COMP90024 Cluster and Cloud Computing

The Deadly Sins: Wrath

Team 48

Wenqi Sun(928630), Yunlu Wen(869338), Fei Zhou(972547)

Pei-Yun Sun(667816), Yiming Zhang(889262)

Table of Content

1 Introduction

This is a group which aims at analyzing data from Twitter and Aurin using a cloud system. Eventually, we will tell a wrath story based on the data we collected.

There are 7 sections in this report. It includes an overall description of the software we build, a guide on how to deploy the system, discussions on the key components in the system, concrete analysis on the data we gathered and a debate on the pros and cons of the Melbourne Research Cloud.

2 System Functionalities and Scenarios

2.1 System Functionalities

The system we've built contains three major functionalities: Twitter harvesting, MapReduce based processing and web visualizations.

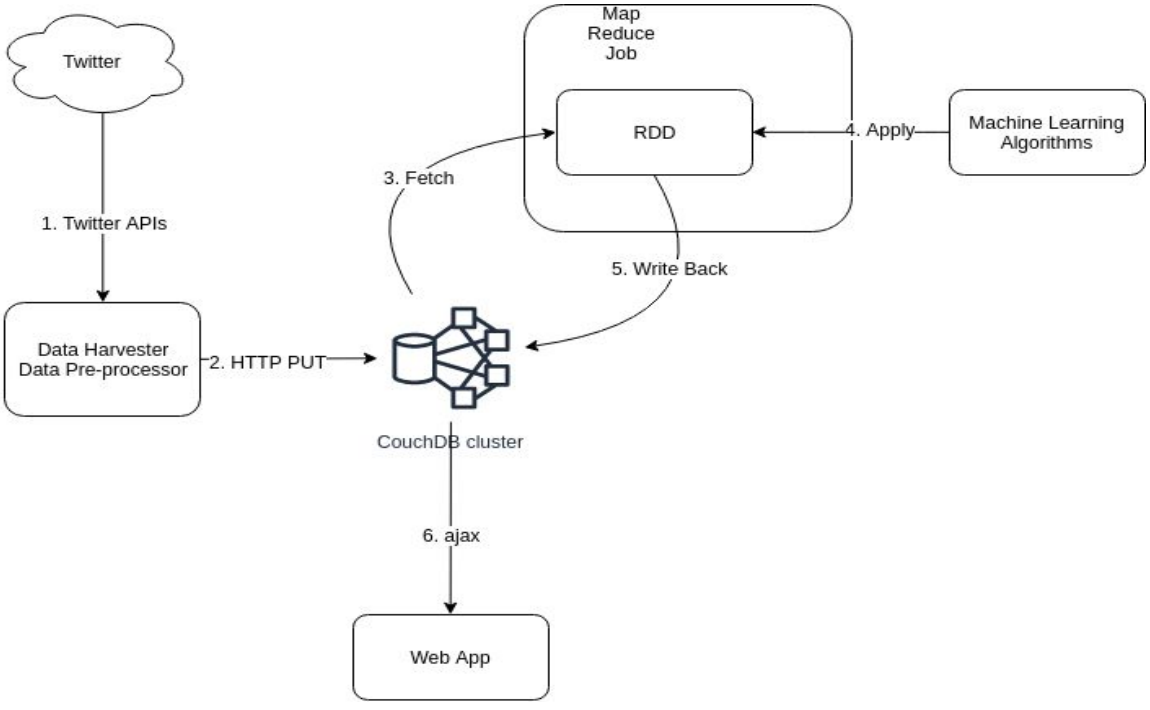


Figure 1: User Journey of the platform

- Twitter harvesting: As shown above, the data harvester uses Twitter APIs to harvest the tweets. After pre-processing, only the tweets with coordinates are stored into the CouchDB cluster.

- MapReduce based processing: The tweets stored in CouchDB are referenced and parallelized to create Resilient Distributed Datasets (RDD). Then, the machine learning model is used for doing sentiment analysis and the area that each tweet belongs to is found during the processing. The processed tweets are stored back to CouchDB.
- Web visualization: The result of tweet analysis is compared against the statistics from Aurin in each area. These comparisons and results are presented on the webpage.

2.2 Scenarios

Wrath is one of the Seven Deadly Sins, the team explores "Wrath" by harvesting tweets from the Great Melbourne Area and then apply data analysis techniques and machine learning algorithms to the data.

