they offer huge amount of data, there has been a major focus on social media, to perform researches, based on big data analytics. Among many social media platforms, Twitter has much popularity for this kind of researches, because of its well defined and available APIs. There have been lots of works [1–4] related to sentiment analysis using twitter data, which mainly focus on figuring out different emotions of people through their tweets and related contexts. But, there haven't been much relevant works on more sophisticated human natures and their personalities like greed, lust, lazyness etc. Schwartz et al. [5] has used data from Facebook messages to validate people's personality, based on their age and gender. In Australia, the team of AURIN[1] has been continuously working on data extraction from various sources including social media platforms, and has many results incorporating data, related to social indicators, economic activity, housing, health, transport etc. They are the baseline for validating our findings and analysis in this project.

[1]`https://aurin.org.au/`

3 Problem Scope

3.1 General Objective

The objective of this project is to explore the 7 deadly sins[2] through social media analytics and compare it with datasets from AURIN. To achieve this objective, we need to harvest tweets from different cities of Australia and create different types of data analysis scenarios to establish a story, that can create a relation between people's lives and one or more deadly sins. Then we need to validate these claims through datasets from AURIN platform.

To develop this system, we are required to implement a Cloud-based solution, that will utilize a number of virtual machines (VMs), available through the UniMelb Research Cloud[3], in order to harvest tweets using Twitter APIs. There can be multiple instances of the application running on the Cloud, together with CouchDB database, that will contain the collection of harvested tweets. The database itself could be running on a single node or in a cluster setup, but removal of duplicate tweets needs to be assured. Finally, to view the results of the analysis and scenarios, a front-end web application is required.

3.2 Our Story Overview

The focus of our story is to explore the current popularity of political parties and their leaders; and to figure out the feeling and sentiment of Australian citizens for the upcoming federal election on May 18, 2019; through Twitter data analysis. The aim is to collect people's tweets mentioning the political parties, the leaders, the popular election hashtags and what they are talking about. Our hypothesis is, this should be the reflection of current popularity (in both positive and negative senses) of the parties, their leaders and people's general opinion or sentiment. The overall aim of our story is to find which party people like or hate and correlate with the spatial claims using the previous election voting results from AURIN. We will also justify why twitter data can be a fair representative of popularity of political parties across Australia at a fixed period of time.

4 System Architecture and Design

4.1 Architecture

Our system architecture is shown on Fig. 1. We elaborate different parts of this architecture in the following sub-sections.

4.1.1 Back-End

Our back-end comprises of several components. They are the CouchDB, the Twitter Harvester, RabbitMQ and the restful API Server.

[2]https://en.wikipedia.org/wiki/Seven_deadly_sins

[3]https://research.unimelb.edu.au/infrastructure/research-platform-services/services/research-cloud

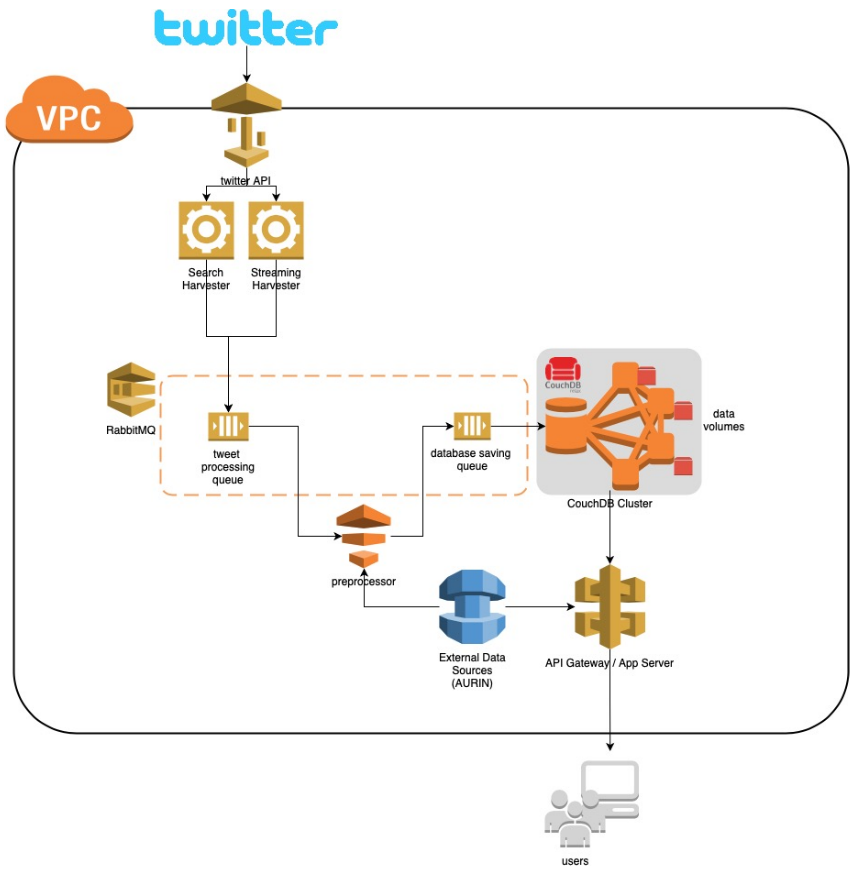


Figure 1: System Architecture

4.1.1.1 CouchDB

Volume, velocity and variety of data are key concerns in social media analytics. In this context, CouchDB delivered scalability, reliability and fault tolerance to support a non-relational model, based on text, captured from tweets. CouchDB, as document-oriented DBMS offered an efficient way to store collections of documents related to the followings.

1. **Tweets:** Raw tweets collected using Twitter APIs.
2. **Users:** Entities of interest extracted from the data.
3. **Configurations:** Special configuration data needed to use harvester.

The system can be used in single or a clustered setup to support high availability. The configuration, by default for each database are:

1. Number of replicas: 3
2. Number of shards: 8

In this configuration, the system should be able to scale up to 24 nodes and provide support in case of at most 2 nodes failure. With this cluster design, all nodes answer requests at the same time and sharding is done at every node.

The interaction between the system and the database cluster is managed by default, through a node, selected dynamically from deployment and monitoring. A load-balancer is not included as part of the architecture but considered as future implementation to increase high availability.

In addition, CouchDB performs MapReduce Algorithms to use in views and we used this to calculate Hashtag counts, Party counts, leader mentions, Aggregate tweet locations, Aggregate parties, Aggregate keywords in tweets related to parties etc.

4.1.1.2 Twitter Harvester

The purpose of the harvester is to collect data from twitter. Specifically, the data is captured using Twitter API as follows.

- **Search-Tweets:** The Search API is used to find historical tweets published in the past 9 days. This corresponds to the *standard/public* set of twitter APIs.
- **Real-time Tweets:** Real-time tweets use only one filter rule (only one filter used per connection). This also corresponds to the *standard/public* set of twitter APIs.

4.1.1.3 App Server - RESTful API

The API layer is delivering all the contents from the databases to the user through a simple web site. The system provides a list of endpoints to get the data for predefined scenarios e.g. get opinion (sentiment) polarity from a database view.

4.1.2 Front-End

The system provides a single-page user interface with minimal interaction functionality. The technology stack consists of HTML+CSS+JavaScript libraries supported by Leaflet, Open-StreetData and Google. Fig. 2 shows how our front end interacts with different technology.

