Cluster and Cloud Computing (COMP90024) Project Report

Team 28
Ruihan Zhang 865529
Linrong Chen 854645
Ming Yin 816159
Hongyun Ma 805266
Jinge Zhang 769474

May 2018

1 Introduction

The project aims to achieve a one-click deployment on a cloud system, collecting tweets and data from Australian Urban Research Infrastructure Network (AURIN) around cities in Australia, doing sentiments analysis and eventually visualizing scenarios on a web application to let people discover more stories about Australian life.

Four instances from Nectar Research Cloud are used as resources. Boto was used to create instances. The resource allocation is going to be explained in detail in following paragraphs. Tweets are stored in MongoDB, which is a kind of document database mainly used for big data storage and management. It is "NoSQL" conceptually. Tweets and AURIN data are in JSON format, which inherently suitable to be stored in MongoDB. Additionally, MongoDB is currently used worldwide, more popular than CouchDB. Furthermore, the database in MongoDB is distributed. For tweets harvester, what we use is tweepy, a python library for accessing the Twitter API. By using the id of tweets as a unique identifier for each document, tweets duplication can be handled at the beginning. For sentiments analysis, we use the deep learning method LSTM to classify tweets into three classes: positive, negative and neutral and it performs well. We also implement Navie Bayes as an alternative choice. In this stage, a new key is inserted into each document to identify its sentiment.

Datasets from AURIN are mostly statistical demographic information, which is an important reference for scenarios.It provides information to comparing degrees of education, health, salary etc among areas. What we use is SA2 databases, which are databases based on areas similar to "neighborhood" place level in Twitter API.

As we worked in a cloud environment, resource allocation and isolation should be automated. Docker is deployed in this case because it is a containerization technology used as a container which plays a role as an operating system. Ansible is used to support the one-click deployment. We have written scripts as YAML Files. Additionally, we develop several methodologies to handle different levels of errors.

In terms of analysis, we have designed several scenarios, which will be explained in detail later. Results are presented in a series of web pages. Map is made by Google Map API and statistical graphs were drawn by chart.js. MapReduce is used here to generate data shown on the website.

2 Architecture

2.1 Nectar

Our application is built on Nectar[1] research cloud, which is an abbreviation for National eResearch Collaboration Tools and Resources project. Nectar is an online cloud computing platform which provides infrastructure-as-a-service to Australian research community. Nectar is based upon OpenStack[2], an open source cloud computing platform. With support from Nec-

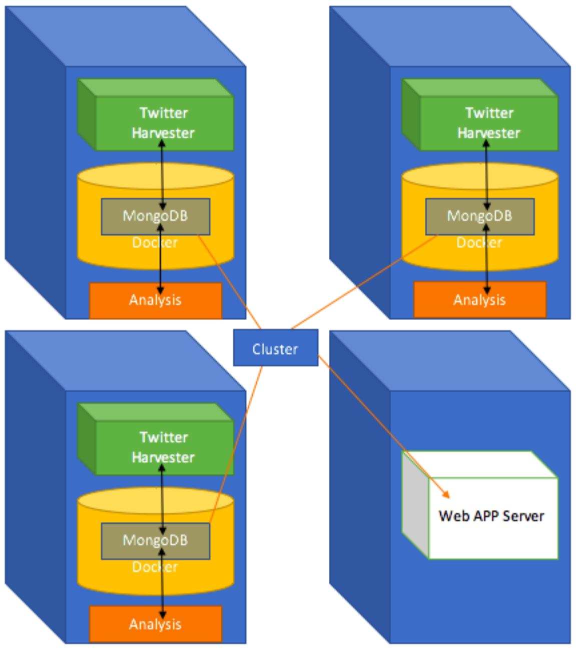

Figure 1: System Architecture

tar, we can easily set up virtual machines from existing images to harvest and analyze data in the cloud.

2.2 Relationship

An appropriate architecture can make the system be easy to use and maintain. Besides, the rational allocation of resources makes the whole system more efficient and more fault-tolerant. The whole system architecture is shown in figure1.

