COMP90024 Project 2 Report

[Group 22 - Melbourne]
Na Chang 858604
Zepeng Dan 933678
Junhan Liu 878637
Yanjun Peng 906571
Peishan Li 905508

Table of Contents

1 Introduction

1.1 Application Overview

The system is consisted of 4 major sections:

- A data harvester module that utilises Twitter's streaming and search APIs to harvest tweets. The data storage is powered by couchDB.
- A data analysis module that processes the tweets for sentiment detection and topic classification.
- A data sorting module that groups tweets of an area according to the account, and time.
- A web application module that provides visual presentation of the analysis result, which is connected with the web server using RESTful API.

1.2 Code and Invocation Methods

Youtube Link:
https://youtu.be/kImrtLE3XcE

Github Link:
https://github.com/tryerleader/missionimpossible

System	**Entry Point**
Internal System Start	main.yaml
External End User Entry	http://115.146.86.149/index.html

2 Architecture

2.1 System Architecture

The system contains 4 instances, each with one data volume attached to it.

Servers

All four servers have the same capacity, as in most stages, the processing load is equally distributed among the four machines.

During the data harvesting stage, all four instances have the full harvester (streaming and searching) running at full power to collect data. On instance 1, 2, 3, the harvester will pass the data to the volume attached. On instance 4, the data will be passed to Volume 2, which is the main node of the cluster.

In the production stage, the harvester will be powered down to only streaming, to provide real-time data update to the web application. The web application will be hosted on instance 3.

Persistent Volumes

Volume 1, 2, 3 will form a couchDB cluster, with Volume 2 (60 GB) being the main node. Volume 2 is used as a primary source for computation, and Volume 1 (40GB), Volume 3 (25GB) acts as fall-back points, should the connection to Volume 2 fails.

Volume 4 is isolated from the cluster, and acts as a light-weight backup database for the main cluster. In Volume 4, only the raw data will be stored, and all the views built will be eliminated. Regular snapshot of Volume 4 is taken to further back up the system data.

Web Application Interface

The web application is connected with the backend using RESTful API. Backend web service is developed with Node.js and express framework. The web application server processes the requests from front-end users and responses with static JSON files or real-time streaming data in JSON format respectively. It connects to CouchDB with minimalistic CouchDB driver nano and implements monitoring the changes feed.

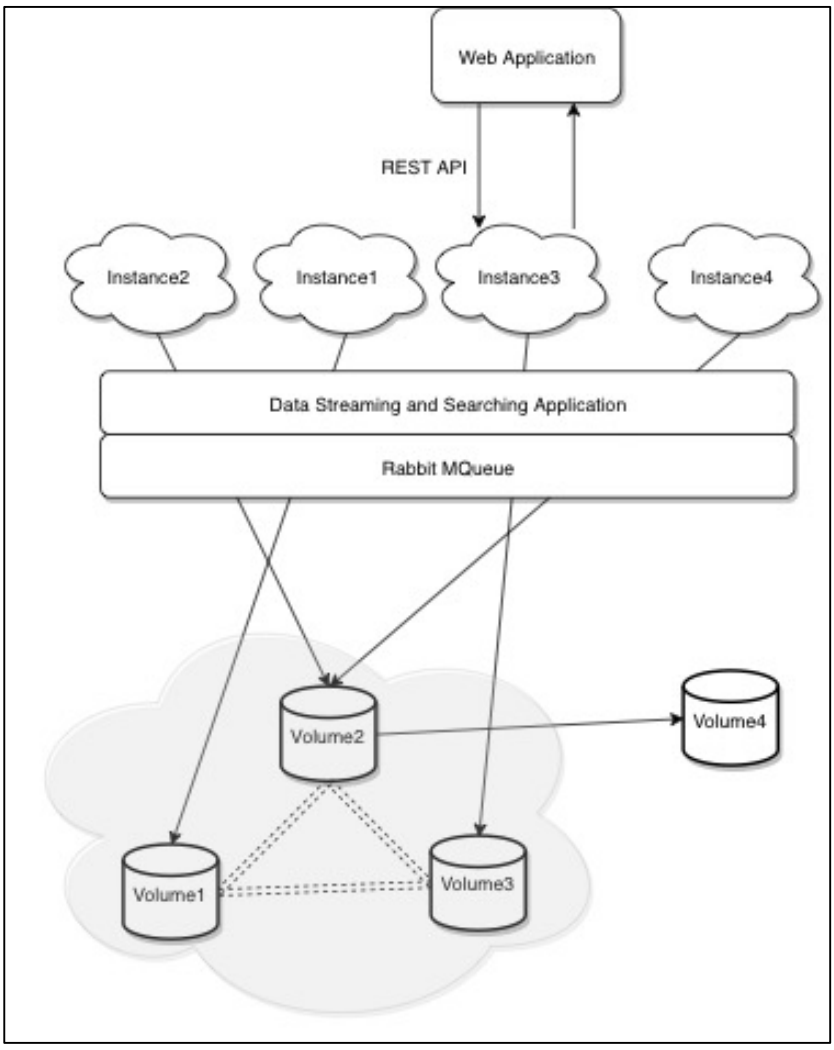

Figure 1 System Architecture

2.2 Data Storage Architecture

Data are stored in three major sources.

- CouchDB – Analysis Database

The harvested tweets are stored in CouchDB. Different views are built to support the data processing and analysis.

- JSON Files

To avoid the long latency during CouchDB view rebuild so that the front-end has a smooth user experience, the result of the data analysis is exported as JSON file, and stored in web application server.

- CouchDB – Streaming Data Feed

As the relatively low data volume, the real-time data feed on the home page are stored in CouchDB, and the web server directly fetch the data from CouchDB by exploiting the _change interface.

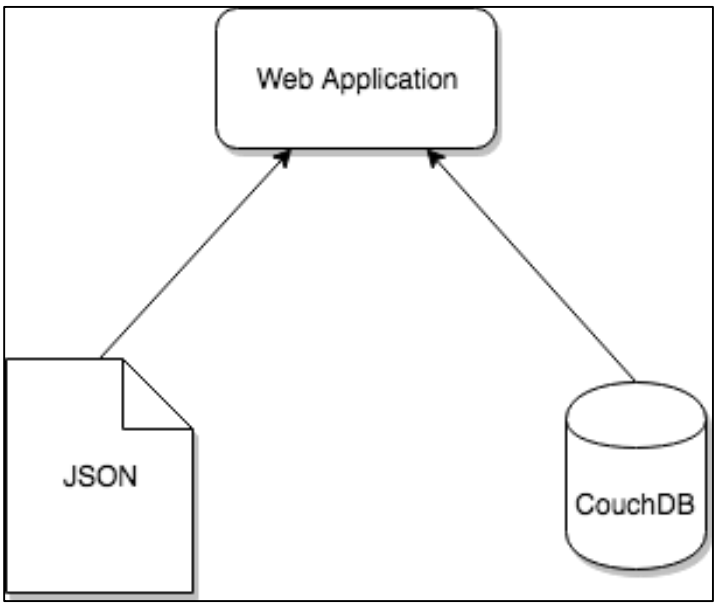

Figure 2 Data Structure

3 System Functionalities

3.1 Data Gathering

3.1.1 AURIN Data

AURIN data is used for macro analysis of different cities in Australia, which is imported to CouchDB via JSON file from AURIN.