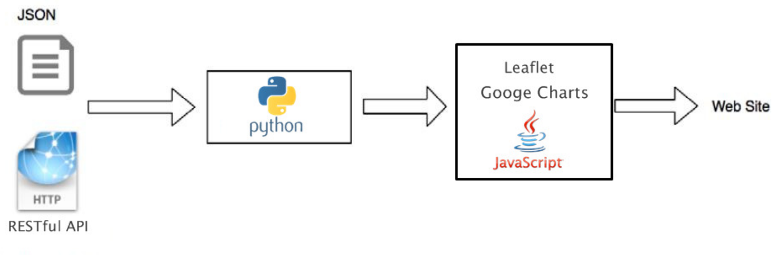


Figure 2: Visualization process.

4.1.3 Reason to Choose this Architecture

We have used two instances for database cluster, one instance for both tweet harvester with queue service and the application server. The last one is left to demonstrate the scalability. We have used two instances for the cluster to ensure availability of data. We have used a standalone server to support the web application.This will provide the HTTP request response from client side. The same instance is launched to collect the streaming and historical tweets, which uses threading process internally. The same instance also supports the queue service which is used to synchronize the harvesting process.

4.2 Security

The security of the cloud solution relies on the built-in security feature of the UniMelb Research cloud service. All instances are allocated with respective security groups with minimal inbound packets to instances and do not reveal all the ports. Ports for database and queue service were opened to allow access to the instance services. The system provides restful API's that can be used without any security credentials (open data), however, access to the system services are secured with credentials and managed during deployment time like the followings.

- OS user credentials are changed during instance creation.
- CouchDB admin accounts are configured during deployment and communicated with nodes in cluster.
- RabbitMQ access is configured during deployment and communicated with all relevant nodes (nodes with this service)

4.3 Fault Tolerance

- **CouchDB**: Problems related to an instance are isolated in the specific request. The designed solution can work with it as long as just one server is up. The cluster was tested to assess the system in terms of consistency, availability and partition tolerance. For example, with a 3-node configuration, we were able to test the system with 1 and 2 nodes down. If current VM instance, with a volume attached, goes wrong, that volume can be remounted to another VM instance.
- **RabbitMQ**: We deploy queue services dynamically in other hosts, however, we are not setting up a cluster for this service. If this service fails, our harvester at this point won't work. So, this one of the failure points we have in the system.
- **Harvester**: Whenever a harvester fails due to service failure, it will restart the process and read the configuration file again and retry forever, exploiting catching the relevant errors. We had a plan to add a monitor, which would always monitor and reinstate the services along with updating the configuration file. This way the harvester will read the updated file and can start a working process. However, this is left as a future improvement.
- **App Server**: App services are deployed dynamically in our system, however, their is no load balancer that can distribute the request on the go. We use Ansible scripts to put the configuration file in all relevant services, to update the system when a failure is present.

4.4 Scalability

Currently, the main components of our system support scalability. For CouchDB cluster, we can scale up to 24 nodes and every instance is being supported with extra storage. Automation using Ansible helps us to create new nodes and adds them to existing nodes. Scalability is provided in such a way, that we can expand volume if operational data store demands more. A limitation comes with the shards, since the system is not copying and balancing the shards, as for new databases, new shards are spread using new nodes.

Harvesters can be scaled as needed in different instances. Each harvester process is able to collect data from twitter and deliver it to the queue. A different thread in the harvesters will pick queued jobs, preprocess the tweets and queue it to persist in database.

4.5 Dynamic Deployment

Deployment of architecture is performed by Ansible. This was very helpful in setting up the database cluster to ensure CouchDB configurations.The scripts managed the connections of each instance to the cluster and automates the setting up of all the instances. It also installs all necessary software automatically in those instances. We used groups to tag our instances according to different services: RabbitMQ, CouchDB, application server and harvesters. The user can configure in a file `(host_vars/nectar.yml)` all the instances that are wanted to be deployed in the environment. One of the main points of our Ansible script is that it uses dynamic inventories from OpenStack sdk to have an updated and controlled inventory during execution. This allow us for example, to deploy the current IP address of our queue server (RabbitMQ) to the harvesters. We grouped playbooks and organized into roles according to similar features or scope, for example, some playbooks share one or two roles for ssh communication, install common packages, etc.The main automation tasks are done by different playbooks coordinated by the script run-nectar.sh. Groups defined for hosts allow us to execute each set of playbooks accordingly. We have used two main playbooks, one for setting up all the instances, and another for setting up all the software and other dependencies on those instances. We can run individually each playbook and are designed to be independent, however, a single script with several options makes the operation easier.

5 Technologies Used

- **Github:** It was used for collaborative development and version control mechanisms during the implementation process.
- **Nohup:** This was used to keep our processes running in the instances, even when we exit from ssh session. It keeps them running in background.
- **Visualization Tools**
  - **Javascript:** Simple javascript was used for the front-end development, for visualization.
  - **Pandas:** Pandas has been used to create necessary charts and graphs, based on analysis scenario, to include in the report.
  - **Google Charts:** We show different comparative scenarios in our front-end, using google charts.
  - **Leaflet:** It was used to show the map on the web application and exploit different functionalities related to it.
- **Ansible:** It was used for the automation of setups and the whole system configuration. The playbooks were used for setting up all the servers. A modular and simple approach has been used to write the playbooks, to reuse them in setting up of similar servers.
- **Docker:** There was no major use for this. We only had an exploratory docker image, which was used in our system for development purpose and experimenting.
- **Flask:** It served the main purpose of connecting the the back-end with the front-end. It was mainly used to fetch results of the analysis, which was finally shown on the web application.

- **Shapely Lib:** This library was used used mainly for the twitter preprocessing services. It helped with merging geoJSONs, calculating distance to a polygon, checking whether a point lies within a polygon or not, figuring out centroids etc.
- **CouchDB:** This was the only database system used in this development process. For MapReduce capabilities, we used the CouchDB's built-in implementations for data aggregation.
- **RabbitMQ:** It was used as a messaging broker, which acts an intermediary for messaging, to send and receive messages between harvester, preprocessor and database saving operations.
- **Tweepy:** It was used for collecting tweets.
- **Yaml:** It was used to define the configuration file for the application.
- **NLTK:** This natural language processing library was used to process the tweets and identify the sentiments.

6 Tweets Harvesting

Tweets are the center point of our overall cloud based system. The main goal of the research started with the idea of finding out spatial popularity of political parties in state and city level of Australia, aiming to collect tweets for a fixed time period. As the election is on May 18, 2019, we collected the prior three weeks' tweets to do the analysis, starting from 27th of April, 2019.

The total eligible voter of Australia is over 16 million for 2019 federal election and over 13 million people voted in last election. So to do a fair analysis and to find a correlation between the tweets and voting result, we need a good representative number of tweets to compare the popularity. Now, the first task is to determine how to collect relevant tweets regarding this upcoming election. To do this, we set predefined set of keywords which can be passed to Twitter API and get relevant tweets. The keywords are comprised of the name of the political parties, their leaders along with the Twitter screen name. We have also used the top trending hashtags of this federal election such as *'auspol', 'ausvotes', 'AusVotes19', 'ausvote2019', 'auspol2019', 'ausvotes2019', 'ausvotes19'* etc. to collect people's opinion regarding this election.

We have divided the harvesting process into three parts. It starts with collecting tweets followed by preprocessing and afterwards saving the processed tweets to the database. The heart of our harvesting process is the queuing service, which makes all the processes independent and separable. Each process performs its own operation independently and publishes the result to a queue service, where the following processes listen to.

