COMP90024 Cluster and Cloud Computing Project

* Group Members
* Jiyu Chen 908066

# Abstract

Twitter is a popular social media platform containing large amount of texture data. Aurin provide series of datasets developed and contributed by Australia’s leading researchers. In this project, we will leverage the NECTAR facility to create a four instances cluster environment and mining interesting geoinformation by summarizing tweets from eight cities around Australia and combine them with open sourced Aurin data. We will discuss the system structure, cluster design, tweet crawler, tweet sentiment analyser, sets of Aurin data we have leveraged, views of our data and a demo of the system user interface.

Content

[Introduction 3](#_Toc513390325)

[System Design 4](#_Toc513390326)

[Cluster architecture 4](#_Toc513390327)

[System architecture 4](#_Toc513390328)

[(I guess we have a level structured execution system? ) 4](#_Toc513390329)

[Data Processor 4](#_Toc513390330)

[Hybrid crawler for tweets 4](#_Toc513390331)

[Embedded Sentiment Analyser 6](#_Toc513390332)

[Topic parsing and Hashtag Parsing 6](#_Toc513390333)

[Aurin collector and parser 7](#_Toc513390334)

[Format of processed data 7](#_Toc513390335)

[CouchDB as Database 8](#_Toc513390336)

[CouchDB in cluster 8](#_Toc513390337)

[Duplication prevention 8](#_Toc513390338)

[Scenario Study 8](#_Toc513390339)

[System UI and Web Implementation 8](#_Toc513390340)

[Reference 8](#_Toc513390341)

# Introduction

Nectar stands for National eResearch Collaboration Tools and Resources project. It gives us fix number of computational resources and allowing researchers to create a cluster in their needs with high flexibility for system architecture design and management. We create an one-master-three-slave structure cluster and making use of CouchDB to control passing messages between nodes including message storing, duplication prevention, resource backup, location transparency, synchronizing. We also implement error handling mechanism and parallel computing to enhance the fault tolerance ability of our system. We have also tested the scalability of the system with different number of instances and result in good performances. Twitter data and Aurin is used in our study. For tweets, the twitter api allows us to search for tweets in the past 7 days. The retrieved tweets are in twitter json format containing the user information, tweets, timestamp, geo tag and so on. We will discuss how we utilize the information in later section. One problem for twitter search api is although it’s fast in retrieving many numbers of data in a short period, a access time limit will force it to stop. The stream api instead can solve the limitation well but relatively slower than search api. Therefore, we implemented a hybrid crawler leveraging both search and stream api. <add intro of aurin n aurin harvester>. In tweet crawler, we designed an embedded machine learning sentiment analyser for classifying whether the tweet is sentimentally positive or negative. We also designed a baseline for helping us getting a preview of our task. We tested our sentiment analyser on nltk twitter sample [1] which result in 98.41% average f1-score and on sent140 corpus [2] which result in 66.7% average f1-score. We also implement a basic pattern matching method in generalising tweets related to sports, crime, tobacco consumption and so on. Hashtags are extracted from each tweet for us to conclude some trending hashtags around Australia. <map/reduce> <web> <boto>

# System Design

## Cluster architecture

<add something>

## System architecture

## (I guess we have a level structured execution system? )

<add something>

# Data Processor

## Hybrid crawler for tweets

In this study, we find some interesting relationships among eight cities around Australia based on the tweets posted in Melbourne, Sydney, Canberra, Brisbane, Perth, Adelaide, Darwin, and Hobart. Twitter API provides both standard search API and stream API for harvesting tweets in past 7 days while there is a 15 minutes rate limit for search API and a connection limit for stream API. In our study, the quantity of data strongly influences the analysis result as more data coming in, more normal and general our conclusion will be. Hence, we created a hybrid crawler leveraging both search API and stream API for fast harvesting without touching the rate limit.

Firstly, we created a geo-location filter box by getting marginal coordinates from klokantech[3]. We use eight squares (Figure 1.1) to crop out the area we are interested in so there might be some mis-crops at the edge as city areas are not squares. Since our studying granularity is on city level instead of suburbs, a few mis-crops on the edge are statistically tolerable. Secondly, some tweets filtered from stream API may not contain precise coordination as point longitude/latitude but only contain the city name and a bounding box. In our study, we will not leveraging the precise coordinates of each tweets and only focus on the city they came from.