Data Category	**Usage**
LGA Data by Region - Income, Education, Employment and Health 2011-2016	Extract income information to identify correlation with sentiment analysis result
LGA15 Self Assessed Health - 2014-2015	Extract health information to identify correlation with sentiment analysis result

In our project, we applied two data sets from AURIN for macro analysis of different cities in Australia. One is LGA Data by Region - Income, Education, Employment and Health 2011-2016, which presents the income, education, employment and health data available from the ABS's Data by Region statistics. This dataset is based on the 2016 Local Government Areas boundaries, and covers data for individual years between 2011 and 2016. Here, we mainly used income and name of local government area to analyse further. Another data set is LGA15 Self Assessed Health - 2014-2015, which includes the estimated number of people that had 'poor' or 'fair' health and those people as a rate of the total population. The data on which the estimates are based are self-reported responses, reported to interviewers in the 2014–15 National Health Survey. Here, we considered the estimated number of people who assessed their own health as fair or poor per 100 population as our health index to measure the people's health status of each city.

Thanks to the datasets from AURIN, we could explore the potential relationships between income, health and happiness of people in Australia. We measure the degree of people's happiness through analysis of sentiment of tweets data, which are stated in detail in subsection 3.1.2. And approaches and resources we used for sentiment are discussed in subsection 5.3.

3.1.2 Tweet Data

Data Category	**Gather Method**	**Usage and Selection Criteria**
Tweets Table	Twitter's streaming and search API	• Macro analysis for different cities across Australia. • Only tweets located inside Australia will be retained and further processed for storage.
Streaming Table	Twitter's streaming API	• Homepage real-time data updates. • Only data with accurate coordinates will be kept so that it could be pin-pointed to the map. The tweets with no

		geo-location information, on merely just a vague bounding box will be discarded.
Tracking Table	Twitter's search API	• Extract the top 300 accounts amount the most active users from 800,000 data entries of Tweets table, to project their individual tweeting patterns. • During View building, the tweets without precise coordinates are filtered out.
Victoria Tweets	Data Import	• Group by time slots to project the tweeting patterns across Melbourne • During View building, the tweets without precise coordinates are filtered out.

3.1.3 Data Support

As CouchDB system defined, map and reduce functions are for processing and generating big data set with a parallel algorithm. Based on the data need for our project, the map and reduce functions are fully exploited to extract specific data from original documents. In our project, we propose to extract the following information.

Application	**Information Need**
City Page	Location, Tweet content
Melbourne	Location, Time
Stalking	User (ID, Name), Location, Time

For different information needs, the generated views are presented with different key-value pairs. On one hand, the data filter process could be easily made such as category tweet basing on its created time. On the other hand, newly inserted tweet could be handled fast since inserting new branch to B-tree is fast.

3.2 Web Application Functions and Analysis Results

3.2.1 Home Page

Home page displays the live feed of harvested tweets from Twitter streaming API. The client fires update request to the server every 10 seconds, which will grab the latest data entries, and return the coordinates and the tweet text.

The coordinates will be projected to the map, while the tweet text will be added to the word cloud. As the text accumulates, the word cloud will reflect the latest trend on twitter.

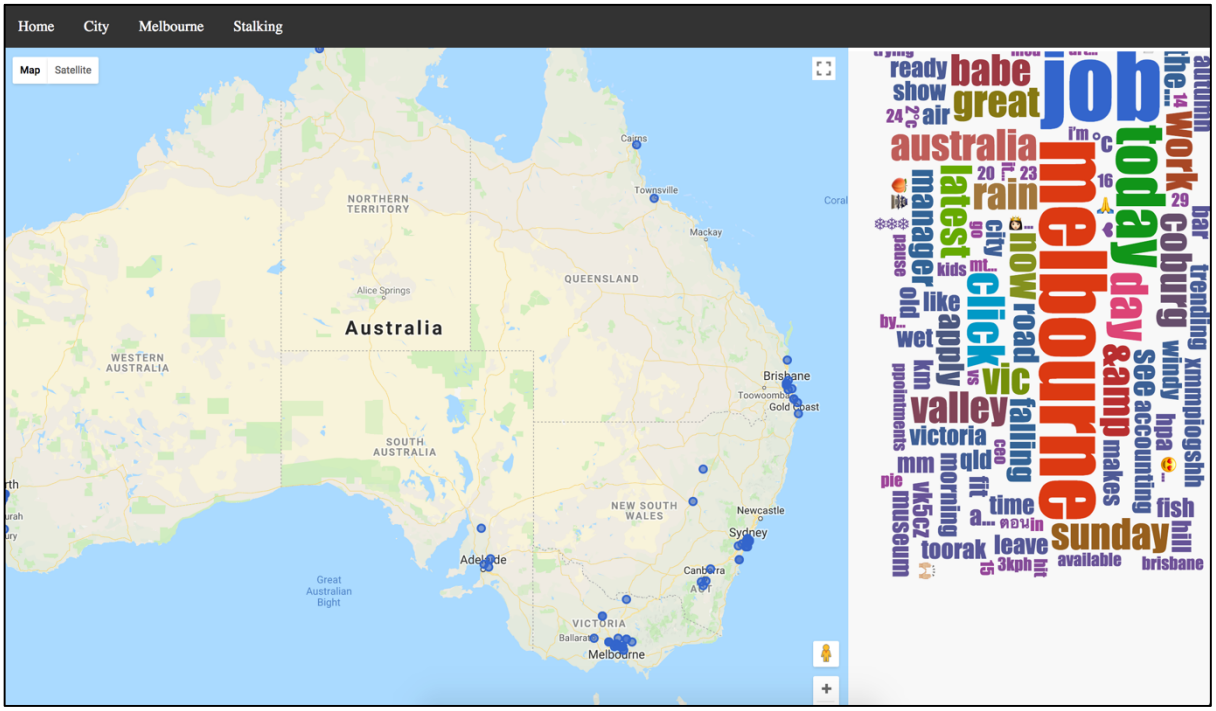

Figure 3 Home Page

3.2.2 City Page

The page displays the sentiment analysis result, coupled by the official income and health data from AURIN, aiming to identify potential correlations between the attitude and income/health level.