6.1 Collecting tweets

Twitter offers free public APIs that allow developers to collect streaming and historical tweets with limitations and different configurable parameters. For this study, we have used Streaming API and Search API of Twitter to collect adequate, relevant and recent tweets across Australia for the upcoming election. Whenever a harvester is collecting a new tweet, it publishes the tweet to a specific queue, running at a port in the configured server, to perform the next step. The class that collect tweets from twitter is the *Harvester* in our project and the job of it is to download a tweet and send it to a queue. We have created three harvester which inherits this harvester class. However, we have only used two of those classes. Our study is all about recent tweets, so we did not need to go to a user's timeline to collect historical tweets. It is also worth to mention that, each of our harvesters uses Tweepy python library to connect to Twitter's restful API's and collect the tweets.

6.1.1 Twitter Stream Harvester

Twitter Stream Harvester uses Tweepy python library to connect to Twitter Streaming API and fetch real time data from Twitter. We have used some predefined tracking keywords to filter the relevant tweets for our study purpose. The keywords are all related to party name, their leader, party twitter account, leader twitter account and some predefined hashtags. The streaming API supports several filtering options with 'OR' logical operations. For example, if we want to collect tweet with 'scottmorrison' hashtag in 'Australia', it will return tweets of either with the hashtag or any other tweet, which is generated inside Australia. For this specific study, we only used 'tracking' filtering to collect relevant tweets. We left the location filtering task for the preprocessing part.

6.1.2 Twitter Search API Harvester

Twitter search API can collect historical tweets over time. For our study, to ensure the adequate collection of tweets and overcome the limitation of streaming harvester, we have used this search API to go to previous 9 days time-frame, and collect relevant tweets using our predefined keywords from any place in Australia. This API has option to collect tweets relevant to keywords, along with geo-location. So, we collected the center point (latitude, longitude) of Australia and calculated a fair radius to get the tweets only from Australia. We will talk about the limitations of this harvester in Section: 12.

6.2 Preprocessing

Our Preprocessor collects downloaded tweets from a queue and starts to process it before saving it to the database. This preprocessor is the heart of our data analytics system. The prime duty of the preprocessor is not only to collect relevant tweets, but also extracting other information which will be used at the stage of data analysis from each tweet. Our main focus was to extract information at this stage, so that we do not need to go to the database again to process it.

Furthermore, another focus was to use the CouchDB's views efficiently, so that no further processing needed. After preprocessing, this publishes the processed tweet to a queue for database to save it. The main functionality of our preprocessor are discussed in the following sections.

6.2.1 Location filtering

This study aims to analyze the popularity of political parties in Twitter at city/county and state level. So, we needed geo-located tweets from the Twitter. Unfortunately, less than 1% of the tweets we collected, had exact geo-location and 3% had bounding box's. From the API documentation we have seen that, location of the tweet can be obtained from *'coordinate','geo','place'* or from user location field. As the first 3 fields mentioned above were 'null' for the political tweets, we defined our own location filtering function mapping for this study purpose. Due to this reason, we collected the Electoral Division names from AURIN dataset and collected the city/county and state names from the latitude and longitude using reverse geo-coding. The process of tagging location to a tweet is mentioned below.

1. Using exact geo-location: If we get the exact geo-location from ’coordinates’ or ’geo’ (not applicable for new tweets) of the tweet, we find the city/county name by reverse geo-coding, and take the city/county, state and country field from the response.

2. Using bounding-box: If the previous option is not available, we check the ’place’ field to get the bounding box and collect the full location (from ’place’ field) and try to match it with our interested city/county names of Australia. If we find the match, then we take the tweet.

3. Using user location: This is the last option we choose to do the location filtering. In this case, if the name of the location of the user exactly with any of our interested area, we map this to our own location format.

It is worth to mention that, if a tweet can not be tagged to a location, we do not save it to the database.

6.2.2 Tagging Political Party

We got the registered party names from the Australian Electoral Commission, and found more than 20 parties. For each party we collected the party’s leader names and their twitter accounts. During preprocessing, we try to tag each tweet to at least one political party. If we can not tag, we do not consider the tweet and discard it. We tag a tweet to a party if the following conditions match.

1. The tweet contains any leader or party name. This should be an exact match of the names of the party or leader, we have in the list.

2. Tweet directly mentioning a leader or a party. If we find such tweet, we assign it to that party.

In this way, a tweet may be tagged to more than one party which is not an issue for our study. Our aim is to find how many people are talking about the parties regardless they like, hate or compare the party or criticizes. So, a tweet may talk about two parties, which is completely fine to measure the popularity. However, for some specific scenarios; for example to collect top negative keywords or sentiment regarding a party, we define the scope to only tweets which mentions a single party. Though it has few limitations, but worked fairly for this study purpose.

6.2.3 Keyword collecting

We collect processed keywords of tweets during the preprocessing stage for data analysis purpose. The Keywords we collected are processed by lower-casing, removing stopwords, lemmatizing, discarding party related words and discarding hashtags.

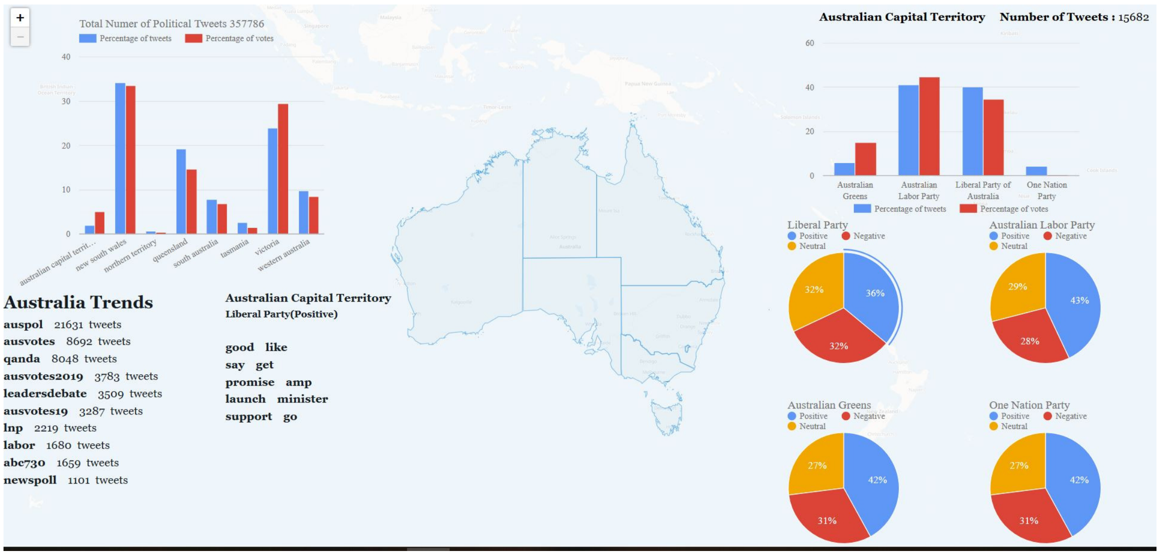
6.2.4 Sentiment tagging

For each tweet, we tag the sentiment of tweet using the Sentiment Analyzer provided by NLTK library of Python. If we provide the processed keywords to the Analyzer, it gives sentiment scores of ’Positive’, ’Negative’ or ’Neutral’ along with the intensity. We categorized the intensity to categorical values such as ’Strong’ or ’Moderate’ based on the variables provided by the library. If the ’compound’ field of the score is negative, then it belongs to ’Negative’ sentiment and otherwise as ’Positive’ sentiment. The intensity of sentiment is also measured by using the the absolute value. If it is more than 0.50 for any sentiment it is defined as ’Strong’.

