Scalable AnalyticS System for Twitter

SASS 4 Twitter

Harvesting geo-specific twitter data to co-relate with datasets present in AURIN to present scenarios of immigrant happiness in Australia and view real time trends in suburbs of 6 metropolitan cities of Australia.

TEAM 31

Student ID	Name
955387	Archith Menon
949136	Kartik Kishore
976508	Nizamuddin
949125	Pinal Gadhiya
963443	Sneh Chirag Thaker

INTRODUCTION

A simple cloud based analytics solution was developed by our team on the NECTAR Cloud platform. The main intent of this project is to extract twitter data, store it on a database such as CouchDB. The harvested data is then used for analytics to bring out stories related to data. Sentiment analysis and other post processing techniques have been used to draw insight from data. The harvesters are extracting data from Twitter using Twitter's streaming API. Harvesters based on Search API were also implemented but were later taken down due to relevance of data for our scenarios.

Around 0.2 million tweets have been collected with help of our live streaming Twitter harvesters. This is a continuous process, the harvesters are continuously running and pushing data to the database. We have designed the system to take into account scalability i.e. the harvester-setup can be grown as per the business requirement and can adapt to increase in infra.

Mango queries have been used to fetch targeted data from the CouchDB Servers. The front end of the application are comprising of simple HTML5 pages which have data analytics information formulated using Plot.ly.

Logging solution logs.io was implemented to monitor statistics of the application server while Project Fauxton provide real time usage and performance statistics of the database server.

Infra on Nectar was spawned using boto scripts and server automation has been provided using Ansible.

ABSTRACT

Initially when we were planning on what to choose as the topic, we thought that we should make use of a scenario where we can find tweets that would have positive and negative sentiments. One of the best scenario available was that of “Australian Politics” where we could find the measure the positive and negative reactions to the current Turnbull government. But we had to scrap the idea since the AURIN data for voters wasn’t available due to some issue with AURIN Data, which was flagged to concerned authorities.

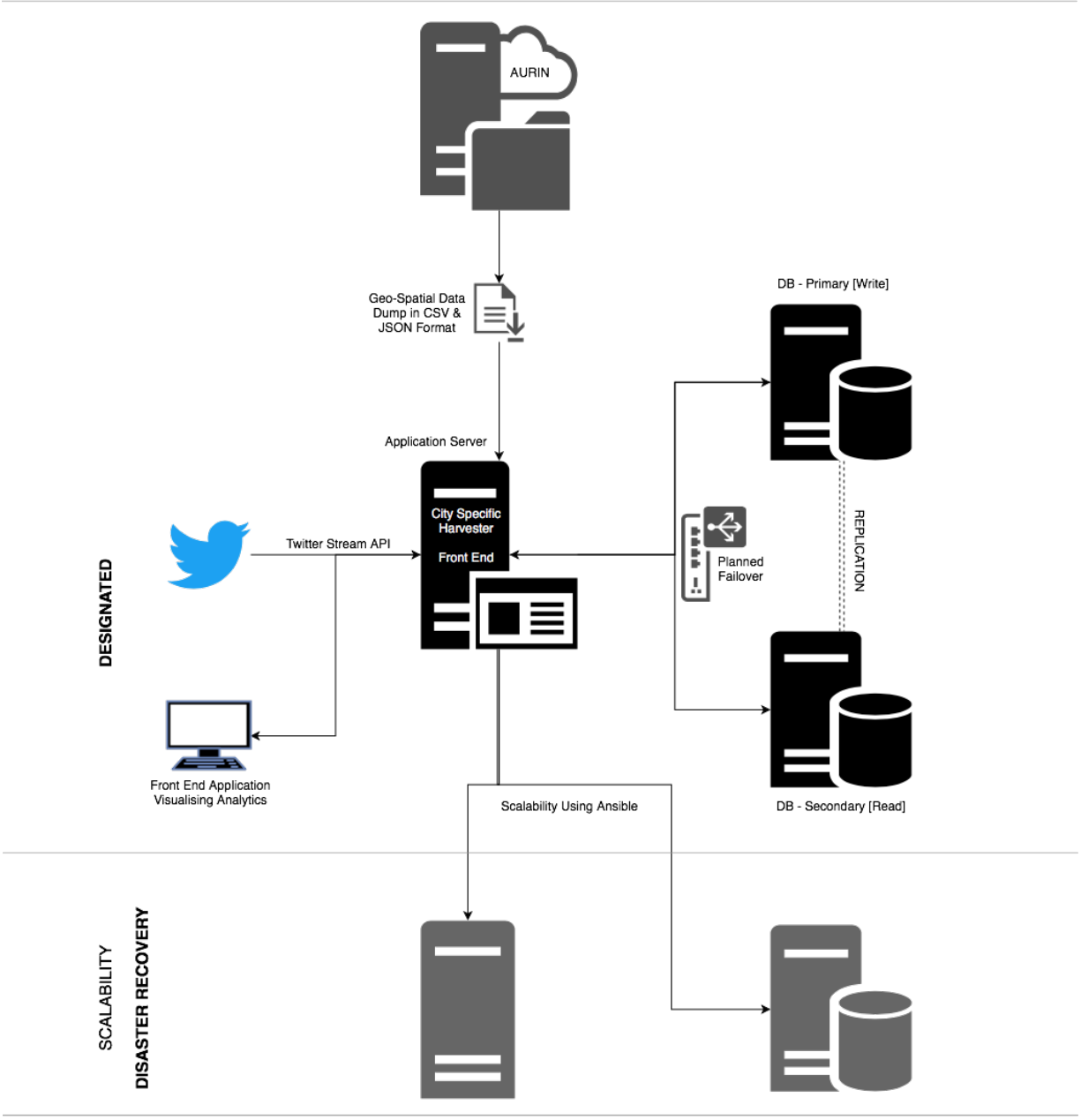
Twitter is a platform where people come and speak their mind. It is also one of the most reliable and fastest means to know what’s happening around the world, most of the times even faster than any news channel. So, we decided to work on real-time tweets and find the location with most number of tweets in a time span. When we determine the sentiments of the tweet, we can ascertain whether something good or bad has happened in the location. If needed, we can even notify the authorities, in case they aren’t at the scene already. This system can also be used to gauge the general reactions to a particular event, like a concert. But unfortunately, finishing this in the given timeframe of 4 weeks would have been a little difficult and we would be cutting it too close for comfort. In the end, we thought “Lets concentrate on Twitter on a granular level”. We took into consideration the main 6 cities of Australia, namely Adelaide, Brisbane, Canberra, Melbourne, Perth and Sydney.