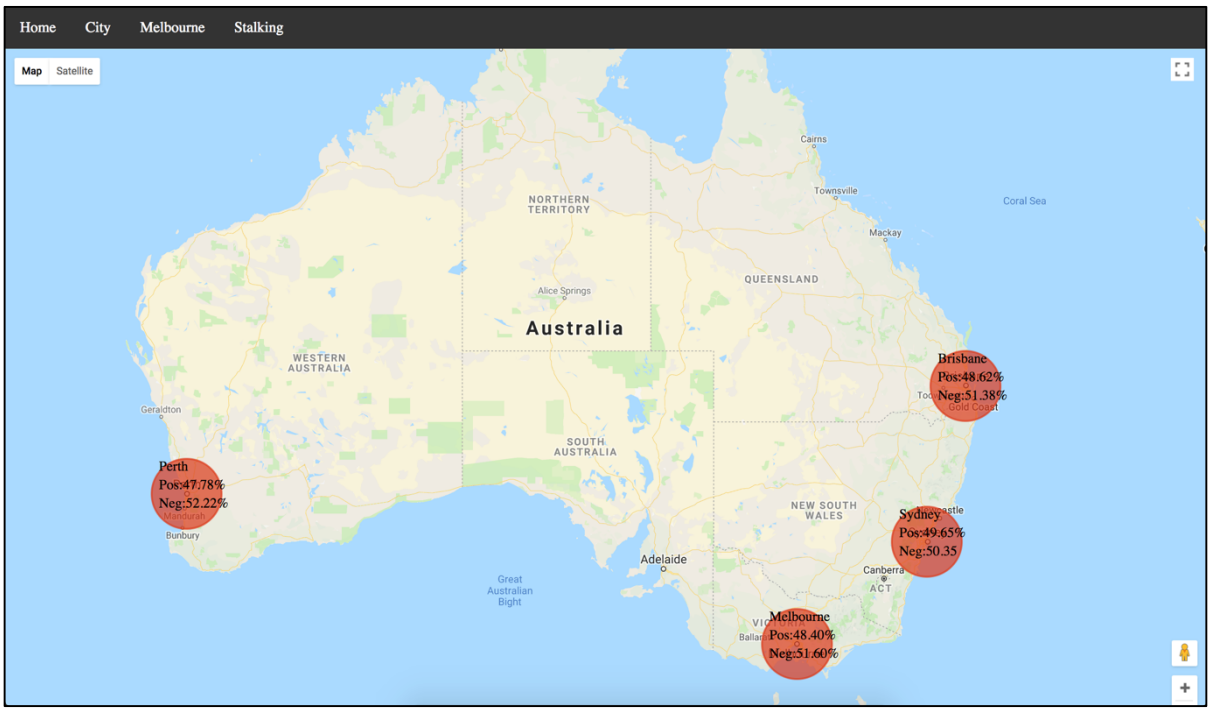

Figure 4 City Landing Page

In the example below, multiple suburbs are selected for comparison. The negative correlation between Income and Health Issues are quite prominent, the higher the income level, the lower the health concerns. There is also a slight positive correlation between income and positive tweets, with exception of Port Phillip. Considering that there are Luna Park and the beach in Port Phillip, the high number of tweets and positive tweets is not too surprising.

Same pattern can be seen in other cities and suburbs.

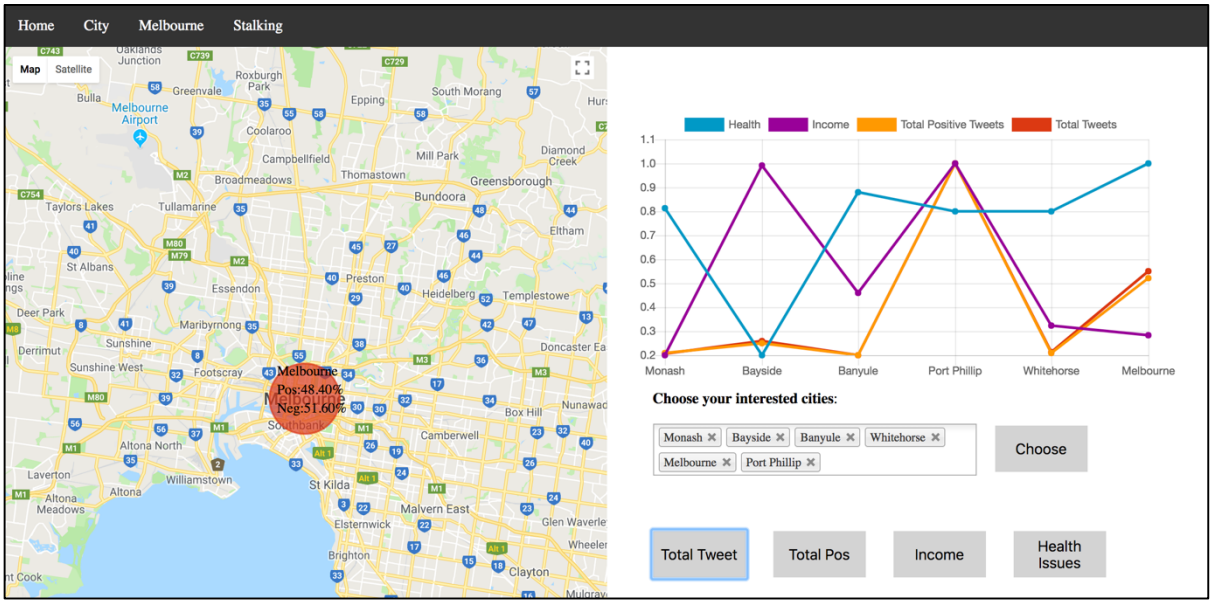

Figure 5 Suburb Analysis Result - Melbourne

Figure 6 Suburb Analysis Result - Perth

3.2.3 Melbourne Tracking Page

The tweets with coordinates in Melbourne are extracted and grouped according to time slot, and projected to the map. This aims to find any potential patterns in the peak time of tweets and locations. 3,000 tweets are uniformly sampled from 140,000 tweets from each time slot.