We have used multiple data visualization tools such as map, pie chart, bar chart to demonstrate the emotion and sentiment in each region of Greater Melbourne, and used a word cloud to find out the possible reasons of wrath.

In addition, we have also compared the tweet wrath rate that we calculated, with multiple datasets from AURIN, which includes homeless number, work overtime percentage, death percentage, median age, median income, etc. The comparisons indicate that wrath rate of each region has some correlations with work overtime, income, and unemployment, and has no correlations with other factors.

Finally, we have trained a Multiple Linear Regression Model that allow us the make predictions of the tweet wrath rate for a region given its AURIN data.

3 User Guide

3.1 System Deployment

3.1.1 Infrastructure Deployment

Infrastructure deployment includes creating security groups, creating key pairs, launching instances, creating and attaching volumes. This is accomplished by running ansible playbooks. In order to make everything look clean and consistent, all the components that need to be created or attached are defined in a JSON file, which is passed into playbook by setting fact_path in the setup stage. Then, several loops are used to iterate over all the resources that need to be created.

3.1.2 Software Deployment

Software deployment differs from infrastructure deployment in that, the software may require running setup script or even manual operations.

CouchDB cluster is deployed by:

1. Send configurations files and docker compose files to the server
2. Create the directory which is to be bind mounted to the database container
3. Launch single instances by running docker compose up
4. Setup administrator accounts and join nodes to a cluster.

When deploying CouchDB, the first three steps can be automatically finished by ansible playbook, while the fourth step requires some additional scripts from somewhere else.

Hadoop & Spark are deployed by:

1. Send configurations files and docker compose files to the server
2. Format the name node
3. Launch containers in docker swarm
4. Create ssh keys from the master node and copy them to slave nodes
5. Start dfs
6. Start Yarn
7. Start Job history server
8. Launch Spark master
9. Join Spark slaves to the master

The Hadoop/Spark cluster is even more complicated to deploy because it involves some manual operations. The reason is that the steps shown above do not always succeed. It is important that someone who's familiar with Hadoop executes the scripts at the master node and observe the logs/outputs then decide which step to take next. For example, if ssh copy script fails at a certain node, the status of the swarm should be checked to see whether the container is launched successfully at that node; If all the commands finished successfully, the Yarn and Hdfs logs should be verified if there is any error message presented. Sometimes it's necessary to test the functionality of the cluster by performing some file uploads to HDFS.

3.2 End User Guide

3.2.1 Map

We have used two choropleth maps:

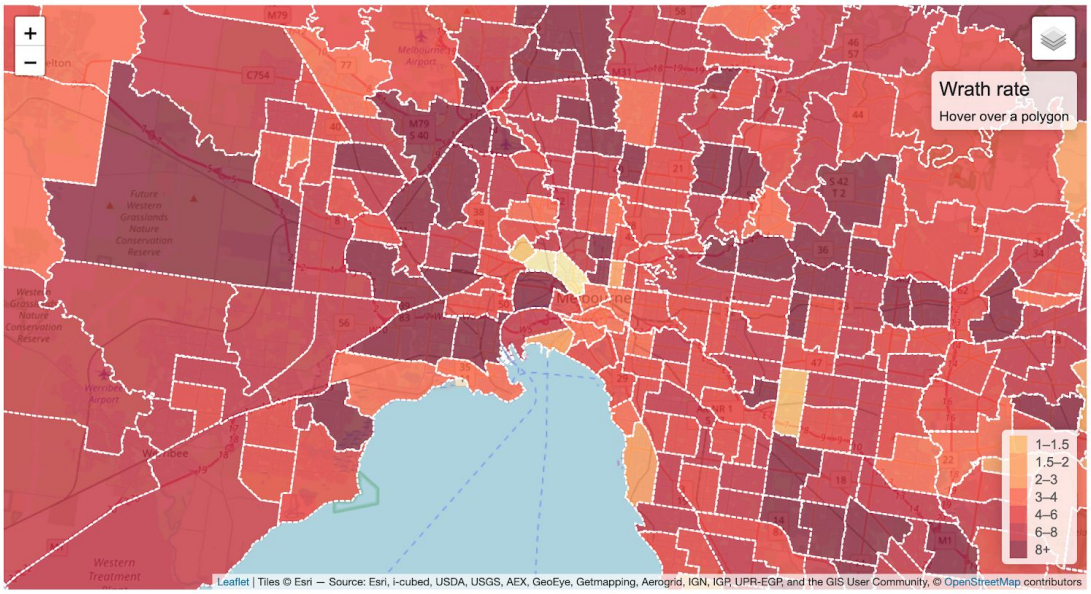
1. Tweet Wrath Rate (Red)
2. Average Tweet Polarity (Blue)