There are four instances in our system, and three of them are as a cluster, and the fourth one is as a web app server using data from the cluster's database to respond user request. In the cluster, these three instances have the same structure. Each of them has a Twitter harvester, a sentiment analyzer and a MongoDB database instance built within Docker. First of all, the twitter harvester retrieves data from Twitter continuously and stores them into MongoDB. Then, the analysis program gets data from the database and processes the data to classify each tweet into positive, negative or neutral sentiment and adds an extra tag called sentiment and store it back to the database. Finally, the web app server will use data from MongoDB directly or get data calculated from MongoDB built-in MapReduce method and show the results to clients.

2.3 Allocation

Before the start of the system, all resources should be allocated rationally and properly. There are four instances in total. For each instance, two virtual CPUs, six gigabytes memory and one volume are allocated. And, every volume has fifty gigabytes capacity. In order to make the whole system to be safe, there is only SSH connection allowed with RSA authentication. Moreover, the availability zone is only on the scale of Melbourne. As the figure 1 showing, each blue box is an instance on the Nectar cloud platform and in the first three instances, green boxes stand for Twitter harvesters and orange ones are the analysis programs, and yellow ones are regarded as Docker program and in each Docker program, there is a MongoDB instance. In the last instance, the white box means a web app server.

3 Methodologies

We divide the whole project progress into three stages. In the first stage, we setup MongoDB with collected tweets. Getting enough collections of tweets is a milestone. Second, we apply sentiment analysis on tweets gathered and use MapReduce to process and analyze sentiment-classified tweets for further study. AURIN data is also studied at this stage. In the last stage, we present our analyzed data in the form of the web application for visualization.

3.1 Twitter Harvester with Tweepy

After establishing the running instance and configuring environment, the first and the most basic section is to retrieve data from Twitter. Our system uses a program called Twitter harvester to do so. This program calls official APIs[3] to get tweets in JSON data format. In order to be convenient, Tweepy[4], a Python wrapper library for offical Twiiter API, is applied in our Twitter harvester program. This library encapsulates all Twitter APIs and makes them very simple to be used. In our program, the most used functions are geo_search, search and stream. Around 50,000 tweets can be collected per day.

In order to make our program extensible, some parameters like host IP address, MongoDB collection, the place that we want to search and the Twitter access token which we want to use are not fixed. They are passed as arguments or just written in a setting JSON file.

3.2 MongoDB

The database choice of our application is MongoDB[5]. According to [6], MongoDB is the fifth most popular DBMS and the most popular No-SQL DBMS.

The main reason MongoDB is chosen is that it is a document-oriented database management system. It stores documents in JSON format, which is exactly the format that twitter API gives, makes it the ideal choice of our application. The second reason is that MongoDB is designed to store a large amount of data, which fits the amount of data nowadays in social media. MongoDB provides various ways of querying and processing data like Mongo-query, secondary indexes and MapReduce, makes processing a large amount of data flexibly and efficiently. The third reason is that MongoDB can easily be scaled horizontally. This feature enables us using auto-deployment strategies like Boto and Ansible to easily scale up our system.

3.3 Docker

Docker[7] is used together with MongoDB to support easy deployment. Docker utilizes containerization technology. Docker container is like a light-weighted version of the virtual machine. The difference is that virtual machines are built on hypervisors while containers run directly on Linux kernels. Only the application and related libraries and included in a container. However, the performance of application inside docker is almost same as running on host OS.

The advantage of applying docker is that once a container is built successfully on one machine, it would run on any machine regardless of host OS or hardware. This gives docker much power in cloud computing. By combining docker and MongoDB, our system can easily scale horizontally.

3.4 Sentiment Analysis

Two machine algorithms are experimented on sentiment analysis. The training data used in these two experiments are from NLTK[8] twitter corpus. It contains 5,000 positive tweets and 5,000 negative tweets.

3.4.1 Naive Bayes

Naive Bayes is a simple probabilistic based classifier. This classifier can distribute class labels which represent as feature values to an instance of a problem, to be specific the class labels are coming from a limited set. It is a series of algorithms based on the same principle: assuming that each feature has no relationship with other features in any samples.