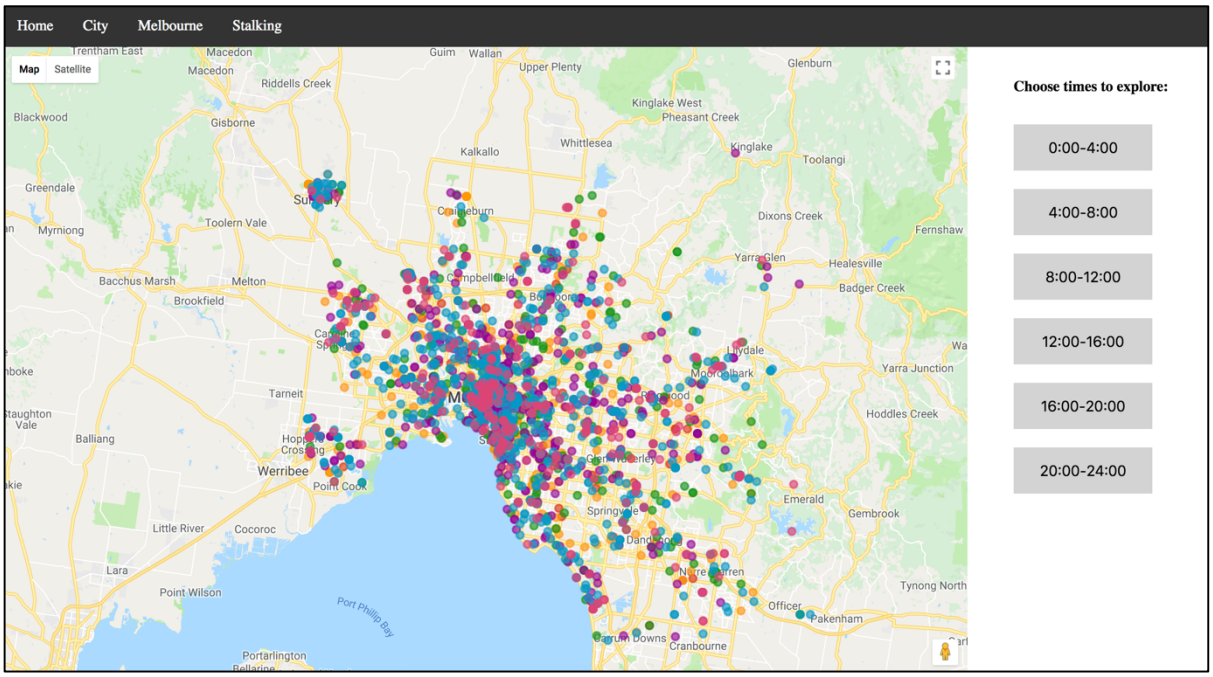

Figure 7 Melbourne Page

Overall, the time patterns are quite obvious. From 0:00 – 4:00, there are very minimum tweeting in this area. Moving from 4:00 to 8:00, the tweets increases significantly. From 8:00, the number of tweets continues to grow until 20:00. From 20:00 to 24:00, the tweets decrease dramatically. Furthermore, it is easy to see the trend of people gathering to the city centre during the day, and scattering to suburbs during the night.