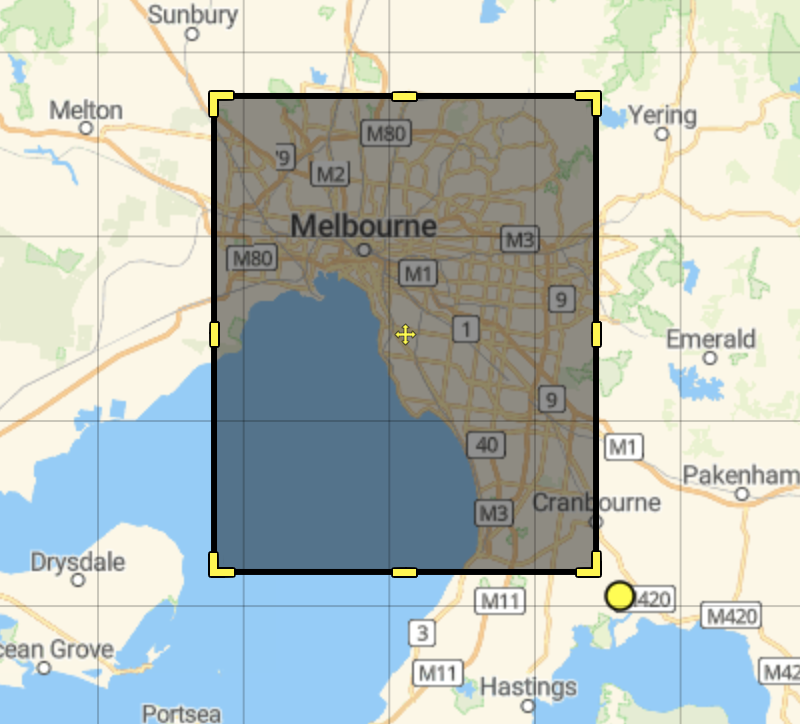


Figure 1.1 Bounding box for geo-filtering

Thirdly, we made an assumption that when a user posted a tweet from a typical city, other tweets from this user are likely to be posted from the same area. Therefore, we embedded a search API after harvesting one stream tweet and to make queries on that user’s timeline. In our test, the success search rate (number of useful tweets divided by number of total queries) increase rapidly. The query time decrease from 100 times per search to 15 times per search which successfully held up before accessing rate limit without slowing the search performance. In the meantime, we also implement a suspend/wake mechanism for search API to cease query after hitting rate limit and restart after a given time. After harvesting tweet we need, we implement an embedded sentiment analyser.

## Embedded Sentiment Analyser

One of the most interesting information in our study is the polarity of people in different areas. We get this data by analysing tweets and classify them into two classes, positive and negative. We implemented a machine learning method leveraging textblob [4]. Textblob can score a sequence of texture symbols including English alphabets and some emojis ranging from -1(negative) to 1(positive). We firstly designed a baseline method to parse tweets without doing any pre-processing and test on nltk twitter samples [1] and sent140 [2]. The average f1-score on nltk corpus is 96.32% and f1-score on sent140 is 42.24%. Then we implemented a pre-processor and tried different combination of pre-process methods including lemmatization, remove stop-words, lower alphabets and so on. We also created an extendable rule-based text parser to transform internet glossaries such as transforming OMG to oh my god. The average f1-score on nltk corpus is 98.41% and on sent140 corpus is 66.7%. The reason that the performance on nltk corpus is higher than sent140 is all data in nltk contain emojis while not in sent140 corpus. This also justified our classifier is successfully parsing emojis.

Table1.1 average f1-score compare of baseline and analyzer

Our analyser is still not accurate enough in sentiment analysis, but it will not cause bad influence on our final scenario study. We assume with sufficient number of tweets, the mis-labelled positive and negative tend to be normally distributed and mutual neutralized. So when we calculating the positive/negative rate, some numbers of mis-labelling are tolerable.

## Topic parsing and Hashtag Parsing

Tweets may contain special topics that we are interested in. Are people more positive with AFL or cricket in the same city is an interesting study. Therefore, we implemented a basic pattern match topic tagger in the data processor. We constructed different extendable topic glossaries and make sure they will not overlap with each other. We created four topics including Tobacoo, Crime, AFL, and Cricket. For tweets that exclude our defined topics will be tagged with null topic. Another interesting study is hashtag parsing. Each year, twitter will publish a summary on most popular hashtags people used around the world. We decided to summarize some popular hashtags around Australia in past 7 days and study people’s polarity trend on different hashtags. We use regular expression to extract all hashtags in each tweets and store them under hashtag keyword in a json format.

## Aurin collector and parser

<add something>

## Format of processed data

|  |  |
| --- | --- |
| \_id | CouchDB unique document ID |
| \_rev | CoudhDB document rev |
| id\_str | Unique tweet id |
| coordinates | Twitter json coordinates |
| place | Twitter place json |
| Place\_type | Granularity of the place, city in our study |
| name | City name |
| Bounding box | Twitter json geo bounding box |
| Country\_code | AU |
| user | Information of user who posted tweet |
| id | User id |
| name | User name |
| description | User profile discription |
| lang | Language of tweet |
| text | Text content of tweet |
| sentiment | Sentiment analysis information |
| polarity | Range from -1 to 1 |
| subjectivity | Range from -1 to 1 |
| label | Positive or Negative |
| topic | Tobacco/Crime/AFL/Cricket/null |
| hashtag | A list of hashtags or [] |

Table 1.2 Processed tweets

# CouchDB as Database

## CouchDB in cluster

<add something related to sharding, structure, etc..>

## Duplication prevention

In our system, three data processors in each slave work in parallel and save processed tweets into uniform CouchDB running on database instance. We leverage the automatic document duplication prevention mechanism in CouchDB to help us ignore harvesting redundant tweets. Each tweet was given unique id by twitter, and each document in CouchDB is given a unique id. Therefore, we use tweet ID as document ID and if there is a duplication exception from database, we will discard the tweets.

# Scenario Study

## Tobacco and Alcohol Reaction

<add something>

## Crime Reaction

<add something>

## Shout out for AFL or Cricket

<add something>

## Hashtag Ranking

<add something>

## Etc…

# System UI and User Guide

<add something>

# Reference

[1]nltk twitter sample

[2]sent140

[3]http://boundingbox.klokantech.com.

[4]Textblob