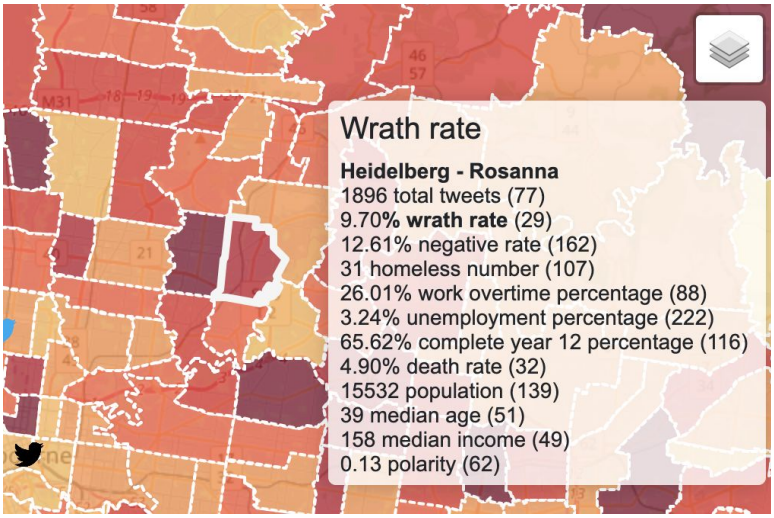
For average tweet polarity, we used the sentiment polarity of each tweet obtained by Textblob, with values -1 to 1, summed up all the polarities in each region, and divided by the total number of tweets in that region.

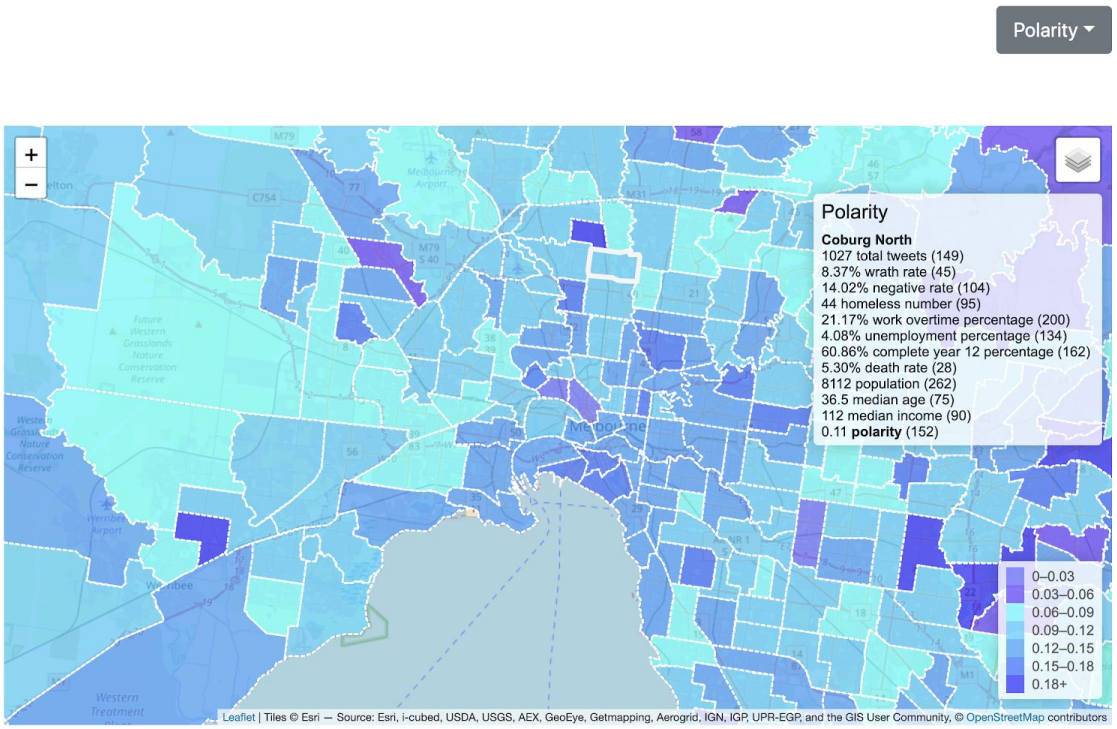
Besides that, our map also presented the statistics data of each region, including emotion, sentiment, and aurin data. Since the values are not straightforward for comparisons between the regions, the rankings of those values are added. For example, region *Hadfield* has 21% work overtime rate, which is ranked 210th among the 309 regions in Greater Melbourne.