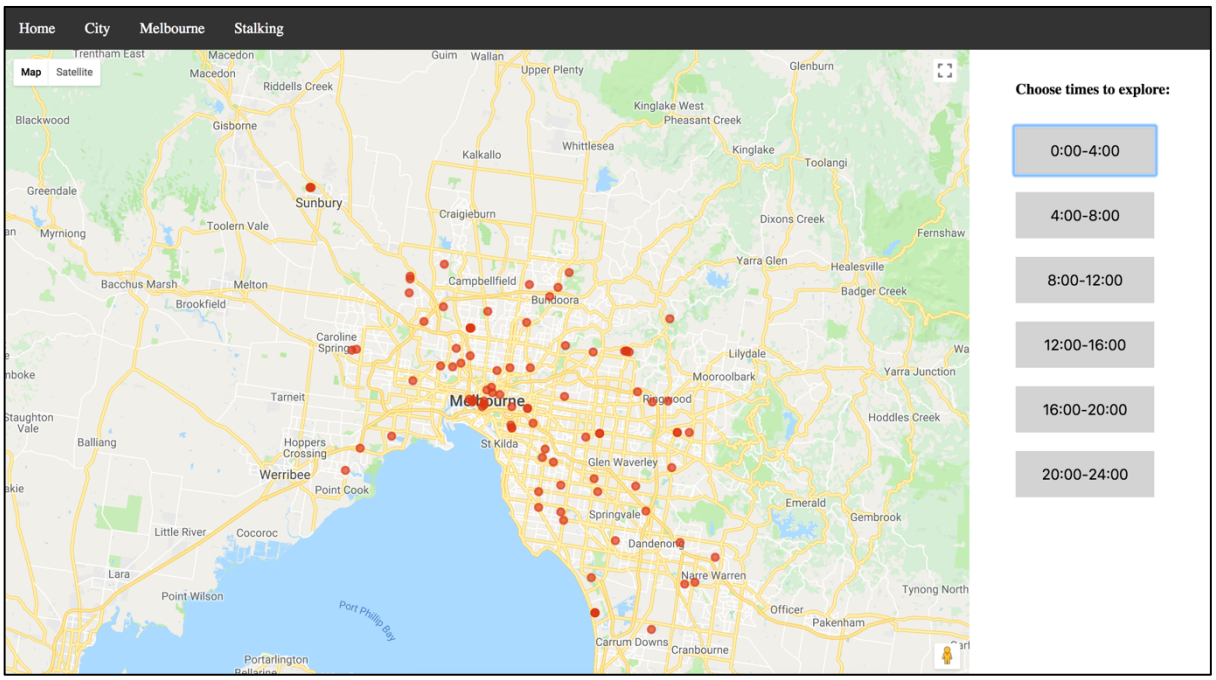

Figure 8 Melbourne Page - 0:00 - 4:00

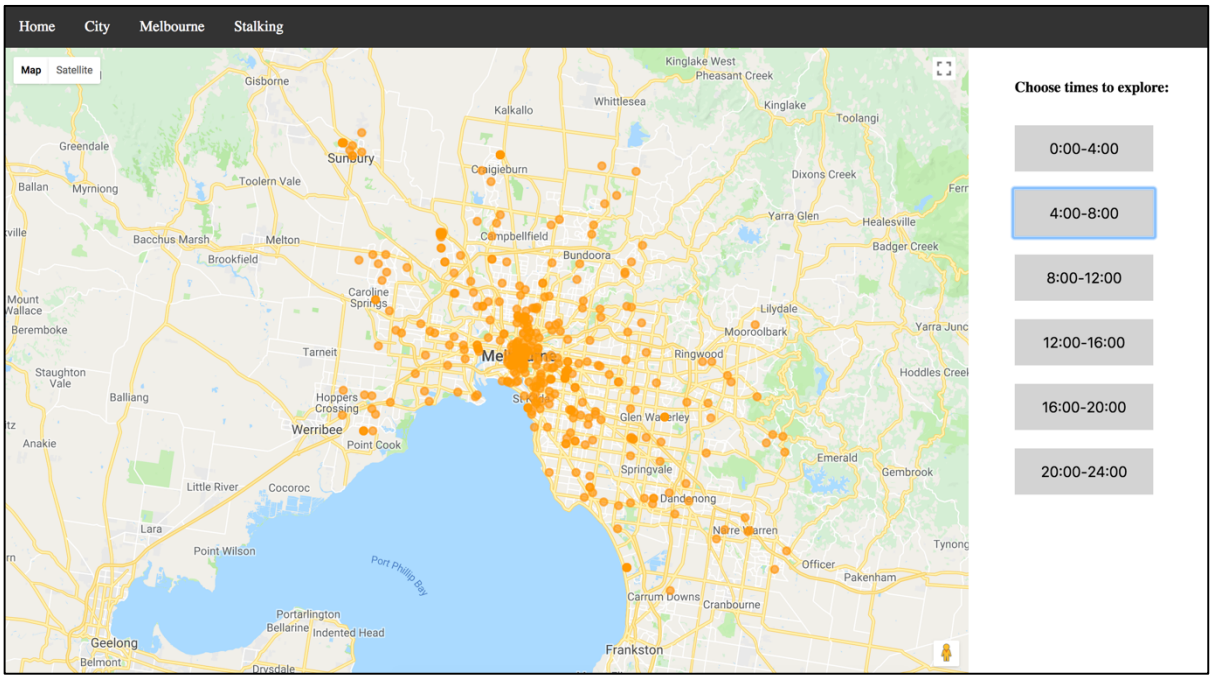

Figure 9 Melbourne Page - 04:00 - 08:00

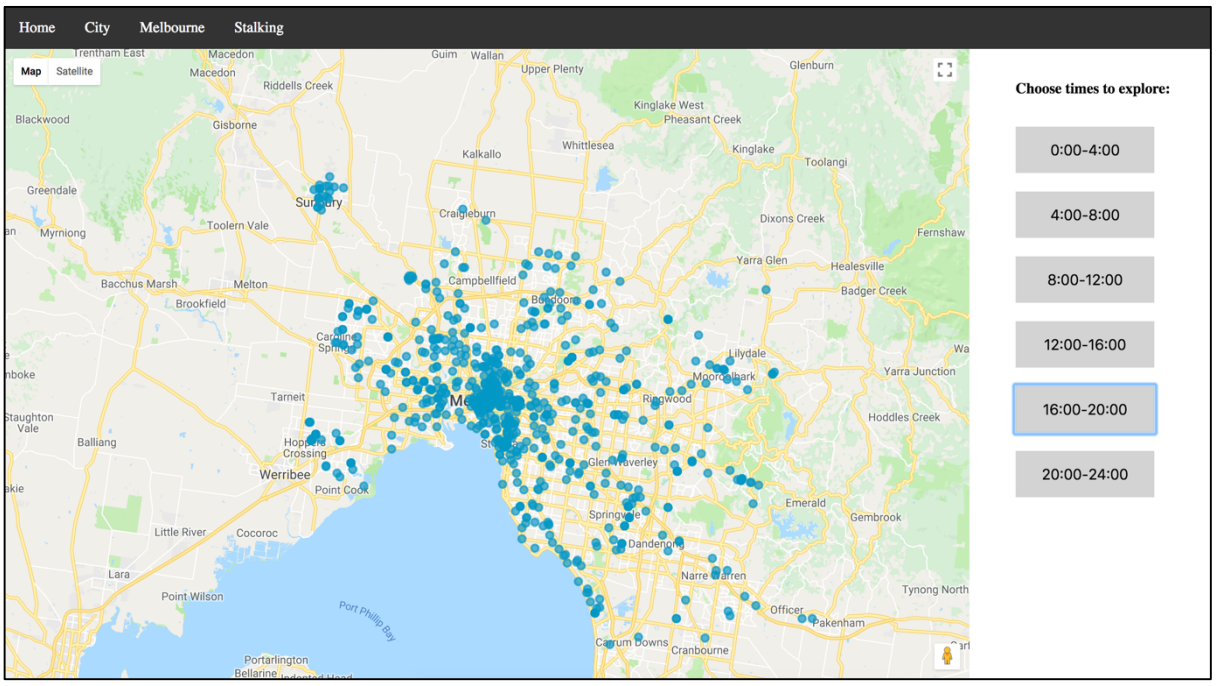

Figure 10 Melbourne Page - 16:00 - 20:00

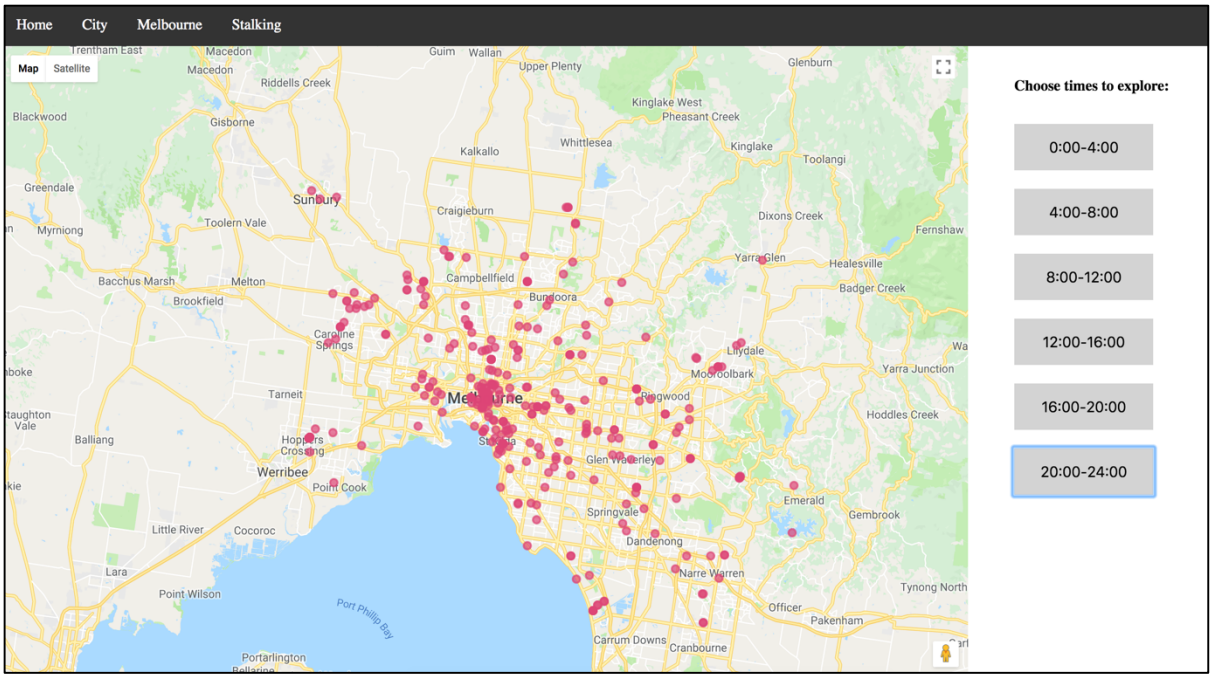

Figure 11 Melbourne Page - 20:00 - 24:00

There are several interesting scenarios identified during the implementation.

- There is a slight gathering at the position of Flinders Station during 8:00 – 12:00, which matches the commute peak. And the concentration disappears afterwards also as expected.