This idea of Bayesian inference has been known since the work of Bayes, and was first applied to text classification by Mosteller and Wallace[9]. Naive Bayes classifiers make two simplifying assumptions besides estimating the probability of every possible combination of features. The first one is the position of a token is not matter in a document, the second one is the probabilities of feature i are independent given the class c. Hence, the basic

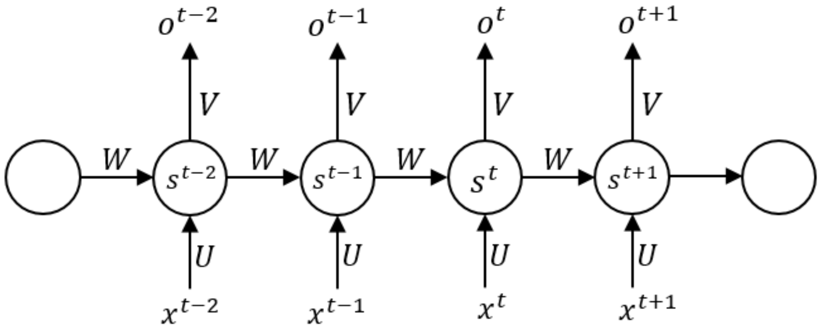

Figure 2: Vanilla Recurrent Neural Network

assumption is shown as follows:

$$P(f_1, f_2, ..., f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot ... \cdot P(f_n|c)$$

For training the Naive Bayes model, we used all words in NLTK twitter corpus[8] build the word bag."Add one" method is used as the smoothing method, which add a value of 0.01 to each of the 0 vector. Also we used the "English stopword" corpus in the NLTk to remove some very frequent words in English.

Finally, the accuracy is around 0.70, which is much worse than LSTM result.

3.4.2 LSTM

The second algorithm used is Long Short Term Memory(LSTM)[10]. LSTM is a particular architecture of recurrent neural network(RNN). Figure 2 shows how vanilla RNN works.

At each time step t, input x^t and previous cell state s^{t-1} are fed into the RNN cell. The output of the cell is calculated by

$$\begin{aligned} a^t &= g_1\left(U\, x^t + W\, s^{t-1} + b_1\right) \\ o^t &= g_2\left(V\, a^t + b_2\right) \end{aligned}$$

where g_1 and g_2 are activation functions. However, this kind of architecture suffers from vanishing gradients[11]. Every time the network parameters are updated, the gradients from final time step need to be back-propagated to each time step till the very first. In the process of such long matrix multiplication, the gradient propagated to early time step can be small enough that has almost no influence on network parameters.

In order to resolve this problem, LSTM introduces four gates on top of vanilla RNN, namely input gate, forget gate, output gate and cell gate. At each time step, the cell state and output are calculated by

$$\begin{aligned} \tilde{s} &= tanh\left(U_c\, x^t + W_c\, a^{t-1} + b_c\right) \\ \Gamma_i &= \sigma\left(U_i\, x^t + W_i\, s^{t-1} + b_i\right) \\ \Gamma_f &= \sigma\left(U_f\, x^t + W_f\, s^{t-1} + b_f\right) \\ \Gamma_o &= \sigma\left(U_o\, x^t + W_o\, s^{t-1} + b_o\right) \\ s^t &= \Gamma_i\, \tilde{s} + \Gamma_f\, s^{t-1} \\ a^t &= \Gamma_o\, tanh\left(s^t\right) \end{aligned}$$

where $tanh$ is hyperbolic tangent function and σ is sigmoid function. The cell gate calculates candidate of current cell state $\tilde{s}$ from input x^t and last cell output a^{t-1}. Input gate Γ_i is in charge of how much we should input from candidate cell state $\tilde{s}$. Forget gate Γ_f governs how much to forget on last cell state s^{t-1}. And finally, output gate Γ_o decides how much to output.