1. As the first scenario, we find the current sentiments of the people in a city based on the tweets collected in the last one week. Then we look into every suburb in the city at SA3 level and find the sentiment of the general populace in each suburb. We then compare this with the data on immigrants that we get from AURIN. This helps us answer the question, “Are people happier where there is a higher percentage of immigrants or local Australian communities?”
2. In the second scenario, we thought let’s look at each suburb and find who’s the most famous personality in that area. So, we took into account the number of followers and display the tweets made by the top 3 people with most followers.
3. For the third scenario, we recreated the “Trending” feature of twitter for each suburb in the 6 major cities taken into consideration. Here we list out the “Top 3 Hashtags” used in each area. This helps us to make sure that the things happening at a suburban level can be focused on without any hassles.

In this way, we try and concentrate on the smaller levels to ensure that the voice of a suburb is not mellowed down and lost between the voice of the whole country. This helps to overcome one of the drawbacks of Twitter of only being concerned with the ‘Big Picture’ and not the smaller parts.

ARCHITECTURE

High Level Diagram of the System:



COMPONENTS

1. Twitter Streaming API & Harvester

We have consumed Twitter Streaming API via **tweepy** for python to create our harvester. Tweepy was used because:

- Python was the language of choice for playing with big data.
- Easy to implement and provides lots of built in features for finding workarounds to get more data from twitter in a certain time period.
- Tweets can be customised to have a string which identifies the app which was used.
- The application doesn't reply on a password, so even if the user changes it, the application will still work. **This helped us recently as an official release from Twitter conveyed that they had stored user password unencrypted in their internal logs.**
- We used a bounding box concept provided by tweepy to get tweets of a particular area. Steps were taken to ensure that we get data from area boundaries specified in AURIN Datasets.
  AURIN Dataset: SA3 - P09 Country of Birth by Sex-Census 2016
  Bounding-Box created via: https://boundingbox.klokantech.com

SENTIMENT ANALYSIS

We started using Vader for sentiment Analysis of the tweet as the analysis processor is portable and needs no immediate training for getting fairly accurate scores.

Python Package: VaderSentiment

Problem faced:

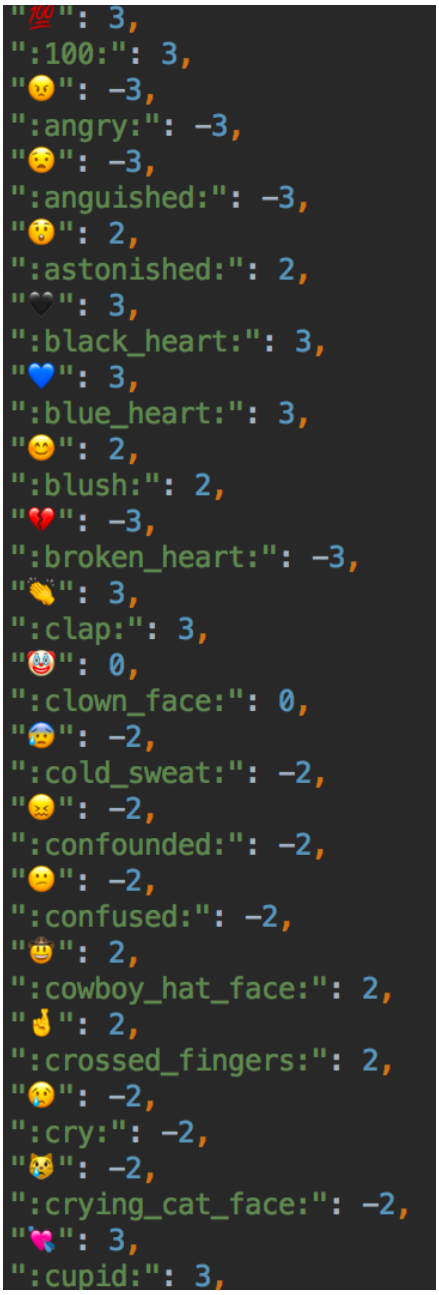
When analysing multiple tweets during early runs of the code Vader was giving extreme polarity scores for the tweets i.e. -1, 1 and 0. When we reviewed the score for the tweet we found out that the sentiment calculated was accurate for 78 out of 100 sample tweets. We then tested TextBlob against the same sample of 100 tweets, the results that we got were fairly the same, 82 out of 100 tweets were accurate with respect to polarity.

