

**COMP90024 Cluster and Cloud Computing**

**Semester 1 / 2018**

**Assignment 2 - Australian Social Media Analytics**

**Team 42**

Zijian Wang 950618

Ivan Ken Weng Chee 736901

Thuy Ngoc Ha 963370

Lan Zhou 824371

Duer Wang 824325

Table of Contents

A description of the system functionalities, the scenarios supported and why, together with

graphical results, e.g. pie-charts/graphs of Tweet analysis and snapshots of the web

apps/maps displaying certain Tweet scenarios;

● A simple user guide for testing (including system deployment and end user invocation/usage

of the systems);

● System design and architecture and how/why this was chosen;

● A discussion on the pros and cons of the NeCTAR Research Cloud and tools and processes for image creation and deployment;

● Teams should also produce a video of their system that is uploaded to YouTube (these videos

can last longer than the NeCTAR deployments unfortunately!);

● Reports should also include a link to the source code (github or bitbucket).

It is important to put your collective team details (team, city, names, surnames, student ids) in:

● the head page of the report;

● as a header in each of the files of the software project.

**Introduction**

According to Vivid Social, a specialist advisory service for business strategies with social media, 19% of Australian internet users are using Twitter, accounted for 2.9 million monthly active user profiles (Feb 2018) (<https://www.business.qld.gov.au/running-business/marketing-sales/marketing-promotion/online-marketing/twitter/who>, <https://www.socialmedianews.com.au/social-media-statistics-australia-february-2018/>). The data suggests business opportunities to utilise consumer social media habits to promote sales that suit customers’ needs. This not only benefits large corporations but also small local businesses.

Inspired by the idea, our study focuses on supporting market research for food suppliers by providing insights to predict the foods that people are likely to consume using location, time, sentiments, number of friends, and whether they are homeless. By using our demographics, companies may gain quality understandings of customer behaviours and preferences to adjust existing services as well as develop new initiative products.

Our statistics are based on tweets collected from Twitter users living in Australian cities using harvester tasks running on four instances on NecTAR Research Cloud. Next leveraging CouchDB, MapReduce, Spark, Google Cloud API and Python libraries, under classification and sentiment analysis, the raw data is analyzed together with the data from AURIN focusing on our topic of foods according to selective analytic scenarios. Finally, classified data will then be visualised on our team website connected to our server by the web service module.

In this report, we will discuss our findings for three following scenarios:

* Scenario 1: General sentiment analysis utilising mapreduce.

* Scenario 2: What kinds of food do people around Australia eat?
* Scenario 3: How many homeless people are there around different areas in Australia, and how do their numbers vary over time?

1. to predict the food people eat from other information

2. to predict the future trend of homeless with other information

By studying the scenarios, it is observed that <summary of findings>

**System Design and Architecture**

<Diagram of system architecture>

**System Components**

**Data Collection and Harvesting**

Ivan Ken Weng Chee

<Diagram of component>

* Technologies Used

We have chosen to use Tweepy, which is an easy-to-use Python library for accessing the Twitter API (<http://www.tweepy.org/>). We have also experimented with, Twarc, another library geared towards archiving Twitter data (<https://github.com/DocNow/twarc>).

* Authentication
  + Tokens

In order to use these APIs, we have had to create Twitter applications using our Twitter accounts via Twitter Application Management (<https://apps.twitter.com/>). Once we do so, we can generate four strings which enable us to login and verify our identity autonomously. These are the consumer key, consumer token, access token and access secret.

* Structure of tweet
  + Fields and description

Tweets, also known as status updates, are the building blocks of Twitter. A tweet has an extensive list of attributes associated with it, and is represented via the JSON data structure (<https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object>). These attributes are broken down into three categories, namely:

Root attributes - Fundamental attributes about a tweet

* id
* created\_at
* text
* source
* lang
* truncated
* retweeted\_status
* retweeted\_count
* reply\_count

Child objects - A tweet is a parent object to these objects

* user
* coordinates
* place (for tweets that are geotagged)
* entities
* extended\_entities

Additional attributes - Sometimes provided by Twitter APIs

* Types
  + Streaming
  + Timeline scraping
  + Hydrating

There are a few different ways and strategies to harvest tweets. The most obvious of them is to stream live tweets as they are being posted. <>

For past data however, we can scrape the timeline of a given user and obtain tweets that the user has posted in the past. <>

Another method for obtaining prior data is to hydrate existing Twitter datasets. <>

* Limits
  + Scraping limits
  + How to overcome

While we harvest, we have to keep in mind the rate limits imposed by the Twitter API. Obtaining tweets at a rate of more than 60 tweets per second puts us at risk of having our accounts banned.