Before training our LSTM model, pre-processing on twitter text needs to be done. Since we cannot feed pieces of text directly into our model, tokenization needs to be applied to the text first. Due to the informality of Twitter texts, simple techniques like tokenizing by spaces does not work well. First, we defined several special tokens like ⟨url⟩, ⟨url⟩, ⟨hashtag⟩ etc. to represent common patterns in twitter texts. Many people tweet with expressions like

:) or XD, which are strong signals of tweets' sentiment. The second thing we do is define families of punctuation to represent eyes(:=; etc), noses('/- etc) and mouses(pD etc). Again, we define special tokens for expressions, like ⟨smile⟩, ⟨sadface⟩, based on different characters in mouses family. After that, regular expressions are used to catch these patterns and transform them into special tokens. At last, punctuation that does not match pre-defined expressions is separated by spaces. Therefore, we can perform tokenization by split processed texts by spaces.

However, tokenization is still not enough for our model. What we need is a method to turn tokens into vectors so that we can do mathematical operations on them. The simple solution is to use vector space model, which turns a token into a vector of length $|V|$, where V is the size of vocabulary. Vector space model gives one for the entry that corresponds to that specific token and zero anywhere else. However, this method makes the vector space unnecessarily huge which costs a lot of memory and computation resources. It's also hard to decide the vocabulary size for unseen data.

Instead of using vector space model, we decide to use word embeddings. Word embeddings are compact word representations that are usually trained on some toy tasks[12]. The embeddings are hidden representations of these tasks. These word embeddings have been shown to be able to capture syntactic and semantic similarities between words[13]. The specific word embedding of choice of ours is GloVe[14]. GloVe takes both word co-occurrence and frequencies into account and tends to have superior results in capturing word similarities. We use a pre-trained GloVe word embedding with 50 dimensions, which was trained on over 2 billion tweets. Because of the word embeddings' ability to capture word similarities, we are able to generalize our model to unseen data and use it to perform sentiment analysis on tweets we gathered. Part of previous pre-processing are performed to match the same pattern that GloVe vectors are trained.

Our model is built in TensorFlow[15]. It consists of a single layer LSTM with hidden state dimension of 128 followed by a single fully connected layer. In order to perform batch training, we set the length of each tweet to be 30. Tweets with more than 30 tokens are cut to 30 and tweets less than 30 are zero-padded to 30. For every training sample, we keep a mask m on the length of each tweet. After feeding training data into LSTM cell, the m^{th}

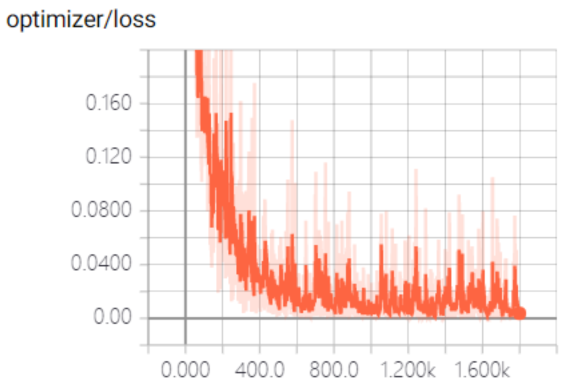

Figure 3: loss change over training phase

output(a^m according to previous notation) is extracted and fed into a one-layer fully connected network. Softmax operation is performed on the final output to turn it into a distribution over sentiment classes. Finally, we use cross-entropy loss to measure the difference between model output and labels and perform backpropagation and Adam optimization[16] to update network parameters(automatically).

After 10 epochs of training, our model converges to an optimal point. As shown in figure 3, the loss goes down to about 0.01 after about 1,800 steps. Figure 4 shows the change of accuracy during training. Our model reaches about 99.58% accuracy after training. On test set, our model reports an accuracy of 99.2%.

With the performance of the LSTM model, we are ready to put it into production. However, there is still some work need to be done. The training data consists of only positive and negative tweets. Meanwhile, a tweet can also have a sentiment of neutral, which is neither positive nor negative. An assumption made here is that neutral sentiment lies between positive and negative. Therefore, we can set some confidence interval to the softmax output to predict three classes. After running the model on tweets we gathered, we find that our model is a little over optimistic. It predicts most neutral tweets as positive. Finally, we experiment with real tweets and find that a positive confidence interval of 99% to 1 and a negative confidence interval of 80% to 1 performs well in practice.