Solution:

We then decided to use TextBlob as the analyser of preference.

Upon further review of the data and the polarity scores we found out that a majority of tweets contained emojis which were not being successfully analysed for sentiment. We then created a snippet of our own which identified the emoji present in a tweet and

compared it with a dictionary of emoji's that we have created with their respective polarity scores. The more emoji's we add in the dictionary the more spectrum of emoji's can the analyser handle.



So now our sentiment analyser has 2 parts:

- Text Analyser - takes the value associated with attribute *str* in case of small tweets and *full_text* in case of full 140 character tweets and performs sentiment analysis on the tweet.

- Emoji Analyser - cleans up the tweet of any text and gives a sum of polarity scores associated with emoji present in tweet. This score has been then normalised with respect to the range of the polarity scores provided.

Blending

We have then decided to blend the scores achieved from the above 2 analysers in order to give a final sentiment score that gives of a +ve value for positive sentiment, 0 for neutral and -ve value for negative sentiment.

GEO-PROCESSOR

The module which finds the suburb name where the point is falling on

Input: Coordinates [Either from coordinates explicitly or Centre of Bounding Box] , City_SA3_JSON_File from AURIN

Output: Area Name & Area Code of the area where the point lies in.

Mechanism: Using Shapely package from python we have created polygons of the coordinate data found from AURIN for each and every suburb of the city. Then using shapely.polygon.contains check which suburb the point lies on.

HANDLING DUPLICATES

There are multiple cases when the stream can provide duplicated tweets to the harvesters, we need an approach to identify and make sure no redundant tweets are saved to CouchDB. This is one of the key approaches for saving storage space, improving the speed of retrieval of data as well as ease of indexing when requesting views.

The solution for this is quite simple, each tweets from twitter API has a unique id and id_str. To make sure CouchDB does not lose precision of number we will use id_str to replace the build-in _id key. Each document in CouchDB has same definition of _id and twitter's id_str. So whenever a new tweet returns, we can easily identify whether it exists in CouchDB with extremely low time complexity, as _id could be used for searching and indexing.

PRE-PROCESSING

The RAW JSON obtained from twitter is modified for following cases:

- TweetID has been set as the unique key for storing the document, this helps to filter out duplicates.
- Additional XML attributes of SA3_Code and SA3_Name obtained from Geo-Processing have been appended to the document.

- Additional Sentiment attribute obtained from Sentiment Processing has been appended to the document.

Problem faced:

Getting error when the harvester starts to read a large volume of data.
HTTP Error 420: Harvesting Rate limit threshold breached.

Solution:

Using different credentials to spawn different harvesters for each city. Configure Twitter Streaming API via built in tweepy functions to retry again after sleeping for a while if connection gets closed from twitter end.

Exception Handling:

We have installed a package known as log.io on the harvester server. In the python harvester code we have included proper logging mechanisms. The log.io helps us to visualise logs for different harvesters in real time over the port 28778.

2. Database - CouchDB

FEATURES USEFUL TO US

- CouchDB is a No-SQL database that stores documents in a JSON format, which facilitates self-contained data. This helps us to store the data coming in from the Twitter Streaming API whose throughput is in JSON format.

CouchDB is flexible to store semi-structured documents, which is not the case with RDBMS, thus it helped us in rapid prototyping and testing of how to store in our JSON dump obtained from Twitter API.

Project Fauxton - Allowed us to create, update, delete, view and configure settings of CouchDB with ease of a GUI.

It was easy to play with databases using the REST API exposed by CouchDB.