6.3 Saving to Database

After preprocessing and filtering the preprocessor publishes tweets to a specific queue and our database saver monitors the queue to save it. As, we are using Apache CouchDB, which is a document oriented NoSQL database, it maintains a (’key’,’value’) structure. We know that the tweet id of a tweet is unique. To save a tweet to the database we have used the Tweet ID as key and the other properties of the tweet as value. So, in this way, if a tweet is already saved it will raise a conflict error and we catch that exception. In this way, we got rid of the duplicate tweets. However, for the purpose of our study we did not remove the retweets. Because, retweets are also a tweet from a user and it falls into the scope to measure the popularity of a party or a leader.

7 System Features



Australia Trends

auspol	21631 tweets
ausvotes	8692 tweets
qanda	8048 tweets
ausvotes2019	3783 tweets
leadersdebate	3509 tweets
ausvotes19	3287 tweets
lnp	2219 tweets
labor	1680 tweets
abc730	1659 tweets
newspoll	1101 tweets

Australian Capital Territory
Liberal Party(Positive)

good like
say get
promise amp
launch minister
support go

Figure 3: A snapshot of the web application.

The main purpose of our cloud solution is to provide insights of Twitter data and compare the result with the Aurin dataset that we provided. Fig.3 shows how we represent our analysis. Along with this main feature, we want to mention other key features that makes the system robust.

- **Automated Deployment:** By running Ansible script the whole cloud solution gets deployed and user can directly start to use the web interface instantly without any manual installation of softwares.

- **Scalability:** If any new instance is available in resources, one can add the instance to the existing cloud system. For example, If we need to install another harvester, we can just run an Ansible script which will add the new instance by deploying the harvester in the new machine.
- **Interactive data filtering :** Our web application supports some interactive filtering functionalities. However, a great portion of work has been left for future improvement.
- **Creating customized views from database:** It is possible to add new views from the CouchDB web interface to access the documents, which can be used for any new study purpose.
- **Configurable harvester :** The harvesters can be configured by providing the location in the configuration file without changing anything from the code level.
- **Configurable keyword list :** The Search API harvester can be configured to any location with any keywords from the configuration file.
- **Configurable layers :** If we need to change the database provider of the cloud system, our harvesting process can be configured with minimal changes. We just need to update the class which saves the tweets to the database, without touching any other part of the codes.
- **Adding more processing layer:** The processing layer of harvesting step can be configured with new processing layer. Suppose, if we add the topic modelling functionality in the system, we just need to call that from the processing layer.

8 Data Analysis

8.1 Dataset overview

8.1.1 Aurin Dataset

The Aurin dataset we used to validate our hypothesis, can be found in AURIN using this name: *'Federal Election Polling Booth Data (Point) 2016'*. This Polling booth data is from the 2016 federal election in Australia. Fig. 4 shows the result of the 2016 federal election.

Data for each individual party is given in three categories as total votes, swing and percentage. The vote counts represents the popularity of a party at a Electoral division in 2016. We have used only the vote counts of the parties in each city/county, Electoral division, state level. However, the tweets we collected which gives the location information mostly based on the state. So our final analysis focuses only on the state level. For simplicity we are only referring to the top partys' individual vote counts shown on Fig. 5.

8.1.2 Twitter Dataset

We have collected tweets of around 50,000 users starting from April 27, 2019. The total tweet count is over 400,000. Fig. 6 shows the distribution of top party and leader names in our collected tweets.

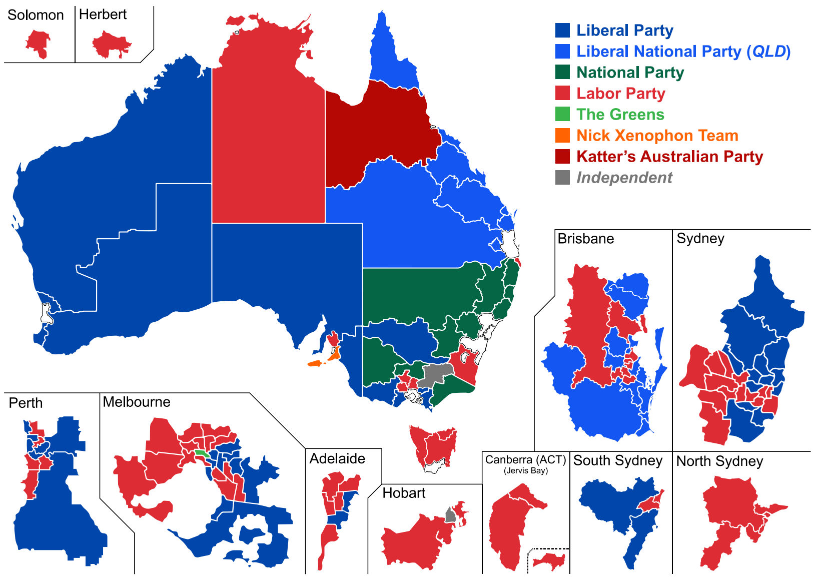


Figure 4: Result of Australian Election 2016.

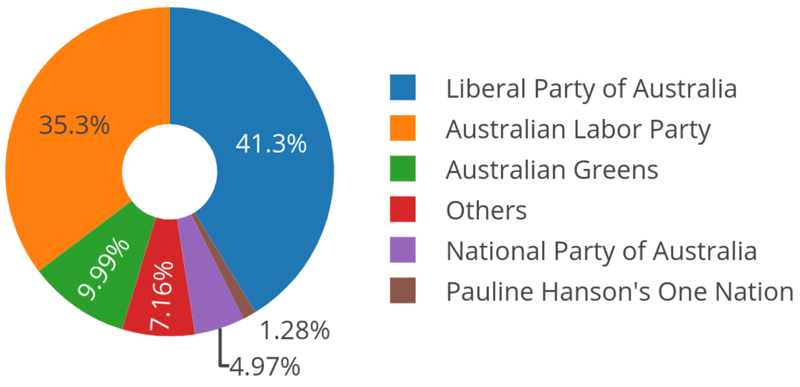


Figure 5: Total Votes Percentage by Party (Aurin data)

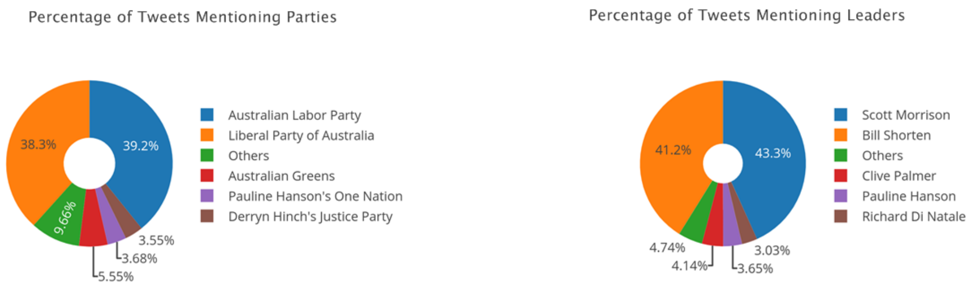


Figure 6: Tweets Mentioning Party and Leaders

To show the performance of our harvesters, we have depicted the count of tweets collected in each day on Fig. 7. The highest number of tweet collection was on May 8, 2019. On that day, our harvester collected over 70,000 tweets without any interruption which is worth mentioning. The reason for this peak is, on that day Liberal and Labor party leaders went on a head to head final debate.