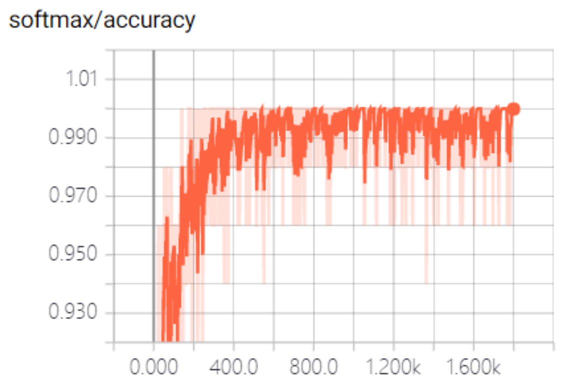

Figure 4: accuracy change over training phase

3.5 AURIN

Established in June 2010, the Australian Urban Research Infrastructure Network (AURIN)[17] is an initiative of the Australian Government under the National Collaborative Research Infrastructure Strategy (NCRIS) and associated programs. There are many statistic datasets provided by academic, government and private sectors. We want a statistic value of every neighborhood level area to estimate the degree of population, income and education from AURIN.

Australian Statistical Geography Standard is a standard from Australian Bureau of Statistics (ABS). It defines the standard for dataset provided by ABS. Statistic Area Level actually set the details for geographical areas for each row of the dataset from ABS. Here we want the neighborhood level area statistic data so we choose the Statistic Area Level 2.

SA2 are medium-sized areas built up from whole Statistical Areas Level 1. It generally contains a population of 3000 to 25,000 persons and an average of 10,000 persons. It is highly coincident with the “neighborhood” level places in twitter system. But actually, they are different in many ways and we need a new framework to show both AURIN data and tweets data on one map.

3.6 Datasets and Estimate Methods

We want some measures for degrees of population, income and education.

For population, we gather data from "SA2 Data by Region - Population & People 2011-2016"[18]. It contains two measurements, density and median age. Density can reflect the lifestyle in this area. Median age tells us the acceptance for Twitter in this area (Here we assume that the older a person is, less likely for him to accept Twitter).

For income, the data is from "SA2-G02 Selected Medians and Averages-Census 2016"[19]. It provides a median weekly income in dollars, only applicable for people over 15. It actually can show the life quality for people living in this area

For education, it's hard to measure but we use a simplified method to roughly estimate its level for every area. The data are from "SA2-G01 Selected Person Characteristics by Sex-Census 2016"[20], contains the number of people with the highest level of education completed. It only applicable for people over 15 and contains 5 different levels for people completed the 8-12 years of study (8-year means 8 or below and 12 years means 12 or above). We assume that there's no below or above in this Dataset and calculate the average years of education for each area as the measurement for education.

3.7 MapReduce

MapReduce is a method to process massive data separately and then aggregate the result together. In our system, MongoDB has a built-in MapReduce operation command[21]. Codes below shows the detailed operation.

```
mapper:
function() {
    var td = {}
    for (var j = 0; j < 24; j++)
          td[j] = {"positive":0,"negative":0,"neutral":0}
    if (this.sentiment != undefined)
```

```
        td[parseInt(this.created_at.split(' ')[3]
            .split(':')[0])][this.sentiment] = 1
    emit(location, td);
}

reducer:
function(key, values) {
    td = values[0]
    for (var i = 1; i < values.length; i++)
        for (var j = 0; j < 24; j++){
            td[j]["positive"] = td[j]["positive"] +
                values[i][j]["positive"];
            td[j]["negative"] = td[j]["negative"] +
                values[i][j]["negative"];
            td[j]["neutral"] = td[j]["neutral"] +
                values[i][j]["neutral"];
        }
    return td;
}
```

3.8 Google Map API

In our first scenario, we want to show some data on a Google map on the website. For heatmap, we use *HeatmapLayer* in *google.maps.visualization* library to generate a heat map layer on the google map. For another dataset, AURIN statistic data, we use *Polygon* in *google.maps.Data* to generate a new polygon with different opacity to represent the statistic value of an area on the map.

3.9 Visualization Tools

Chart.js is used as a tool to draw statistical charts. It helps us display charts on analyzed twitter data.[22].