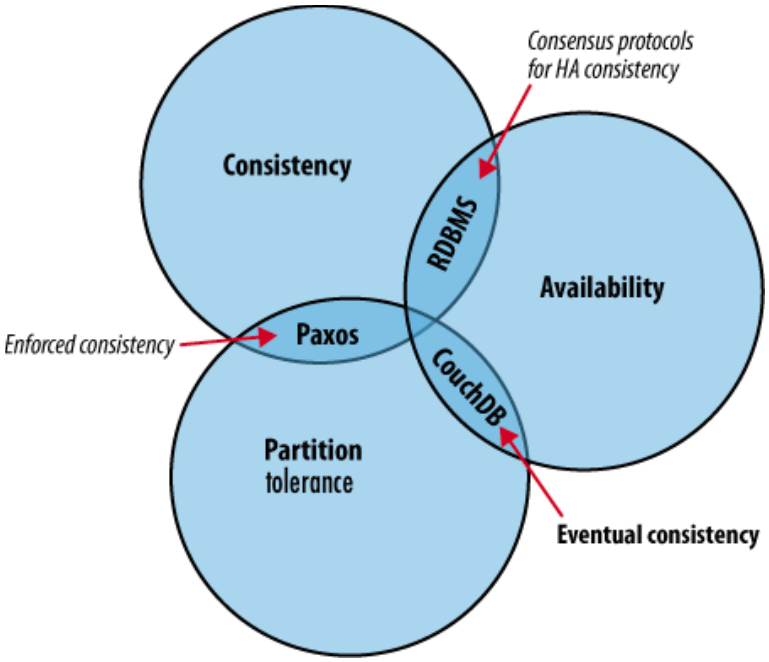


Figure: Through the lens of CAP Theorem | Source: Apache CouchDB Official Website

HIGH AVAILABILITY & DISASTER RECOVERY

- CouchDB provides functionality for clustering where data is made redundant on more than 1 CouchDB instances.
- This can be done by specifying number of shards(q) and number of replicas(n) in the default.ini configuration file of primary database node. This can be done with API Call.

We have installed CouchDB individually on different VMs without Docker.

REPLICATION FOR DISASTER RECOVERY

- To synchronise two or more CouchDB databases. CouchDB replication uses the REST API.

All the nodes in our cluster have a copy of the data present in the primary write node. We have made such a configuration so that if any of our node goes down the other node will be fully capable of functioning on its own.

INDEXING

- Using an index can help prevent full table scans that are usually done iterating throughout the database one-by-one.
- An index is a data structure created on a column of a table; look-ups, deletions, and insertions can all be done in logarithmic time.

We leverage sorting while querying from database by first sorting against the suburb-names for a city - to simulate grouping by - and then against the number of followers - both in descending order. It is necessary to build the indexes to perform efficient sorting.

LOCKING

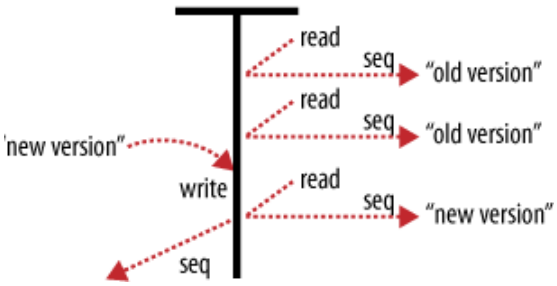


Figure: CouchDB read-write mechanism | Source: Apache CouchDB Official Website

This feature of CouchDB helped us to perform heavy read-write operations on data simultaneously. We were writing data on the DB and creating views of the same, in turn effectively utilising the processing power as clients won't have to wait while the system is processing some other requests.

INTERFACING

Mango query language was used to perform data analytics.

It was preferred over map-reduce because of its JSON alike syntax.Mango provides a single HTTP API endpoint that accepts JSON bodies via HTTP POST.

Each action is specified as a JSON object with a number of keys that affect the behaviour

Supported actions:

- insert:Insert a document or documents into the database.
- find:Retrieve documents from the database.
- update:Update an existing document in the database.
- delete:Remove a document from the database.
- create_index:to create an index on database

Operators:

- syntax:{"$operator": argument}

We have used CouchDB python package for interfacing with CouchDB from our harvester.

As directed, we have used them to lookup and create views from the database.

PLANNING & IMPLEMENTATION from a DBA's Point of View

Problem faced:

First we had planned on using CouchDB deployed on dockers containers across the DB infra. Multiple iterations of the docker CouchDB simulation were run but we were not able to configure the cluster with replication as the official documentation to setup a cluster is very vague.

Solution:

- We went ahead with the vanilla installation of CouchDB v1.6.1 but later found out that it has some issues with clustering and security vulnerabilities, thus we decide to upgrade to CouchDB v2.1.1 which is the latest version with all security fixes in place and provides a GUI dashboard to monitor database.
- The installation of the DB was done from scratch, we downloaded the tar repository and recompiled it on our server, this was done because we wanted to modify the location

where we want to store stuff in our VM. As we recompiled the source and made release folders thus we could move the release folders in any directory of the VM.

- The volume was mounted in /home/couchdb which is the home directory of the CouchDB user. Scripts have been written to ensure that the CouchDB is always run as a service on startup. The data storage and service functionality helps us to ensure that there is no data loss. Even if the VM goes down we can retrieve data from the volume that was attached to the crashed VM and then plug it into a new VM.

Exception Handling:

We have also ensured that proper logging of the database is done in /var/log/ and /etc/couchdb/log/ folders. We can then leverage log.io functionality for real time log viewing & analysis.