* Filtering
  + Keywords
  + Location

The API allows us to filter the tweets we obtain based on particular attributes to narrow down our search, and possibly obtain tweets more relevant to the type of analysis we have in mind.

Initially, we filtered tweets according to keywords related to food. We have a list of five categories corresponding to different classes of food, namely:

* Fast Food
* Fruits
* Meat
* Seafood
* Vegetables

In order to obtain tweets more relevant with Australia, seeing as we are focusing on analysis in this country, we then started filtering tweets based on a bounding box around Australia. This has the added benefit of providing a more focused data range to filter out in the parsing stage.

However, due to the size and nature of Australia’s geographical properties, we do, although rarely, obtain a small number of tweets from the southern part of Indonesia, as the bounding box overlaps that region ever so slightly.

* Interface
  + Connecting to db
  + Inserting data

As tweets are collected, they are checked for format validity and then both written to a file, as well as inserted into the CouchDB database on our nectar instance. At this stage, tweets are just stored as they are collected without duplicate filtering, as that can be expensive to compute while new data is coming in.

* Execution
  + Background task

Our harvesting scripts are run via background tasks using nohup, so as to enable one instance to harvest based on multiple categories. We used three instances to harvest concurrently with three different credentials, to speed up the process.

* Extra
  + Store usernames
    - Regex
  + Scrape timeline based on list

While harvesting, our program uses regex to search each tweet for @usernames and stores them in a file, in addition to the user who posted the tweet. The list of usernames are then scraped to obtain more data, repeating this process each time to obtain more usernames. However, depending on the user’s location, many of the tweets would not originate from Australia.

**Data Parsing and Classification**

Lan Zhou

<Diagram of component>

* Input format

Raw tweet data is given as input to our classifier, which contains all information related to a tweet. The purpose of classifying is to remove unwanted fields and transform the tweet data in a way suitable to be used as input for our analyser.

* Technologies used
  + Vadersentiment
  + Geometry

Our classifier first obtains new tweets from our database, homeless data from Aurin, as well as existing classified tweets, if already stored. Tweets without location data are stripped out, and subsequently tweets outside of Australia. Australian state geometry data is also obtained from Aurin, for purposes of tagging tweets according to their state of origin.

By checking within the polygonal bounds of each state using the Shapely Geometry module (<http://toblerity.org/shapely/shapely.geometry.html>), the homeless data is matched with each tweet, based on its suburb locality. States containing multi polygonal geometry data are merged into a single polygon before bound checking.

Using VADER Sentiment Analysis (Valence Aware Dictionary and sEntiment Reasoner) (<https://github.com/cjhutto/vaderSentiment>), text within each tweet is analysed and given a polarity score. This step is performed in the classification stage as it can be separated from the sentiment analysis for purposes of better efficiency.

In addition to the polarity score, text is also searched for food terms based on types of food, defined in a custom dictionary. Tweets containing terms are given a food\_list attribute listing, again to speed up the analysis process. From here, the classified data is stored in another database, ready to be accessed by our analyser.

* Output format

After classification, tweets are transformed into json objects containing fields such as:

* lang
* polarity
* food\_list
* user
* homeless
* created\_at
* location
* place\_type
* city
* coordinates

**Sentiment Analysis**

Duer Wang

<Diagram of component>

* Input format

The objective of our analyser is to discover the potential relationship between the features we have chosen, and subsequently converts its output into a format needed by our visualiser. The input of our analyser is the output of our classifier program, which are summarised json objects without the unnecessary attributes.