Map

Wrath rate







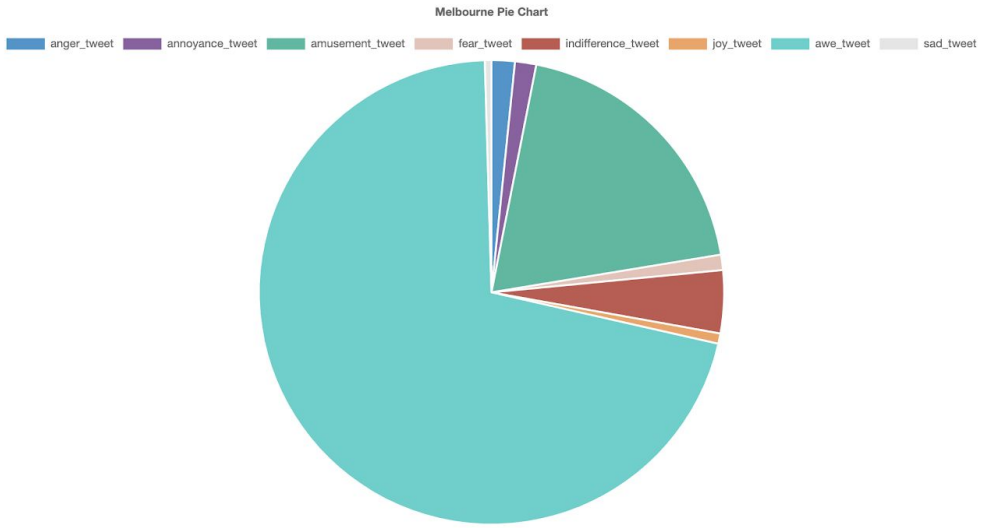
3.2.2 Pie chart and Bar chart

We have used pie charts to demonstrate the distributions of emotions, and use bar charts to present the distribution of sentiments in some regions: *Melbourne, Docklands, Southbank, East Melbourne, and Kensington*.

The emotion distributions seem to be similar among these regions (dominated by awe tweets), while the distribution of *Kensington* seems to be very different (dominated by fear tweets). From the bar chart, we can see that most of the tweets are neutral, and there are usually more positive tweets than negative tweets.