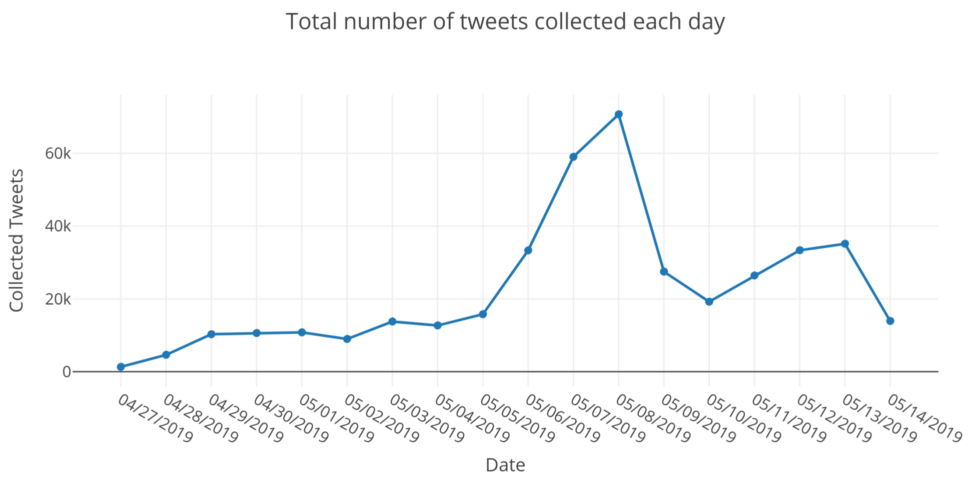


Figure 7: Total tweets collected per day

8.2 Distribution of tweets vs 2016 voting result

We have tried to establish a correlation between the tweet activity of users and the voting data of the last federal election, which was held in 2016. Firstly, we should notice that, this tweet activity does not consider the sentiment of the tweet. It only shows if the party is ’hot’ in each of the state. Considering an overall scenario, we can see from Fig. 8 that, the mostly populated states have more tweets alongside large number votes, comparing to other states. We can also see that the percentage of the tweets about the election in each state are pretty similar with the percentage of votes, showing that voters are getting active on social media during the time of election in every state. Moreover, people living in ACT and VIC have more concerns than other states as the percentage of the tweets in these two states are higher than the proportion of the voters.

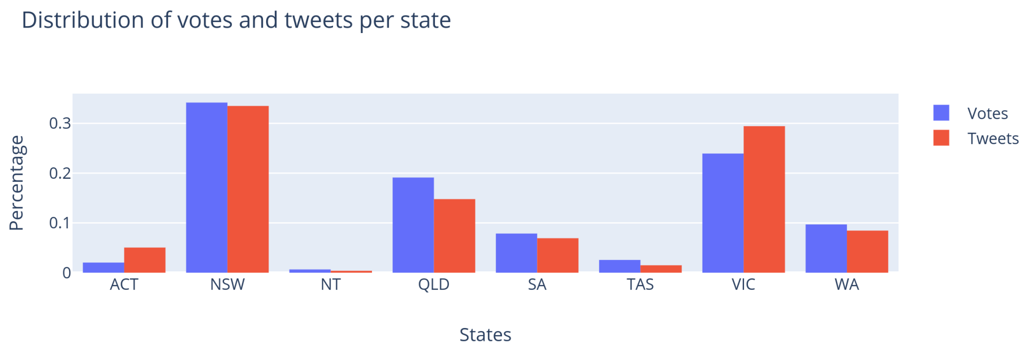


Figure 8: Distribution of Votes and Tweets Per State

8.3 Popularity of political parties vs 2016 voting result

Here, we try to show the popularity of different parties in different states. Almost all the parties have much popularity in the more populated sates like NSW and VIC, which is obvious. But there are some differences as well.

Fig. 9 shows that, Labor Party has more votes in NSW and the tweeter activity is also similar related to them. But, in case of VIC, even though Labor Party is more popular on twitter, they have gotten less votes, compared to NSW.

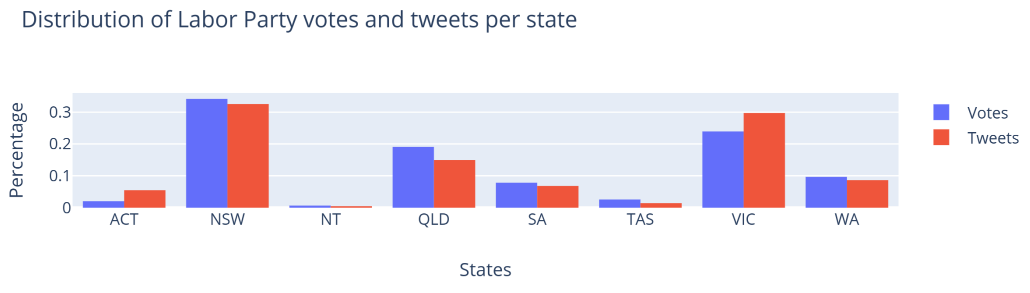


Figure 9: Distribution of Labor Party Votes and Tweets Per State

Fig. 10 shows similar kind of popularity for Australian Green Party in different states. Specially they are most popular in VIC, which is supported by both vote counts and tweet counts.

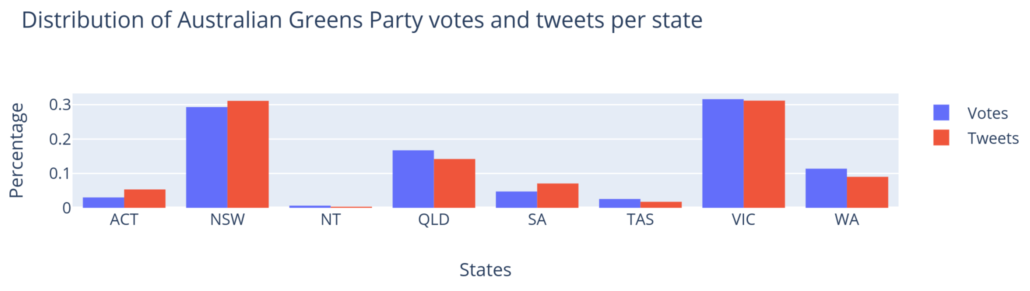


Figure 10: Distribution of Australian Green Party Votes and Tweets Per State

Fig. 11 shows these scenarios for Liberal party, which is pretty similar to Labor party. It is very much relatable as they are one of the top rivals of each other during election.

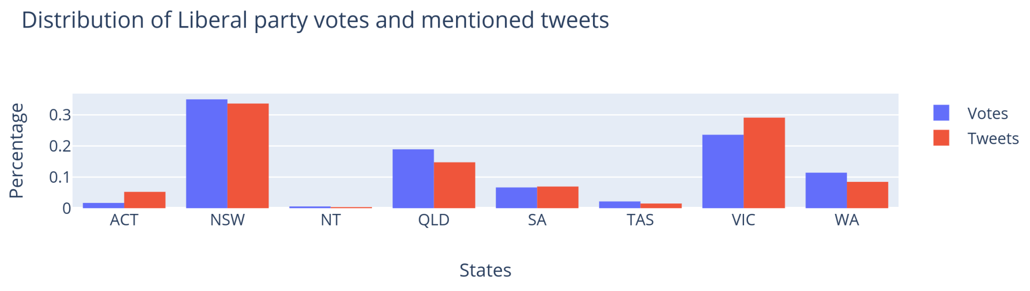


Figure 11: Distribution of Liberal Party Votes and Tweets Per State

Fig. 12 shows the distribution of the votes and tweets for Pauline Hanson One Nation. Their vote ratio is pretty distinguishable from others which is mainly based on QLD. One possible reason is that their headquarters is in QLD. Although they only got votes in NSW and QLD in 2016, the tweets we collected are from every state. This indicates that Pauline Hanson One Nation are looking forward to expand its influence in 2019 federal election.