IP address	VM	Nature	Intended for	Can Handle
115.146.86.187	CouchDB Server	Primary	Write Operations	Read + Write
115.146.86.70	CouchDB Server	Secondary	Read Operations	Read + Write
115.146.85.222	CouchDB Server/Harvester Server	Scalability	Read/Write	Read/Write

Ideally a load balancer should be kept in-front of the database server cluster so as to distribute the load on a server. We have done this part in our code as a workaround, we try to catch the exception:

```
socket.error: [Errno 111] Connection refused
```

And reformulate the CouchDB server connection by pointing to the other node for writing. This process has been automated by us in the harvester code.

3. Front End Application

For our user interface, we planned to make use of Tableau, a framework that provides data visualisation products. But during the research into tableau, we found that even though it will help make the front end interactive, it didn't have a proper connector to CouchDB. Later on, we looked into QGIS and Plot.ly as option. QGIS has the facility of directly connecting with CouchDB but we found Plot.ly as a more suitable alternative. Using Plot.ly, we plot the locations from where we are getting the tweets with the help of "MapBox" functionality. For our scenarios, we needed to show the relation between

immigrants and the emotions of people. For this we made use of “Pie chart” functionality. For the other scenarios, we are displaying the data in tables in HTML.

Plot.ly has a graphical user interface for importing and analysing data into a grid and using stats tools. Graphs can be embedded or downloaded. Mainly used to make creating graphs faster and more efficient.