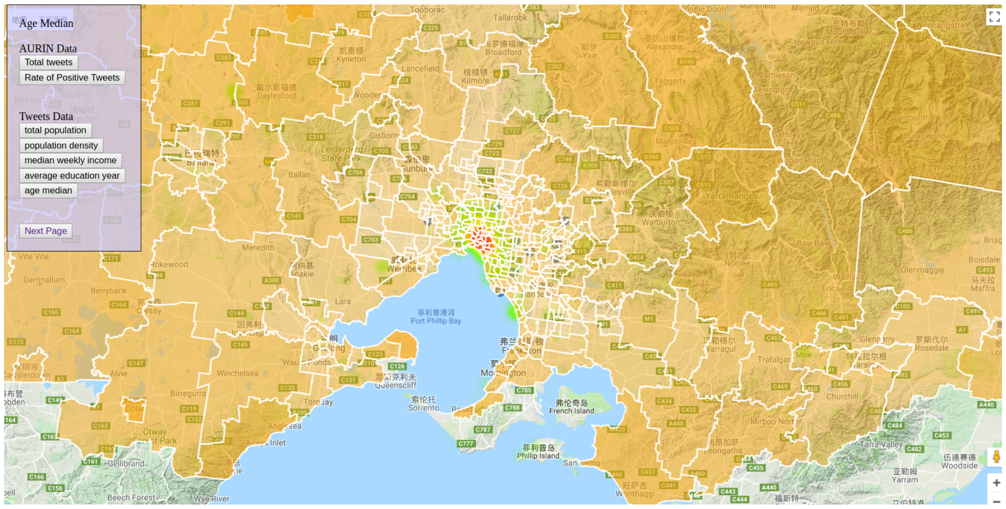

Figure 5: Median Age of Melbourne Area

4 Scenario

4.1 Introduction

Our system includes multiple scenarios, many of which show the comparison between positive tweets rate in each region and local demographic information. We can find some implicit correlation like salary and happiness etc. The result will be displayed on a map. Another one is revealing the positive tweets rate with regard to the time series. From it, we can know people's emotion change over time. At last, we demonstrate general twitter sentiment distribution in different regions.

4.2 Scenarios 1: Comparison between SA2 AURIN Data and Tweets

In this scenario, we show a map with two layers on the website. One layer is a heat map layer contains the data from tweets (including sentiment analysis and counts). The other layer is statistic information of Victoria from AURIN database(including population, population density, income level, education

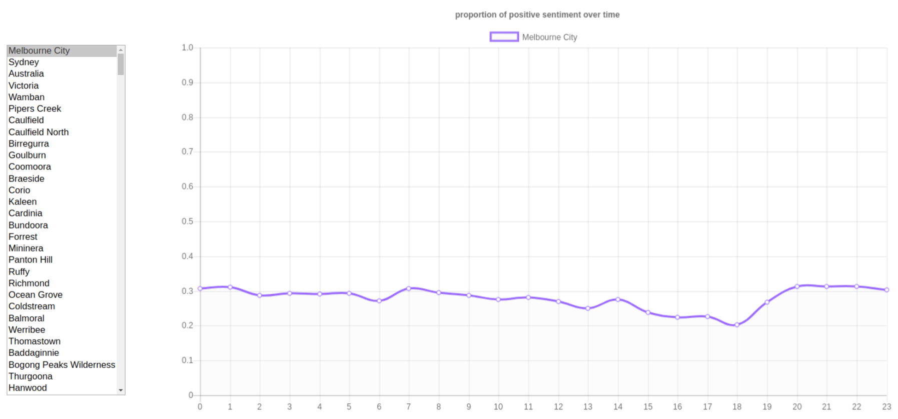

Figure 6: Proportion of Positive Tweets Over Time in Melbourne

and median age) that can be chosen from. Different opacity reflects different levels of statistic data. Figure 5 shows the median age of Melbourne suburbs. From this scenario, we can get the relationship between tweets and areas. For instance, it seems like the order the median age of an area is, the fewer tweets submitted per day.

4.3 Scenarios 2: Charts of Twitter Data

An example of time series charts is shown in figure 6. It shows how the proportion of positive tweets changes over time. On the left is a list of areas that can be chosen from, ranging from the entire area like Victoria to specific suburbs.