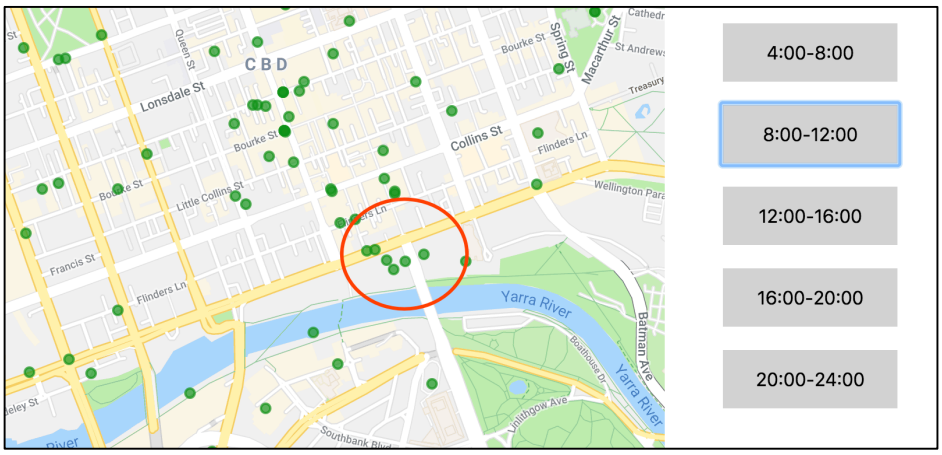

Figure 12 Flinders Station 8:00 - 12:00

- In the area of Dockland and Etihad Stadium, there is no tweet traffic during the early phase of the day. From 16:00 to 20:00, people start to gather in that area for night life.

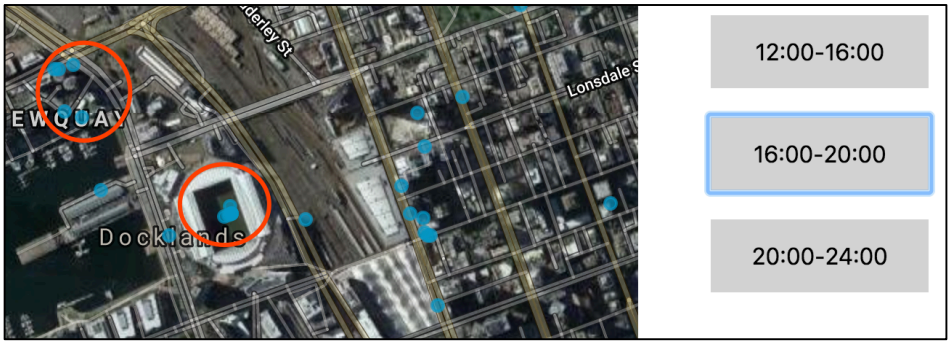

Figure 13 Dockland and Etihad 16:00 - 20:00

3.2.4 Individual Account Stalking Page

When the tweets accumulated to 800,000, the database is ranked to identify the most active user list, according to their tweet frequencies. Afterwards, a separate search harvester is deployed to search all available tweets from the active user list. Then the tweets are passed through sentiment analysis, topic classification, and time grouping, which aims to analyse the user's general attitude on social media, and his/her preference.

The list of users is explored, and several interesting accounts are identified.

- Account 1: M1PacificMwyNSW

The user is a traffic monitoring public account, which feeds live traffic for the M1 Pacific Motorway (F3) between Sydney and Newcastle. The coordinates of the tweets are along the motorway, and the time are from the same time slot.

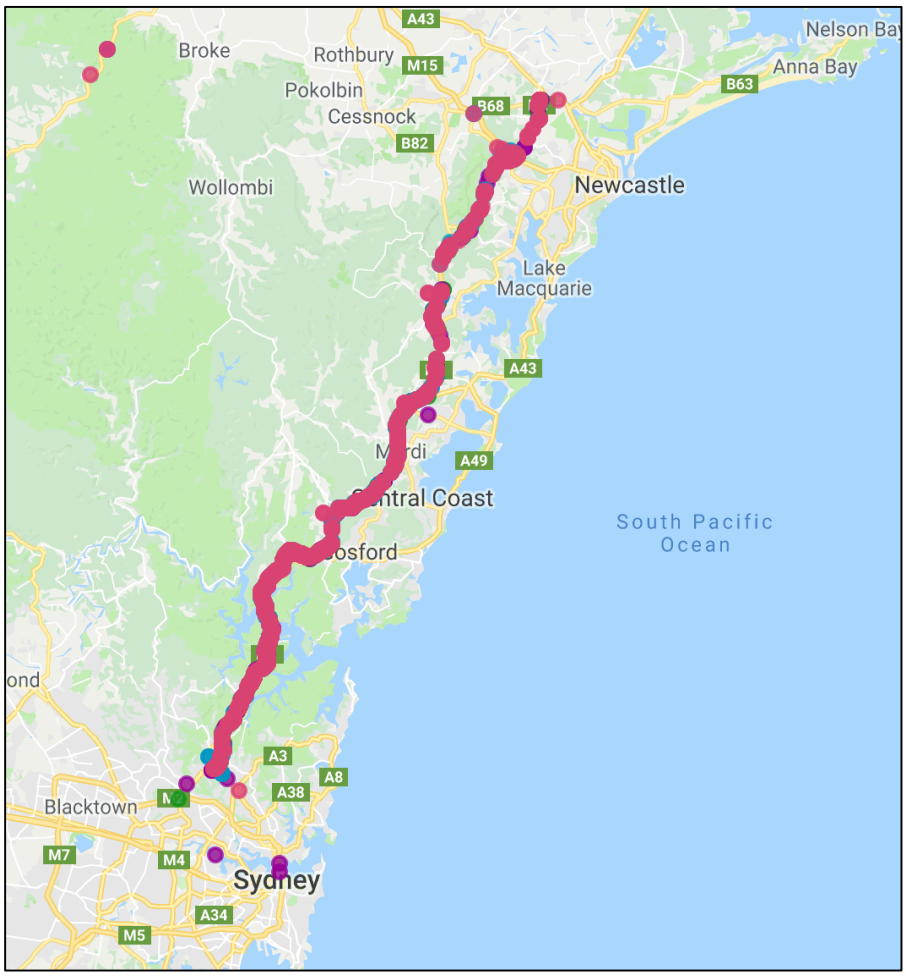

Figure 14 Twitter Account 1

- Account 2: JamesNankivell

According to the introduction on his twitter account, this user is a professional golfer. And the data analysis reflects it. His tweets concentrate on two spots: one in Melbourne Park, which is probably where he works, and one spot in Prahran, which is possibly where he lives. The main topics of his tweets also matches his professional, which are sports and business.