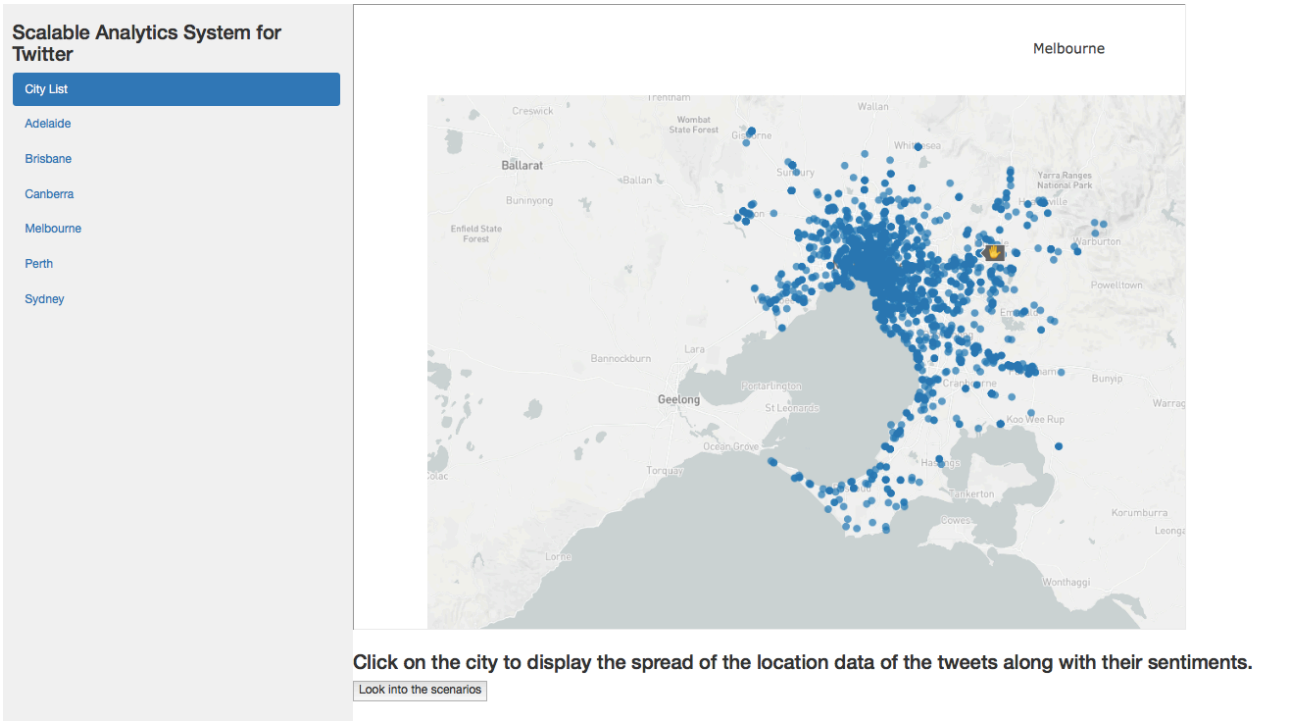


Fig : 1

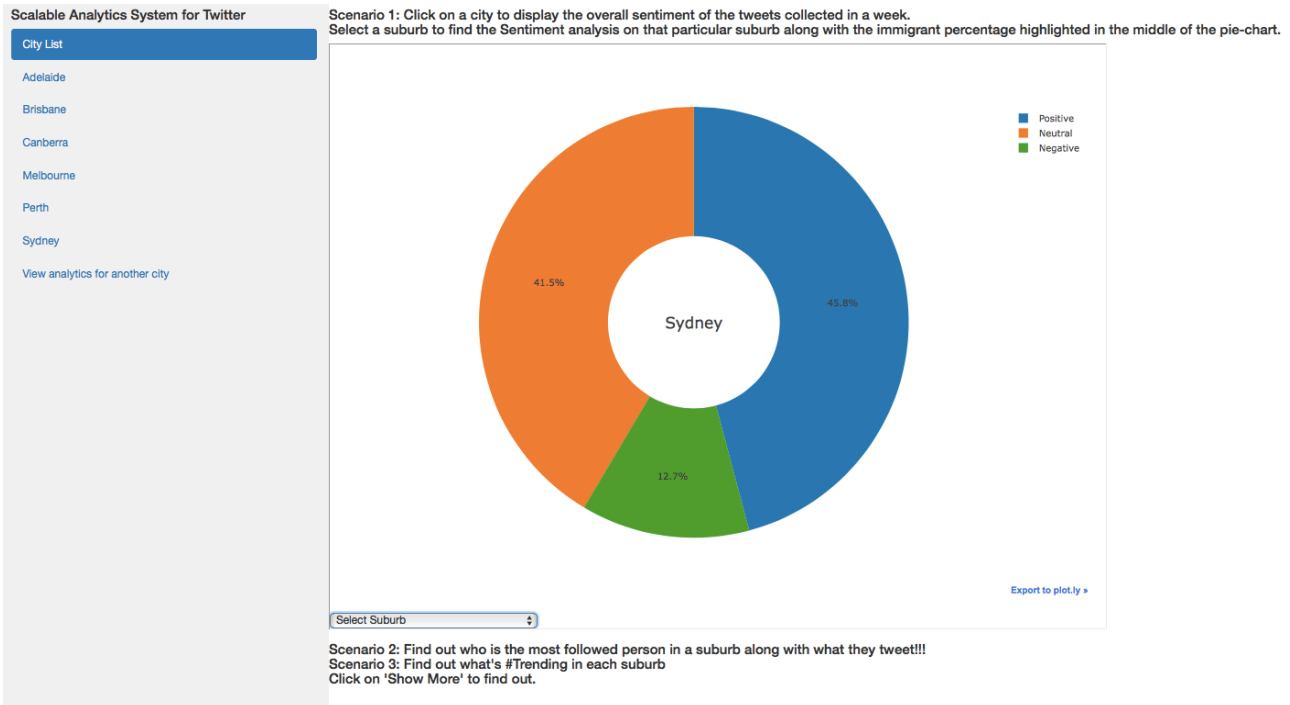


Fig : 2

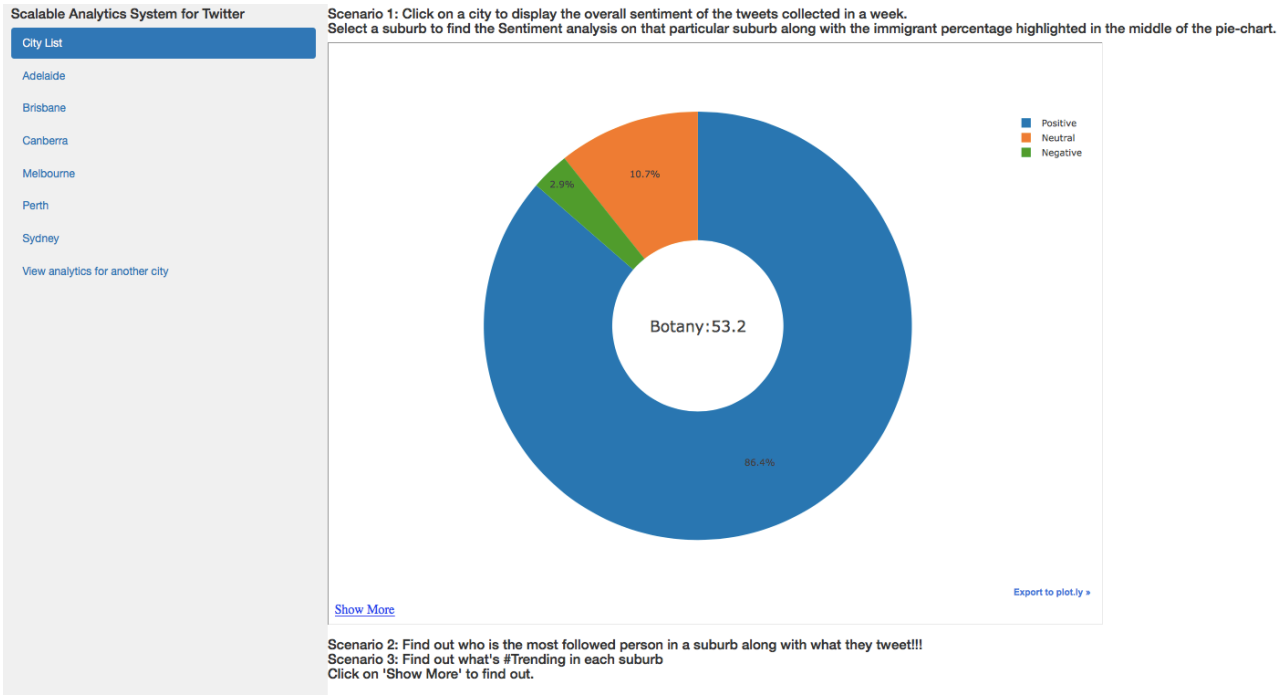

Scalable Analytics System for Twitter
City List
Adelaide
Brisbane
Canberra
Melbourne
Perth
Sydney
View analytics for another city
Scenario 1: Click on a city to display the overall sentiment of the tweets collected in a week.
Select a suburb to find the Sentiment analysis on that particular suburb along with the immigrant percentage highlighted in the middle of the pie-chart.
Positive
Neutral
Negative
10.7%
2.9%
Botany:53.2
86.4%
Export to plot.ly »
Show More
Scenario 2: Find out who is the most followed person in a suburb along with what they tweet!!!
Scenario 3: Find out what's #Trending in each suburb
Click on 'Show More' to find out.

Fig : 3

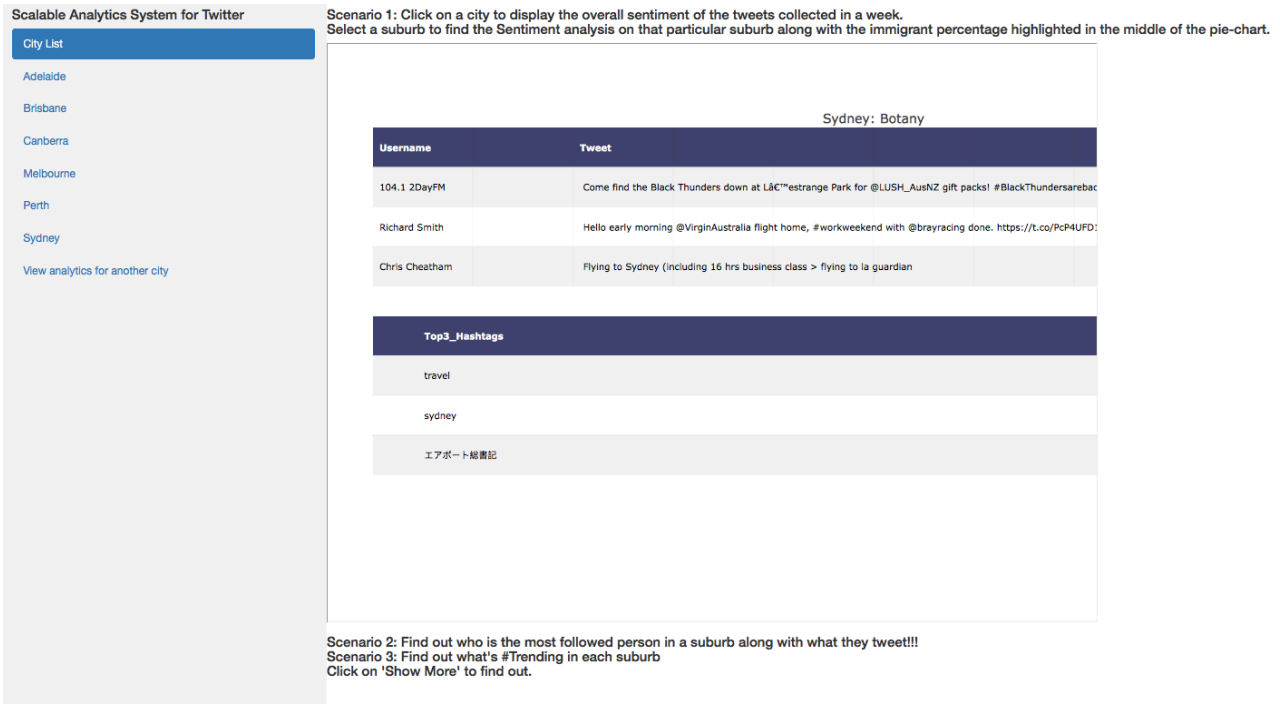

Scalable Analytics System for Twitter
City List
Adelaide
Brisbane
Canberra
Melbourne
Perth
Sydney
View analytics for another city
Scenario 1: Click on a city to display the overall sentiment of the tweets collected in a week.
Select a suburb to find the Sentiment analysis on that particular suburb along with the immigrant percentage highlighted in the middle of the pie-chart.
Sydney: Botany
Username
Tweet
104.1 2DayFM
Come find the Black Thunders down at Lâ€™estrange Park for @LUSH_AusNZ gift packs! #BlackThundersarebac
Richard Smith
Hello early morning @VirginAustralia flight home, #workweekend with @brayracing done. https://t.co/PcP4UFD
Chris Cheatham
Flying to Sydney (including 16 hrs business class > flying to la guardian
Top3_Hashtags
travel
sydney
エアポート総書記
Scenario 2: Find out who is the most followed person in a suburb along with what they tweet!!!
Scenario 3: Find out what's #Trending in each suburb
Click on 'Show More' to find out.