Figure 7 shows sentiment distribution of tweets in Victoria. Similarly, this can be chosen from large areas to specific suburbs. Overall, the sentiment distribution across different areas is roughly the same.

5 Error Handling

Our system is designed to handle both hardware failure and program errors.

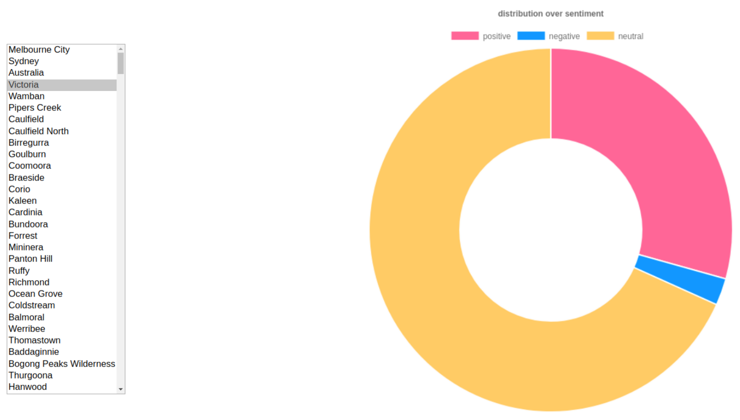

Figure 7: Distribution of Tweets' Sentiment in Victoria

5.1 Database Replication

In order to deal with potential network failure or even instance failure, we replicate our database to three instances. Therefore, we have three copy of our data across three instances.

Following the replica set rule of MongoDB, one primary node is in charge of reading and writing data and secondary nodes are able to read data. Once any unpredictable failure occurs to the primary node, secondary nodes would vote for a new primary node and our system can keep functioning.

5.2 Program Error Handling

To make our system more robust, some program error handling methods are applied. In each execution code part, there is much reason which can cause a program crash. For example, trying to open or read a non-exist file, inserting an existing item into the database, failing to connect a database and so on. However, "try... except" blocks are used to solve these issues. Furthermore, when the Twitter uses the access token to retrieve tweets, frequent requests may cause a limit error of Twitter API. To avoid this, in the Tweepy library, it already provides "wait_on_rate_limit" parameter to wait when the program hits the request limit rather than crashing.

Another situation is that all programs running on the Nectar may crash, so the solution we used is to set a ”watch” process to restart each program in a fixed interval.

5.3 Duplicate Tweets Handling

Since we have three instances harvesting tweets together, it is possible that different harvesters can get duplicate tweets. The approach we use to handle duplicate tweets is storing the unique id of each document in a database as the unique id of each tweet. Therefore, once a duplicate tweet comes in, the database would know and reject that tweet since it already exists.

6 Scalability

Beyond the current system we built, we can scale it horizontally to support larger scale twitter harvest, analysis and data backup.

6.1 Boto and Ansible

We use Boto[23] to dynamically create instances at scale. Boto is a python interface for Amazon cloud services, but it has support for OpenStack as well. Therefore, we can utilize it to automatically create instances on Nectar Research Cloud at need.

After instances are created, we use Ansible[24] to automate configuration process. Ansible is an automation tool that used for configuring and managing multiple machines. It uses SSH to connect to remote machines and perform tasks. Remote machines are specified and grouped in an inventory file. Different tasks are specified in playbook files to perform on different machines.

In our scaling process, we use Boto scripts to create instances and dynamically generates corresponding inventory file and Ansible configurations. Then we use Ansible playbooks to automate software and dependency installation, database configuration, file uploading and program executing.

6.2 Use of Scripts

In order to automate our deployment process even further, we use a shell script to combine previous steps. We use one shell script to manage SSH keys, run Boto script to create instances and run Ansible playbook to configure remote machines.

In terms of our four main instances showed in figure 1, we also use shell scripts to manage program execution.

7 Conclusion

To conclude, in this project, a big data application that can harvest and analyze twitter data and be automatically deployed has been made. Technologies used include: Python, Boto, Ansible, MongoDB, Docker, Tweepy, TensorFlow and Javascipt. From algorithmic point of view, LSTM recurrent neural network is used to analyze twitter sentiment and MapReduce is used to process data.