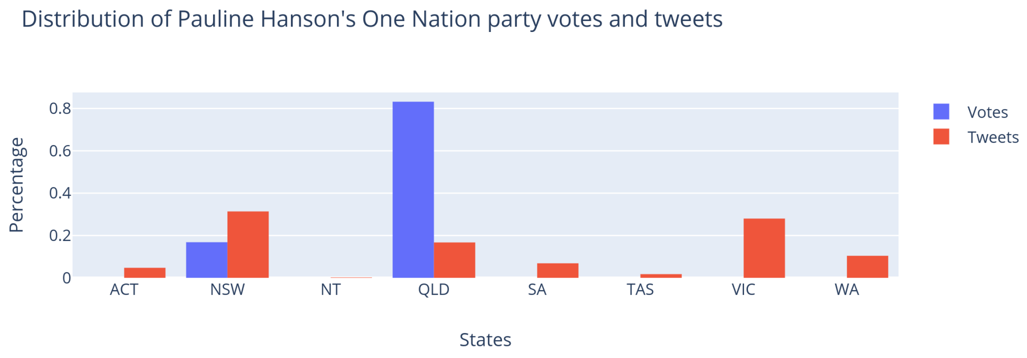


Figure 12: Distribution of Pauline Hanson's One Nation's Votes and Tweets Per State

If we look at the data at a higher level and compare the different parties in the same state, we can find that in NSW, the Labor Party and Liberal Party have a higher proportion of votes than the Green Party. However, both of them have a lower proportion of the tweets than that of votes which indicate they may have good confidence to win the NSW. And for the Green Party, the proportion of the tweets is higher than the votes in NSW which shows the ambition to get more votes in this state. Compared with the NSW, the data in VIC shows a different situation. The Green Party is more popular and got a high ratio of votes in VIC. But, the performance of the Labor Party and the Liberal Party were not as well as in NSW. With regards to the tweets, both of Labor Party and Liberal Party have larger proportion than the voting data of 2016, which shows the great efforts made by both of the parties, for the upcoming election.

8.4 Ratio of positive and negative tweets of political parties

In this section, we try to show some sentiment analysis of the harvested tweets related to different political parties.

From Fig. 13, we can see that people have given tweets related to Australian Green Party and Labor Party mostly in positive ways in Australian Capital Territory (ACT). On the other hand, Liberal Party has the most negative tweets. To validate this finding, we can see from Fig. 14 that, Liberal Party indeed got less votes than the Labor Party in ACT, despite having nearly the same amount of twitter activity. It is because most of tweets on Liberal Party were of negative sense. With regards to the Deadly Sins Scenarios, the tweets that mentioned about Labor Party in positive ways, may related to Greed and the Labor Party indeed got the most votes. In the other hand, the tweets about the Liberal Party in negative ways, may related to Wrath.

Fig. 15 shows some of the top negative words collected from tweets, related to Liberal Party, which also validates our negative tweet findings, because the words, such as lie, attack, rape etc. shown here are indeed of negative senses.

8.5 Supported Deadly Sins Scenarios

Our study covers two aspect of the deadly sins. Tweets that mention political parties and their leaders mean the party is popular. Now, this can be of either negative or positive sense. In

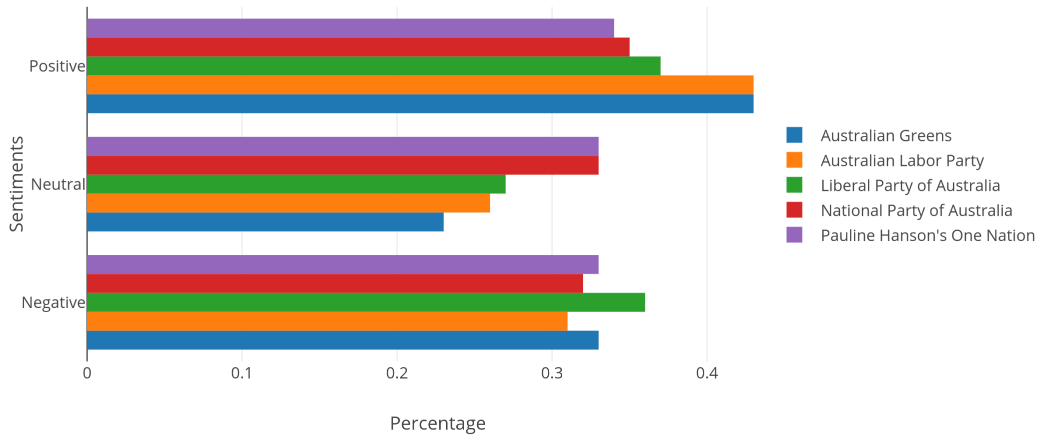


Figure 13: Sentiment of Tweets in Australian Capital Territory

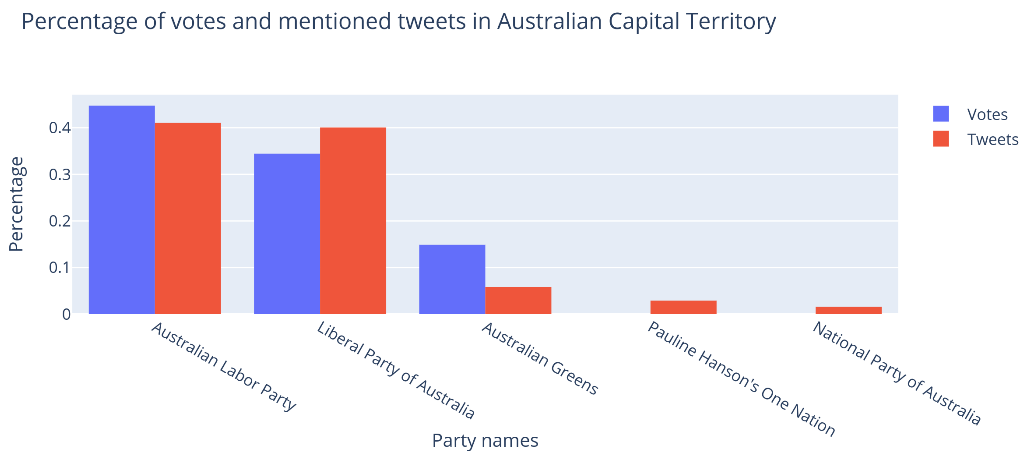


Figure 14: Percentage of Votes and Mentioned Tweets in ACT

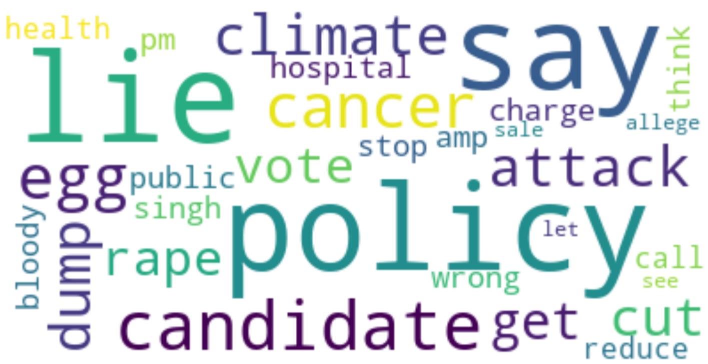


Figure 15: Negative Words in Tweets related to Liberal Party in ACT

our case, it falls into two sins categories.