* Libraries used
  + Apache Spark
* Classifier used
  + Random Forest

Our analyser uses Apache Spark, a unified analytics engine for large-scale data processing (<https://spark.apache.org/>). The type of classification algorithm chosen is Random Forest, due to its efficiency on large datasets, and the accuracy.

Data obtained from our database is reformed into a dataframe object, which is required as input to the spark classifying algorithm. Before processing, duplicates are removed to avoid inaccurate results. The data is then filtered into three subsections, namely:

* Entries without food labels
* Entries without homeless information
* The remaining entries with labelled data

These three sections are then further split into training and test data.

Classification is based on the six food classes defined in the food dictionary. The classification model and regression model are trained respectively over the training data. Then, the accuracy of the classifier is calculated following predictions made on the test data.

Data without food labels, and data without homeless labels are then predicted using the model, which then combines the id and joins the predicted data with the original dataframe. This data is then re-transformed into the original JSON format, and appended with additional attributes to match the format required by our visualiser before storage into our database.

* Output format

The data output of our classifier consists of attributes such as:

* properties
* geometry
* Challenges faced
  + Pyspark on nectar instance
  + Transforming json into format required for classification

Some of the challenges faced involves <insert challenges>

**Statistical Information Retrieval**

Zijian Wang

<Diagram of component>

* Input
* Logic

Our second analyser leverages the built-in Map-Reduce functionalities in CouchDB. Given the input from our classifier, it aims to obtain an average sentiment score based on different time groups of the data.

Time is broken down into four different groups, namely:

* morning
* afternoon
* evening
* night

Our analyser then maps each item’s sentiment score into their corresponding groups, and performs reductions to calculate the average score of each group, as the average of a group of numbers is reducible.

A visualisation of our algorithm is as follows:

<insert something>

* Output

With this, we are able to somewhat determine how Australians feel in general during different times of the day. Australians are generally <insert results>

**Data Visualisation**

Hannah Ha

<Diagram of component>

* Input format
* Technologies used
  + MapBox
  + Other libraries

Our visualiser is the front-end component of our system, which aims to display, in a human readable manner, the data we have collected, processed, and analysed. Input is obtained from our classifier program, as well as statistical information outputted from the CouchDB MapReduce algorithm. Our website is written with a collection of JavaScript libraries including AJAX for sending asynchronous requests to our back-end, MapBox and leaflet for visualising our data, and jQuery for additional functionality. It is hosted on one of our Nectar instances via a Python SimpleHTTPServer.

* How front-end connects to our backend
  + API
* Challenges
  + Load huge amounts of data incrementally

We have written a Java REST API to simplify the connection between our visualiser and our database. The web page uses AJAX to request data without constantly reloading the map display. One challenge we face is the sheer volume of data which may be potentially retrieved from the database. Should the map request and load everything in one go, performance may suffer.

* How we divided states
  + Choropleth

We have obtained a dataset containing boundary data of Australian states and regions from the Australian Bureau of Statistics, in the form of geojson coordinates. With minor processing of the data, we have managed to transform the coordinates into visual boundaries, creating a choropleth map. (<https://blog.exploratory.io/making-maps-for-australia-states-and-local-government-areas-in-r-d78edb506f37>).

Another script was developed to perform boundary checking to match the homeless data with the corresponding regions displayed in our map, as this is an expensive calculation and should not be done on each load of the webpage.

Our map visualisation is divided into two major layers, the first and underlying layer for representing homeless statistical data by regions, and the second for populating markers corresponding to the source location of tweets gathered and processed by the other components of our system.

**Scripting and Automation**

Zijian Wang

* Technologies used
  + Ansible
  + BOTO
* Scripts
* Challenges
* User Guide for Testing
* System Deployment
* Invocation and Usage
* Advantages and Disadvantages of of the NeCTAR Research Cloud

4.1. Image Creation

4.2. Deployment

4.3. Error Handling (Issues and Challenges)

**Analytic Scenarios and Findings**

Scenario Selection

Overall introduction to topic: Why and how

Scenario 1

Screenshots and graphs/charts

Scenario 2

Scenario 3

Summary/Conclusion of Findings

**References**

**Appendix**

Links

* GitHub
* YouTube
* Google Drive