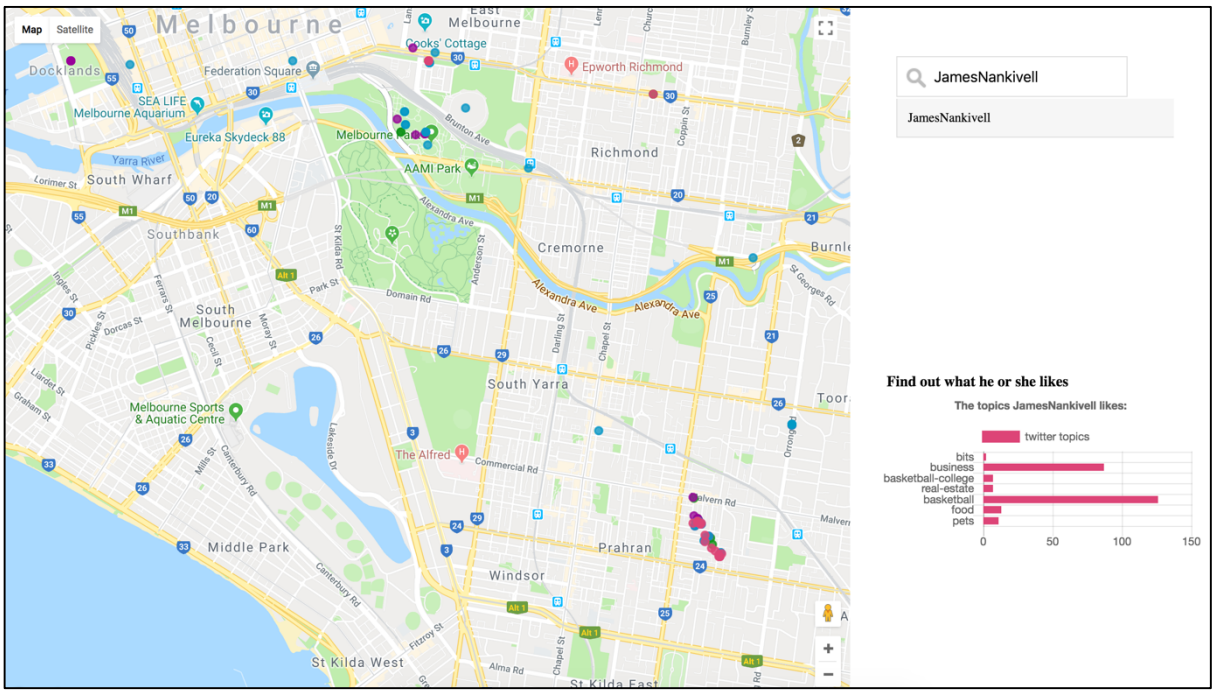

Figure 15 Twitter Account 2

4 System Scalability and Fault Tolerant

4.1 Data Harvester Error Handling

The data harvester includes 4 major components:

- a queuing system powered by Rabbit MQ Server,
- Tweepy module that integrates with both Twitter search and streaming API,
- CouchDB implementation,
- the middle tier for CouchDB Connection.

Potential errors are handled according to the component breakdown, to make sure the system running smoothly.

4.1.1 Heartbeat and System Logging

As the Data Harvester is continuously running as background processes on the instances, in case of any errors, it will not be possible to observe the run-time result in the console. Therefore, a heartbeat and system logging module is deployed to monitor the background process and its performance for data harvester.

Heartbeat periodically checks the status of the process, and record the time to a log file, which simplifies system debugging, and eliminates the need to issue multiple commands to manually check each process. Heartbeat is implemented as a thread-based programme. Within the data harvester module, every queue, searcher, and streamer is assigned a heartbeat thread.

Similarly, the logging system is deployed to each component, which works together with the try-catch sections to capture any errors occurred during run-time.

4.1.2 Rabbit MQ Queuing System

Due to the heavy data incoming flow, and the concurrency issues due to multiple harvesters, a queuing system is implemented. Therefore, an extra step is added in the data harvester (streaming and searching) to ensure that the Rabbit MQ server is running on the current instance.

Furthermore, the channel between the data harvester and the queuing server is automatically shut down in the absence of tweet feed for over a minute. But for the data harvester, it is common to have no incoming tweets due to the processing time, or the upper limit imposed by Twitter API. To maintain the connection between the data harvester and the queuing server, a re-connection mechanism is introduced. If the data harvester detects that the channel is disconnected, it will issue another request, to re-establish the channel for tweet feed.

4.1.3 Tweepy

- **Tweet Harvesting Limit**

Twitter API imposes a request limit of 180 tweets for every 15 minutes. To keep the process alive when the limit is reached, a stopwatch is implemented where it puts the harvester to sleep once the harvester hits the limit, and re-issues request once the limit is lifted. For example, at the start of a request cycle, the stop watch will start to record the time. If the process harvests 180 tweets within 3 minutes, the process will be put to sleep for the remaining 12 minutes. When the time hits 15 minutes and a new cycle could start, the stopwatch resets to 0 and the process will re-connect to Twitter API.

- **Authentication Error**

During implementation, it is noticed that occasionally Twitter API will return HTTP 401 error due to errors in reading authentication. Therefore, a re-connect mechanism is implemented to handle errors in establishing the channel.

4.1.4 CouchDB Connection

In the data saving step, the harvester needs to connect to CouchDB via HTTP request. The system will first check the validity of the CouchDB address, and that the credentials have the required access to write to database.

When saving to database, the harvester first checks if the data table exists, and if not, a new table will be created to hold the incoming tweets.

In case of connection disruptions, a re-connect mechanism is put in place to automatically re-issue connection request to CouchDB.

4.2 Database Fall-over

The database is replicated on multiple volumes. And instance 1 and 2 are used mainly for powering data analysis. Therefore, in the code design for data analysis and processing module, the connection request will fall over to the other instance, should the first instance failed.

4.3 System Scalability

Server Setup and Code Deployment

Boto and Ansible scripts are used to launch and configure new instances. With trivial effort, multiple instances can be launched on NeCTAR, and the required installation will be configured in place.

Ansible script is used to deploy the code to target instances, and start the application.

Data Harvester

Data harvester module is designed to handle duplicates, which means it could easily scale up, by deploying to multiple instances and adjusting the search parameter.

To tackle the bottleneck in data pre-process before saving to database, a queuing system is implemented to handle large incoming flow.

CouchDB

The usage of CouchDB cluster allows the system to easily scale up. The system adopts a setup of 3 nodes with a sharding of 4, which distributes the data load and its replica among the nodes. And the full data can be accessed from any node within the cluster. In the need to handle larger incoming data load, a new node could be easily added to the cluster, and CouchDB will automatically rebalance the sharding and replica.