1. **Greed:** People who wants the party to be in power or loves the party will post much more tweets on that party in positive sentiment. We define people's love or their desire for their supported party to be in power as *'greed'*. We get this by capturing the mentions of tweet. A mention to a party or a leader with a positive sentiment means, they like or support that party.

2. **Wrath:** Tweet that includes negative sentiment means people using negative words for that party, which can be defined as hatred to a party. We categorize this as a *'wrath'* emotion of the general people.

9 Discussion on UniMelb Research Cloud

9.1 Advantages

- **Automation of the System:** One of the main advantage of using this cloud based platform is the automation of the full infrastructure. The creation and deletion of instances can be done on the go, and it could be fully automated based on different scenarios.

- **Backup and Restoration:** There are features to take snapshot of volumes to provide backups. Volumes can also be transferred among projects too. Moreover, a volume can be detached from an instance, and then attach to a new instance.

- **Quick Launch through AMI:** To facilitate quick launch of instances, UniMelb Research Cloud has preconfigured AMIs (Amazon Machine Images). This provides a number of choices to use different Linux based distributions. Users can also create their own AMIs or use other AMIs, already configured for other researches. It saves a lot of time during environment setup.

- **Other Features:** It provides features like Network Management and Control, Heat based Orchestration mechanisms, similar kind of object stores as AWS S3 storage etc. UniMelb Research Cloud is based on Openstack, so, to facilitate different needs of researchers, it can increase its own space.

In a nutshell, UniMelb Research Cloud provides a great platform for researchers, who can utilize its distributed computing power and features, to solve complicated tasks.

9.2 Disadvantages

- **Speed Issue:** Compared to other commercial services like Azure, AWS, the running speed of UniMelb Research Cloud instances are pretty slow.

- **Resource Limitations:** Not being a commercial service platform, its resources are limited. We were given a limited quota on the number of instances and volumes to be used. So, it was a little difficult, when more distribution of workload was needed to speed up the processing.

- **System Support Issue:** This cloud platform has errors like other services. Anything can go wrong suddenly, and then we would need the assistance of the support team. Since it is not a commercial service, the response of the support team is not as fast as other commercial services.
- **Lack of Windows based Instance:** All the available AMIs were Linux based on this cloud platform. There was no windows based AMI.
- **Limited Bandwidth:** The network bandwidth is very limited in UniMelb Research Cloud. So, fetching data from CouchDB took longer than usual, because of the slow data transfer rate.

Tab. 1 summarizes these pros and cons of UniMelb Research Cloud.

UniMelb Research Cloud Discussion	
Advantage	**Disadvantage**
System Automation	Slow Instance
Backup and Restoration	Resource Limitation
AMI Quick Launch	Slow System Support Response
Compatibility Features	No Windows base AMI

Table 1: UniMelb Research Cloud PROS and CONS Summary

10 Error Handling

10.1 Deployment Errors

- Instance images were changed during the process of our installation. We needed to find another proper image of the OS to use.
- The flavor that we were using at the beginning was changed and that gave us some problem with the network connection, as it was not using the public IP's. We used Ansible to modify the network configuration in each hosts to allow the host to be connected to internet.
- We disabled the RSA authentication and enabled password authentication which led to an hacking incident. Eventually the instance was locked and we needed to delete the instance and couldn't use those keys.
- RabbitMQ uses different version of Erlang package, which we needed to download separately.
- Every database server attached a volume to store the data. If the node was down, we would be able to use the volume to create a new node for the cluster, and move the shard from the volume.

10.2 Harvesting Errors

In our designed solution, errors can occur at different stages of processing. Some of those errors and their handling approaches are as follows.

- **Required files not found:** There are few files which are necessary to run the harvester. In the beginning, we check if the required files are present. If not, we don't continue and print proper messages corresponding to the error.
- **CouchDB Database errors:** Database is one of the main parts our application. If the harvester can not create the database, then it raises 'PreconditionFailed' error and the system tries to use the existing one. If the error is raised by authentication issue, it prints a proper message and stops the application to progress further.
- **RabbitMQ connection:** Queuing service is one of the major requirements to run the harvester. If a service is not configured properly, the harvester can not proceed further and stops progress printing a relevant error message.
- **Duplicate Tweets:** Duplication is handled by using the document-oriented design feature of CouchDB, which uses key and value pair to ensure uniqueness. We have created a document in CouchDB using the tweet ID which is unique for each tweet. As, CouchDB does not allow to have more than one document with that ID, we do not have any duplicate tweets in the database. Whenever a duplicate tweet is tried to be saved in database, it raises 'ResourceConflictError' and the system catches the exception and ignores the tweet.
- **Tweepy Errors:** There are several errors that should be properly handled for continuous collection of tweets. Some of them are as follows.
  - **Incomplete Read Errors/ Protocol Errors:** This error raises when there is a high stream of tweets in a particular time. The connection is closed by Twitter if system can not consume data as fast as they are produced. Our system catches the error and continues to start streaming again.
  - **Rate Limit exceeding errors:** We have saved the Tweeter API credentials in the database. So, when an instance of the harvester is launched, this tries to lock a Tweeter credential from the database and if successful, launches the harvester. This ensures that, no two harvester is using the same credential at the same time. Moreover, Tweepy has an option to wait on rate limit exceeding, which has been used to avoid this error.
  - **Others:** To make sure the harvester never stops, we have also used 'retry count', 'retry delay' and 'retry errors' codes to avoid system failure. We have catched the exception of 'keyboard interruption' to stop the harvester application.

11 User Guideline

- Software Prerequisites
  1. Code (Github Repo) $ git clone `https://github.com/danielgil1/social_media_analytics.git`

2. Install Openstack python package Follow instructions according to your system here: `https://docs.openstack.org/python-openstackclient/latest/`
3. Install Ansible python package Follow instructions according to your system here: `https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html`

- Configuration and access files

1. Private key Generate the key in Nectar Dashboard using the key pairs option
2. Openstack RC file Download your file from Nectar Dashboard
3. Explicit use of OpenStack inventory script. The system provides a default file at /ansible/inventory Download the latest version of the OpenStack dynamic inventory script and make it executable
   $ wget `https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/openstack_inventory.py`
   $ chmod +x openstack_inventory.py
4. Update yaml with variables to allow dynamic inventory $ inventory/cloud-openstack-prod.yml
5. Nectar Cloud password
   Follow the instructions in the Nectar Dashboard and reset your password. Update your password in the configuration file. $ inventory/cloud-openstack-prod.yml

11.1 Deployment

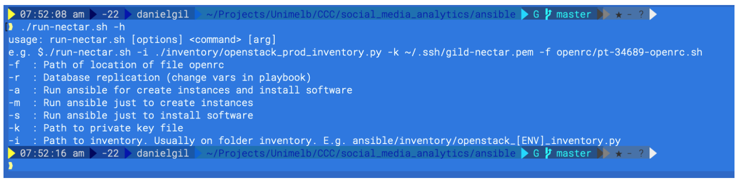
Once you have set up the configuration and access requirements you can deploy the system. You need to perform two steps:

1. Configure the environment you want to setup indicating the instances and services assigned to those instances.

   - Services available:
     - dbservers: Host will include database services.
     - appserver: Host will include application services (ResTFul API) and UI
     - rabbitmq: Host will include queue management services
     - harvester: Host will include harvester.
   - Modify these lines in the file $ `ansible/host_vars/nectar.yml`
     Include in the "instance_group" any service you want for the host.Note: For compatibility risks you DO NOT use the same host for database and queue service. An example is shown below: instances:
     - instance_name: 1
       instance_group: harvester,rabbitmq,appserver
       instance_security_group: ssh,rabbitmq
     - name: 2
       instance_group: dbservers
       instance_security_group: ssh,couchdb

  - instance_name: 3
    instance_group: dbservers
    instance_security_group: ssh,couchdb

2. Run $ run-nectar.sh -h to get familiarized with the options available. In Fig. 16 is provided an output:



```
 07:52:08 am  -22  danielgil  ~/Projects/Unimelb/CCC/social_media_analytics/ansible  G  master  ★ - ?
 ./run-nectar.sh -h
usage: run-nectar.sh [options] <command> [arg]
e.g. $./run-nectar.sh -i ./inventory/openstack_prod_inventory.py -k ~/.ssh/gild-nectar.pem -f openrc/pt-34689-openrc.sh
-f  : Path of location of file openrc
-r  : Database replication (change vars in playbook)
-a  : Run ansible for create instances and install software
-m  : Run ansible just to create instances
-s  : Run ansible just to install software
-k  : Path to private key file
-i  : Path to inventory. Usually on folder inventory. E.g. ansible/inventory/openstack_[ENV]_inventory.py
 07:52:16 am  -22  danielgil  ~/Projects/Unimelb/CCC/social_media_analytics/ansible  G  master  ★ - ?
```

Figure 16: Help option for script

3. `\protect\T1\textdollar./run-nectar.sh-i./inventory/openstack_prod_inventory.py-k~/.ssh/gild-nectar.pem-fopenrc/unimelb-comp90024-group-2-openrc.sh`

4. Confirm with Ansible any required information
   Ansible will ask for credentials to start any playbook. You will need to provide password to access Nectar Cloud (Follow the options to change your password in the Nectar Cloud, provide your password to access ssh and root password.
   Note: Instances created have by default credentials modified in deployment:
   user: ubuntu
   password: p01ss0n

11.2 Adding new node

If you have a new resource, just add it to the configuration file `\protect\T1\textdollaransible/host_vars/nectar.yml`

11.3 Running harvester independently

To run a harvester independently, one can just run the 'main.py' file in the harvester folder. This python file takes harvester type as an argument. If we need to run the search API harvester we need to provide `'api_search'` as an argument while running the 'main.py' file. The other parameters of harvesters is configurable and can be updated from the 'config.yaml' file in the root folder (eg. the geo boundary, the keyword list etc.). Keyboard interruption event has been catched to stop the harvesters.

11.4 Accessing the front end

We are hosting our web site on port 80 of the application server instance. If we run the URL from any browser, we can see the website. At the beginning, it shows a comparison between percentage of vote (from Aurin data) and Twitter data. It also shows total number of political tweets as well as trending political key words with their counts. If anyone clicks a particular state, it will show a comparison among top four political parties in that state. For each party, it shows the current popularity and previous year result. Besides it shows the sentiment of the tweets (positive, negative or neutral). If we click on any of those sentiments, it will show the top keywords of that selected sentiment, from tweets related to that party, in that state.

12 Limitations and Future Scope

12.1 Limitations in Tweet Collection

We could not collect data beyond a certain date due to the limitation of Twitter Search API. This API allows access to tweets of previous 9 days and does not allow to go beyond that using standard services. Our aim was to collect tweets from at least a month prior time period, which couldn't be achieved. Furthermore, there are limitations of hourly tweet rate collection though they are handled properly during coding, but still it's a limitation. The filtering option of Twitter API is an 'OR' operation, which leads to many unnecessary tweets. That is why we only downloaded tweet that matches the keyword for streaming API and left rest of the work to preprocessor to filter the location. It is worth to mention that, to the best of our knowledge, we found Twitter doesn't reveal the exact geo-location of sensitive tweets (example: political, government etc.) and it was a big limitation for this study.

Our harvesters uses threading to parallelize the harvesting process by dividing the downloading, preprocessing and database saving steps. We have implemented an MPI version of this harvester, but could not deploy due to resource issue. We could not find any flavor that allows to use 4 cores which was needed to implement the parallel version of harvester.

The harvester, using Search API, currently downloads a lot of duplicate tweets. This can be avoided by keeping track of already downloaded tweets and start from there in following harvesting time. Furthermore, a scheduled job would be perfect to harvest historical tweets for the study purpose, which can run daily and collect the tweets of that day.

12.2 Limitations in Preprocessing

We are processing every tweet, coming from harvester before trying to save to database. So, this leads to some unnecessary processing of tweets which are duplicates. Secondly, the political party tagging mechanism can be improved by learning a classification / clustering model, which will tag a tweet to party. The mapping of tweets to a city/ county/state is based on either 'place' or 'user location' attribute. User location is not a reliable resource of tweet, as that field is customized and open ended. Our assumption is that people are honest about revealing their actual location.

12.3 Database Limitations

Our Twitter database contains many fields which are not necessary for this study, but has its purpose for future scope. In future, after the federal election, we want to analyze the effect of twitter activities in predicting Federal election outcome and for that, any other Tweet fields might be interesting.

12.4 Harvester Limitations

The architecture of the harvester uses the queue service. So, it is possible to collect tweets, preprocess and save it to database in parallel fashion and with any number of those instances. However, we could not demonstrate this functionality of the harvester due to short period of time. Instead, we have used the threading in harvester to demonstrate the independence between the processes. In future, we can segregate it to independent processes and call all those processes from bash script.

13 Conclusion

Though we have few limitations in our cloud based solution, we were able to understand the feelings of people about the political parties or their leaders. This research also provides a good glimpse on how this system can be used by parties or any organization to evaluate the spatial political popularity in real-time, along with the sentiment of the people. The most advantageous functionality of it is that, anyone can see what people are talking about politics; what they like or dislike about political parties or their leaders. In the end, this research project has helped us to learn the steps of implementing end to end Big Data application and enabled us to experiment in developing a small scalable cloud system, utilizing UniMelb Research Cloud facility, document-oriented NoSQL database CouchDB, queuing service like RabbitMQ and automation tools like Ansible. The whole experience lead us to build a cloud solution, which starts from creating VM instances from the provider, to instantiate the analytic support for the user. Though the chosen political topic didn't give us exact geo-location, it still lead us to collect enough tweets and combine all the analytics we found interesting. It was validated and found meaningful with regards to Australian Politics.

A External Links

Source Code Link:GitHub
Video Link:YouTube

References

[1] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment analysis of twitter data," in *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, 2011, pp. 30–38.

[2] E. Kouloumpis, T. Wilson, and J. Moore, "Twitter sentiment analysis: The good the bad and the omg!" in *Fifth International AAAI conference on weblogs and social media*, 2011.

[3] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining." in *LREc*, vol. 10, no. 2010, 2010, pp. 1320–1326.

[4] L. Barbosa and J. Feng, "Robust sentiment detection on twitter from biased and noisy data," in *Proceedings of the 23rd international conference on computational linguistics: posters*. Association for Computational Linguistics, 2010, pp. 36–44.

[5] H. A. Schwartz, J. C. Eichstaedt, M. L. Kern, L. Dziurzynski, S. M. Ramones, M. Agrawal, A. Shah, M. Kosinski, D. Stillwell, M. E. Seligman *et al.*, "Personality, gender, and age in the language of social media: The open-vocabulary approach," *PloS one*, vol. 8, no. 9, p. e73791, 2013.