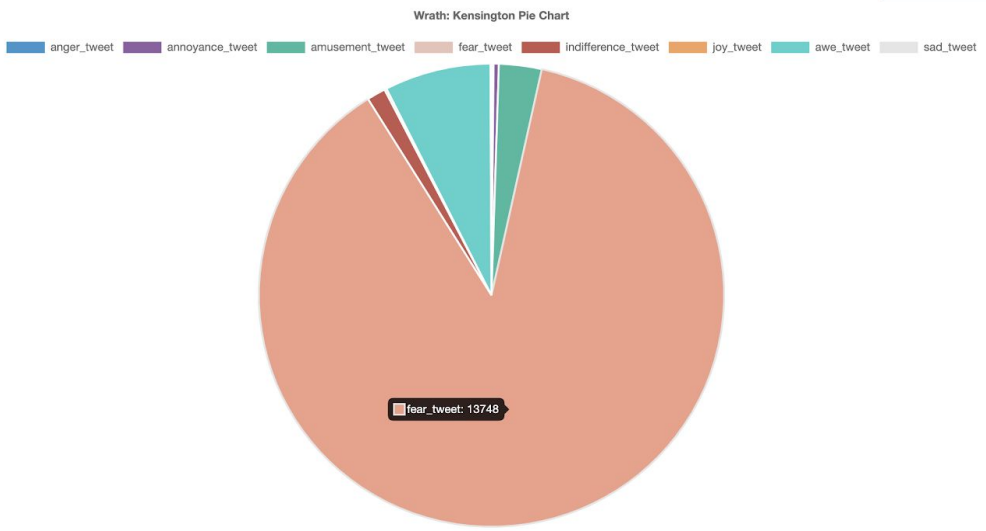
Pie Chart

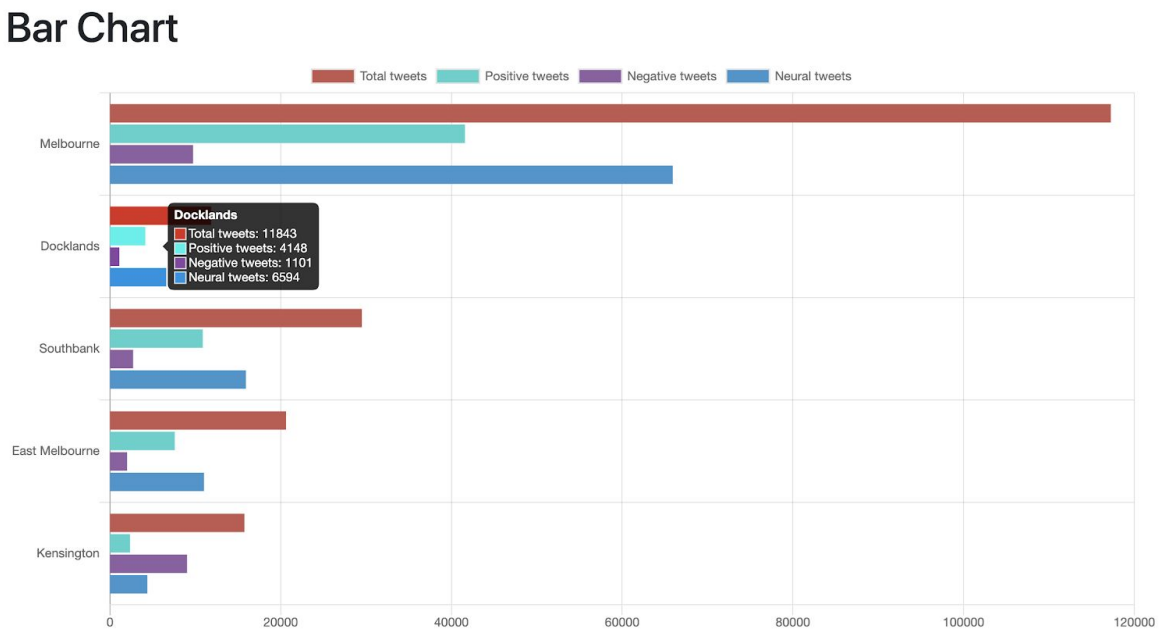
Melbourne



Pie Chart

Kensington





3.2.3 Word Cloud

We have generated a word cloud of the tweets to identify the reasons for wrath. The word cloud is generated by counted the word frequencies of the wrath tweets with text preprocessing, including lowercasing, removing stopwords, and lemmatization. We have also tried to use pos-tagging to keep only the nouns in each tweet, but the results are almost the same, and pos-tagging takes a long time, so we decided not to use it.

From the word cloud, we have found that the high-frequency words are: *victraffic, traffic, station, work, nbn, drinking, job, incident, city etc.* Therefore, we could know that people in Melbourne might be angry about things like:

1. Public transports: trams and trains usually delay, and public transportation could be seriously affected by strikes and protest march.
2. Internet: NBN in melbourne are usually very slow, it could even be affected by the weather sometimes (e.g. rainy days)
3. (etc.)

Word Cloud

3.2.4 Correlation

Aurin Feature	Correlation (with Wrath Rate)
Work Overtime Rate (> 40 hours/week)	**-0.3106**
Number of Homeless People	0.0691
Unemployment Rate	**0.2304**
Complete Year 12 Rate	-0.0762
Death Rate	0.0769
Population	-0.0010
Median Age	-0.0363
Median Income	**-0.2790**