Fig : 4

4. NECTAR Cloud

NeCTAR is based on OpenStack™ which is an open-source Infrastructure-as-a-Service (IaaS) platform. If anyone needs resources to run jobs 24/7 meant specifically for research purposes, he/she can request infra resources on NECTAR.

NECTAR provides a centralised dashboard through which we can manage the computing, storage and networking resources.

We have written python library BOTO scripts for:

- Creating VM from image
- Creating Volume
- Attaching Volume to VM
- Creating Snapshots

Similar functionalities are provided in the NECTAR's dashboard, it allows easy setup and creation of security groups and access to the application.

Problem Faced:

We did not face any major issues concerning the NECTAR cloud other than the issue in installation of packages using *apt-get* for *Ubuntu 16.04 Xenial* OS VMs.

Solution:

We raised this issue with NECTAR Support staff who found out that linux was behaving abnormally and asked us to update files present in /etc/apt/sources_list.d/ to remove the prefix ***au*** from the URLs as a workaround.

5. AURIN

Established in June 2010, the Australian Urban Research Infrastructure Network (AURIN) is an initiative of the Australian Government under the National Collaborative Research Infrastructure Strategy (NCRIS) and associated programmes. It has over 1,800 datasets from over 80 sources covering a wide range of disciplines.

DATASET SELECTION

In AURIN, we were looking for a dataset that wasn't too generic but also helped us to build an interesting relation with the results that we get from our twitter data. In the start we thought we would consider the average income for the family each week in each suburb but we didn't find this relation with twitter data too exciting. Similarly, the datasets with health and happiness quotient seemed too common and overused according to the

examples that Professor Richard had given us on the relations created by students previously.

When we were scraping through the available datasets in AURIN, we stumbled upon the **“Country of Birth by Sex-Census 2016”**. As we, ourselves are non-natives, we wanted to find out the link between the percentage of immigrants in an area with the sentiments of the general populace. We extract the data which depicts the total number of people in a suburb and the number of people born in Australia. Using these values, we calculate the percentage of people born outside Australia.

INTERFACING

Instead of accessing AURIN through API, we downloaded the required data-set and stored it in a CSV file. We followed the following steps to generate the required files:

1. In the Analyse panel, click on “Tools” button to analyse the datasets selected
2. A new window pops-up with a number of options for analysis.
3. Select group “Spatial Data Manipulation” and click on “Spatialise Aggregated Dataset” to generate the co-ordinate values for each location in the dataset.
4. Set the parameters and select the dataset for which .CSV file has to be created and click on Add and Run after setting the name for the file in the Save As section.
5. After this tool produces the required results, open the “Output:” file and on the top left corner, click on the format in which you want to save the file (JSON, CSV, SHP).

6. Fault Tolerance

The whole system has three layers of fault tolerance:

- Crawler level: We have provided Ansible scripts which will be able to spawn fully configured harvesters. Currently we are using a single node for the harvester, but this can be scaled up to add redundant/fail-over nodes in the environment.
- Database level: The cluster of DB Servers is a robust configuration to handle increased workloads and the automatic failover that we have written in scenarios where a node goes down will help with the high availability of the system.
- Web level: Streaming APIs alongside tweepy help to provide resilient services to continue harvesting even if HTTP errors like 403, 420, 500 occur from time to time. Plus we have a real time logging mechanism which can help us to recover from a server failure real quick.

7. Scalability

With the help of Ansible and BOTO we can deploy multiple servers varying in nature.

- Stand-Alone harvesters - This configuration is comprising of a single server working both as a harvester and a database server to function on its own. These can be rapidly deployed across the organisation in order to collect impromptu data.
- Increase in Number of Harvesters - This configuration focuses on running Harvester instances on new servers which can connect to our original database to bring in more data and to distribute workload across data centres. This also helps us to achieve HA-DR. If harvesters are running in data centres spread through out the globe, we can rest in peace even if one of the data centre goes down as the harvester at the other site - Disaster Recovery will be still functional to collect data.
- Horizontally amp up our database tier - New database servers in different geography can be prepped up with couchDB and then can be added to the cluster to facilitate sharding and replication. This ensures high availability and recovery from data theft/ loss.

Different scripts have been provided alongside this report to spawn VMs and prep them for one of the expansions. We have focused on prototyping stand alone harvesters. We have already demonstrated clustering in the Production Harvester Setup that we are

currently using and the expansion of number of harvesters can be done by changing their configuration setting to point them to the existing database cluster.

8. Repositories

Github - https://github.com/orgs/cloudt31/

Youtube - https://www.youtube.com/watch?v=Fc3SU0begs0&feature=youtu.be

References:

1. Couchdb - http://docs.couchdb.org/en/2.1.1/
2. Mango Query - https://github.com/cloudant/mango
3. Plotly - https://plot.ly/python/ipython-notebook-tutorial/#plotting-interactive-maps
4. Ansible - https://www.youtube.com/channel/UCLLumGsi1QboyiFlJf8a-0w
5. Boto - https://boto3.readthedocs.io/en/latest/
6. AURIN Data - https://docs.aurin.org.au/portal-help/analysing-your-data/