5 Project Critical Analysis

5.1 NeCTAR Platform

Advantages

- Support both dashboard and script: The dashboard of NeCTAR is pretty straight forward and easy to use, which enables even non-technical personnel to create instance/volume and attach volume. Meanwhile, it also supports script to manage instances, which is convenient for large number of server maintenance.
- Flexible configuration: NeCTAR provides a wide range of options for server configuration, and booting source, allowing great flexibility in server setup.
- High scalability: It is relatively easy to add new volumes and new distances, which means the system scales up quite easily and fast.
- Built-in Security Setup: NeCTAR provides a range of common firewall options, and allows customisation on the security rules.

Disadvantages

- Service outage: NeCTAR platform is not perfectly stable. It is quite common for fired requests to receive time-out error, especially during system peak time.
- Unable to adjust the capacity for live servers: During the analysis phase, there is a need to increase the capacity of one instance to support the heavy processing load. However, the team failed to find an easy way to increase the CPU for a live instance.

5.2 Data Filter and Duplicate

5.2.1 Tweet Filter and Pre-processing

As the analysis is conducted by on geographical level, and restricted within Australia, not all tweets can be used in the system. Generally speaking, only two types of data can be imported to the system:

- Tweets with geo bounding box and location tagged within Australia, which are used for area analysis and comparison.
- Tweets with coordinates that can accurately pin point the location, which are used for population and individual activity analysis.

Therefore, a filter mechanism is imposed to the data harvester to discard non-relevant tweets (e.g. missing critical geo information), and only save the data that can contribute to the application.

Furthermore, as the application only utilise limited number of fields in tweets (e.g. text, account information, etc.), blindly storing a complete tweet would be unnecessary and resource consuming. Therefore, the incoming tweets are re-constructed to only retain the relevant information before saving to database. For tweets that misses certain fields, a default NONE value will be filled in, and further filtering will be done in Map-Reduce step according to analysis need.

5.2.2 Tweet Duplicate Pruning

As the harvester is running independently on multiple instances, there is a high probability that the harvested tweets are duplicates. Therefore, solely relying on the auto-generated CouchDB document ID is not sufficient for duplicated data pruning.

A two-step approach is deployed to address this issue. First, the Rabbit MQ queuing system is installed to pipeline the incoming tweets, which solves the duplicates rising from concurrent writing. A second check is put in place, utilising the unique message ID generated by Twitter's Snow Flake. If message ID of the incoming tweet already exists in database, that tweet will be discarded as duplicated data.

5.3 Data Analysis

5.3.1 Sentiment Analysis

Approaches

To classify positive tweets and negative tweets accurately, we looked through papers of sentiment analysis published in recent years. And we found the state of the art approaches are mostly based on the neural network models. In this project, we referred the paper. A Convolutional Neural Network for Modelling Sentences. In the paper, the authors proposed a novel approach Dynamic Convolutional Neural Network (DCNN). This network model can analyse and represent the semantic content of a sentence with various length. We utilized this approach and its source code(https://github.com/xiaohan2012/twitter-sent-dnn) for sentiment classification of tweets.

Training Datasets and Parameters

Sentiment140(http://help.sentiment140.com/api) was used to train the network model. This open source dataset contains about 1.6 million tweets extracted using the twitter API. The tweets have been annotated and they can be considered as training data or test data for experiments. There are a set of hyper-parameters in the model, such as layer number, batch size, etc. To tune the parameters, grid search was used. Here, we used the model with following parameters: batch size (10), convolution layer number (2), k values (20,8) (values correspondence are from bottom to top layers, applies to the following entries), filter width (8,6), L2 regulariser parameter (1e-06,1e-06,1e-06,0.0001), feature map number (0.5,0.5), dropout rates (0.5,0.5).

Results Analysis

We applied the model to predict sentiment polar of live tweets. Its output is a score from 0(negative) to 1(positive). In our project, at the beginning, we assigned 0.5 as a threshold. If a score of a tweet is larger than 0.5, we consider as a "happy" (positive) tweet, vice versa. Even though the accuracy of DCNN model to classify sentiment of tweets shown in the paper is about 87.4%, which is pretty high, we found the output scores are always a little lower than our manual classification by intuitive judgment for most live tweets we extracted. Thus, we assigned the threshold as 0.4 for better performance generally.

For implement of this part, we developed multi-processes programs in Python. We analysed 61,0000 tweets in 39m26.042s with 4 cores.

5.3.2 Topic Classification

Models

fastText is a library for efficient learning of word representations and sentence classification, proposed by Facebook Research. We use this algorithm to train the topic classification models.

Basically, fastText first trains continuous word presentations based on general skipgram and cbow models, adding rich n-gram subword information. and then employs linear classifiers

with a rank constraint and a fast loss approximation, which can train huge dataset within a few minutes. Hierarchical softmax and neural networks are also employed.

Training Dataset and Parameters

The second part is to find meaningful training dataset. However, we come across some difficulties, that we cannot find annotated twitter dataset for topic modelling. Therefore, we adopt some short text with broad topic labels (77 topic labels) to train our models. The size of the training set is more than 7000 with the vocabulary of around 23000 words. Next step is about parameters setting and tuning. Here is our optimized parameters configuration.

./fasttext supervised -input train.txt -output model -minCount 5 -wordNgrams 10 -minn 3 -maxn 10 -lr 0.01 -dim 300 -ws 5 -epoch 10 -neg 10 -loss softmax -t 0.001

Results Analysis

After the training process, we use our topic model to predict topic labels for scrawled twitters of a specific user. Finally, we compute the most frequent topics which the users are interested in, based on his or her previous twitters. However, because common short texts from documents or websites (what we use as training data) are different from twitters in terms of format and content, our topic classification models may not achieve high accuracy in twitter topic classification. Topic classification for twitters is challenging.

5.4 CouchDB

- **JSON Format Data Allow Simple Retrieval**

Under the design of document storing structure, retrieving document from CouchDB is fairly simple, since data is stored in JSON format. With the exception of user defined data, all the other changes of replica will also be tracked and recorded in the database. This approach improves the readability of documents and accessibility of content section. However, space consumption increases significantly, as JSON format requires more space.

However, data retrieving from CouchDB has a relatively long latency, which in the system, is mitigated by the CouchDB View.

- **CouchDB View**

The view serves as an index for the raw data, and provides a convenient and fast approach for data retrieval. However, CouchDB View consumes large portion of memory resources. When a new view is built, the corresponding B-tree is created in file for storing all the data specified in view definition. In some cases, it is noticed that the files for views are even larger than the original database.

- **Simple Retrieve Method**

Retrieving data from CouchDB could be as simple as accessing a URL address. This straightforward method clearly express from which database or documents are we fetching the data and what parameters have specified for limiting retrieved data.

- **Built-in Map-Reduce for Relational Data Extraction**

As CouchDB is NoSQL database, it would require extra steps to explore relational data. The built-in Map-Reduce functions of CouchDB simplifies relational data extraction.