From the result, we can easily know that the younger generation tends to generate more tweets. There is a positive correlation between the number of tweets and average education level within a region. Results show that the rate of positive tweets across Victoria are almost the same. People tend to be in the better mood early in the morning and in the afternoon. More negative tweets are shared during the midnight and people who haven't fall asleep at midnight may be worried and anxious or busy with their works. Data can tell stories and there are huge potentials to be discovered.

External Resources

A video on automated deployment: `https://www.youtube.com/watch?v=woZH8WmgbXo`

Source code for this project: `https://github.com/zhangrh93/team_28`

References

[1] National Collaborative Research Infrastructure Strategy. *Nectar.* URL: `https://nectar.org.au/`. (accessed: 04.05.2018).

[2] Rackspace. *Open source software for creating private and public clouds.* URL: `https://www.openstack.org/`. (accessed: 04.05.2018).

[3] Twitter. *Twitter APIs.* URL: `https://developer.twitter.com/en/docs/basics/getting-started`. (accessed: 06.05.2018).

[4] Tweepy. *API Reference.* URL: `http://docs.tweepy.org/en/latest/api.html`. (accessed: 06.05.2018).

[5] MongoDB. *MongoDB for GIANT Ideas — MongoDB.* URL: `https://www.mongodb.com/`. (accessed: 04.05.2018).

[6] Solid IT. *DB-Engines Ranking.* URL: `https://db-engines.com/en/ranking`. (accessed: 04.05.2018).

[7] Docker. *Docker-Build, Ship, and Run Any App, Anywhere.* URL: `https://www.docker.com/`. (accessed: 04.05.2018).

[8] NLTK Project. *Natural Language Toolkit.* URL: `https://www.nltk.org/`. (accessed: 04.05.2018).

[9] James H. Martin Daniel Jurafsky. "Speech and Language Processing". In: (2017).

[10] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. URL: `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

[11] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 6.2 (Apr. 1998), pp. 107–116. ISSN: 0218-4885. DOI: `10.1142/S0218488598000094`. URL: `http://dx.doi.org/10.1142/S0218488598000094`.

[12] T. Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *ArXiv e-prints* (Oct. 2013). arXiv: `1310.4546 [cs.CL]`.

[13] T. Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *ArXiv e-prints* (Jan. 2013). arXiv: `1301.3781 [cs.CL]`.

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: `http://www.aclweb.org/anthology/D14-1162`.

[15] Google. *TensorFlow*. URL: `https://www.tensorflow.org/`. (accessed: 04.05.2018).

[16] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *ArXiv e-prints* (Dec. 2014). arXiv: `1412.6980 [cs.LG]`.

[17] NCRIS. *Australian Urban Research Infrastructure Network*. URL: `http://aurin.org.au`. (accessed: 04.05.2018).

[18] Government of the Commonwealth of Australia - Australian Bureau of Statistics. *SA2 Data by Region - Population & People 2011-2016*. accessed from AURIN Portal on 2018-05-04. 2017.

[19] Government of the Commonwealth of Australia - Australian Bureau of Statistics. *SA2-G02 Selected Medians and Averages-Census 2016*. accessed from AURIN Portal on 2018-05-04. 2017.

[20] Government of the Commonwealth of Australia - Australian Bureau of Statistics. *SA2-G01 Selected Person Characteristics by Sex-Census 2016*. accessed from AURIN Portal on 2018-05-04. 2017.

[21] MongoDB. *Map-Reduce*. URL: `https://docs.mongodb.com/manual/core/map-reduce/`. (accessed: 06.05.2018).

[22] chartjs. *Chart.js — Open source HTML5 Charts for your website*. URL: `https://www.chartjs.org/`. (accessed: 06.05.2018).

[23] The Boto Project. *boto: A Python interface to Amazon Web Services*. URL: `http://boto.cloudhackers.com/en/latest/`. (accessed: 04.05.2018).

[24] RedHat. *Ansible is Simple IT Automation*. URL: `https://www.ansible.com/`. (accessed: 04.05.2018).