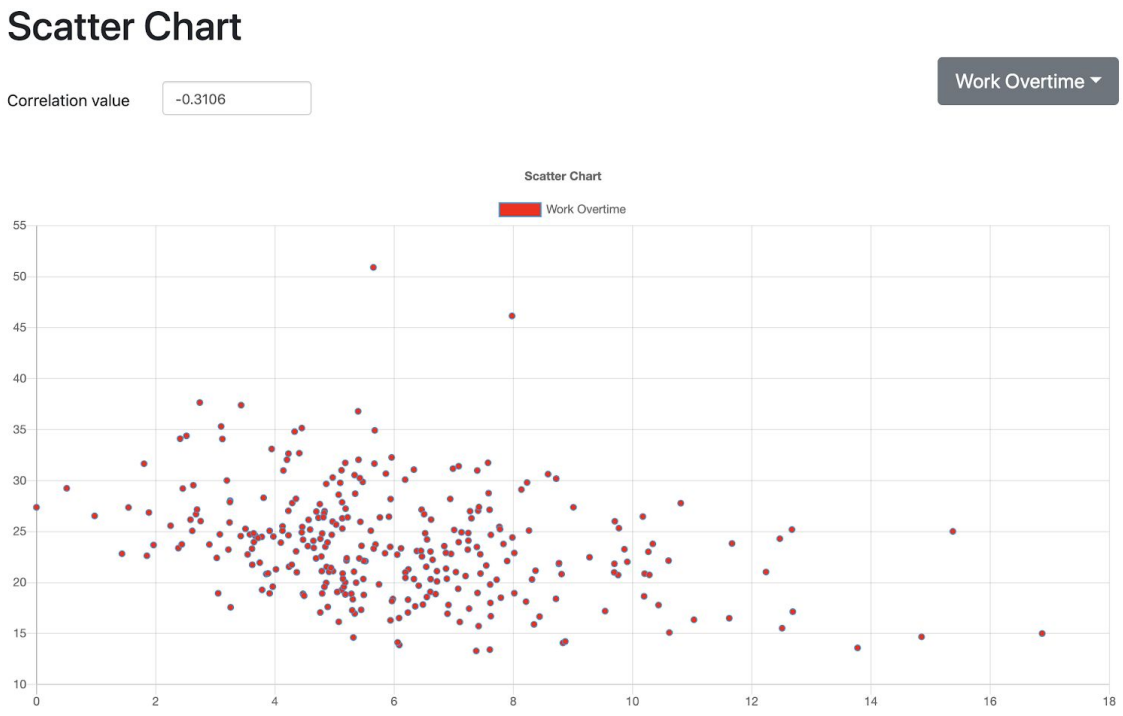
We have calculated all the correlations between the aurin features and the tweet wrath rate in each region, as shown above. In our front-end, scatter plots are used to visualise these relationships.

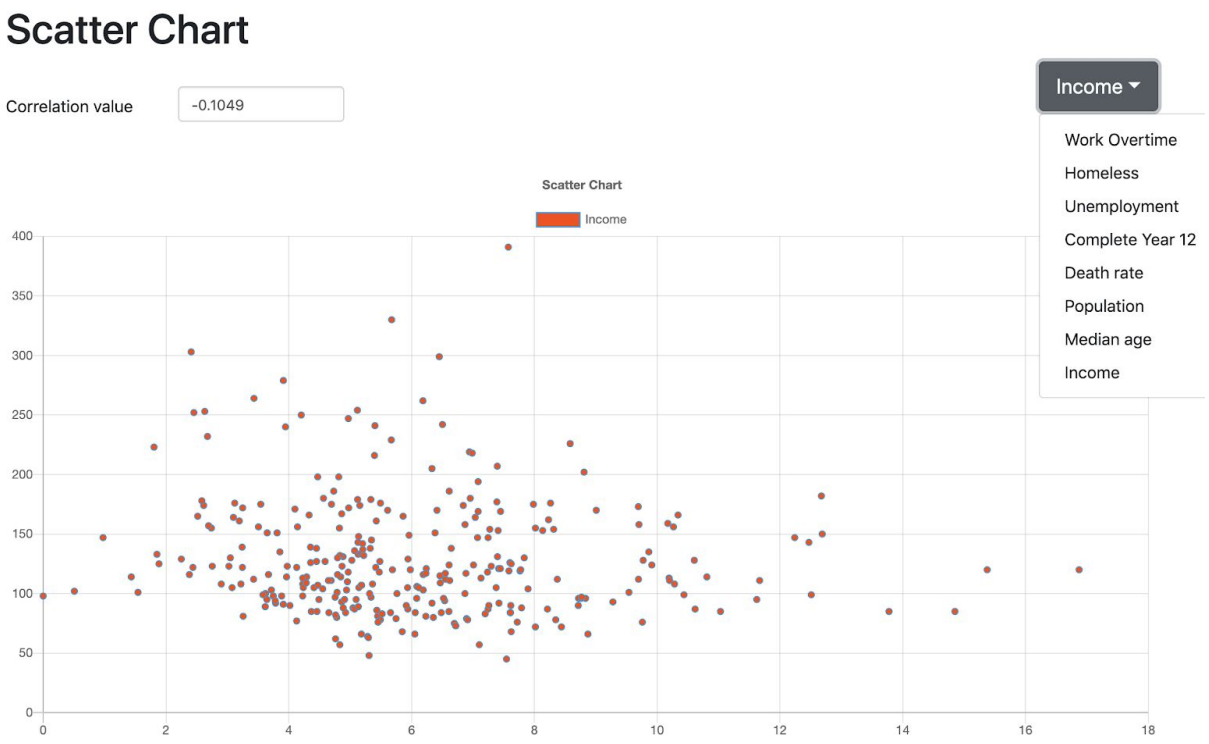
From the table above, the wrath rate has correlations with only: work overtime, unemployment, and income. Although the correlations are not very strong, this result still provides us with some indications to some extent:

1. Higher median income ⇒ Lower wrath rate

- Rich people are more likely to be satisfied with their life, and therefore less often to have wrath.

2. Higher work overtime rate ⇒ Lower wrath rate
    - The reason for this would be: people in Melbourne are usually not forced to work overtime. If they choose to, they will normally get the overtime pay, and therefore have a higher income. We have proved this explanation by calculating the correlation between work overtime rate and median income, and we found a very strong correlation: 0.7673 between them.
3. Higher unemployment rate ⇒ Higher wrath rate
    - People who lost their job or could not find a job would be more likely to have wrath.

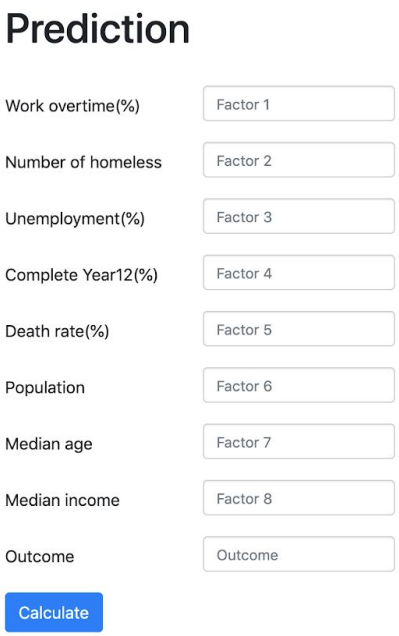




3.2.5 Prediction

We have used Multiple Linear Regression which could predict the (tweet) wrath rate for a region based on its Aurin features. Since multiple regression requires the features to be independent of each other, we have also used Principal Component Analysis (PCA) to construct the independent principal components, and then trained a regression model for predictions.

After fitting the Principal Components and the Multiple Linear Regression model, we can keep the means, coefficients, and intercept for calculation. These data are saved to the database and our prediction can be simply calculated by multiplications and additions, no longer relying on any external library.



4 System Architecture and Design

4.1 Overview

According to specifications of functionality in the previous section, the overall architecture of the platform can be determined. There is a CouchDB cluster which has 3 nodes to store a large amount of data into 3 replicas. Spark is required to perform some MapReduce and machine learning jobs. In order to get spark running, there should be a shared file system for workers from different nodes to access the submitted code, therefore an HDFS cluster is deployed as well. Finally, an Nginx server can be used to host the web service and do some reverse proxy and load balance.

The computational budget for this project is 8 CPU cores, 32GB of RAM and 250GB mountable storage. It was divided into a master instance and three slave instances. Master instance is used to deploy the master node of a different cluster, serves as the main entry point of different services at the same time. Slave instances have relatively higher storage capacity than the master instance, which takes over main storage and computation tasks.

Instance	CPU cores	RAM (GB)	Storage (GB)
master-instance	2	8	40
slave-instance-1	2	8	70
slave-instance-2	2	8	70
slave-instance-3	2	8	70

4.2 Single Instance Architecture

Instances in the platform have similar structures. Each instance has some default software which is essential for development and a docker daemon. All the services running in the instances are deployed using Docker. Some of them are running in docker swarm mode, and the other ones are deployed with plain docker container. There are facts which may affect the decision between docker swarm and container, which will be covered in detail in the following sections. The data and logs generated by containerized services are mapped to host storage using bind mount, so that the database and HDFS data can be persistent.

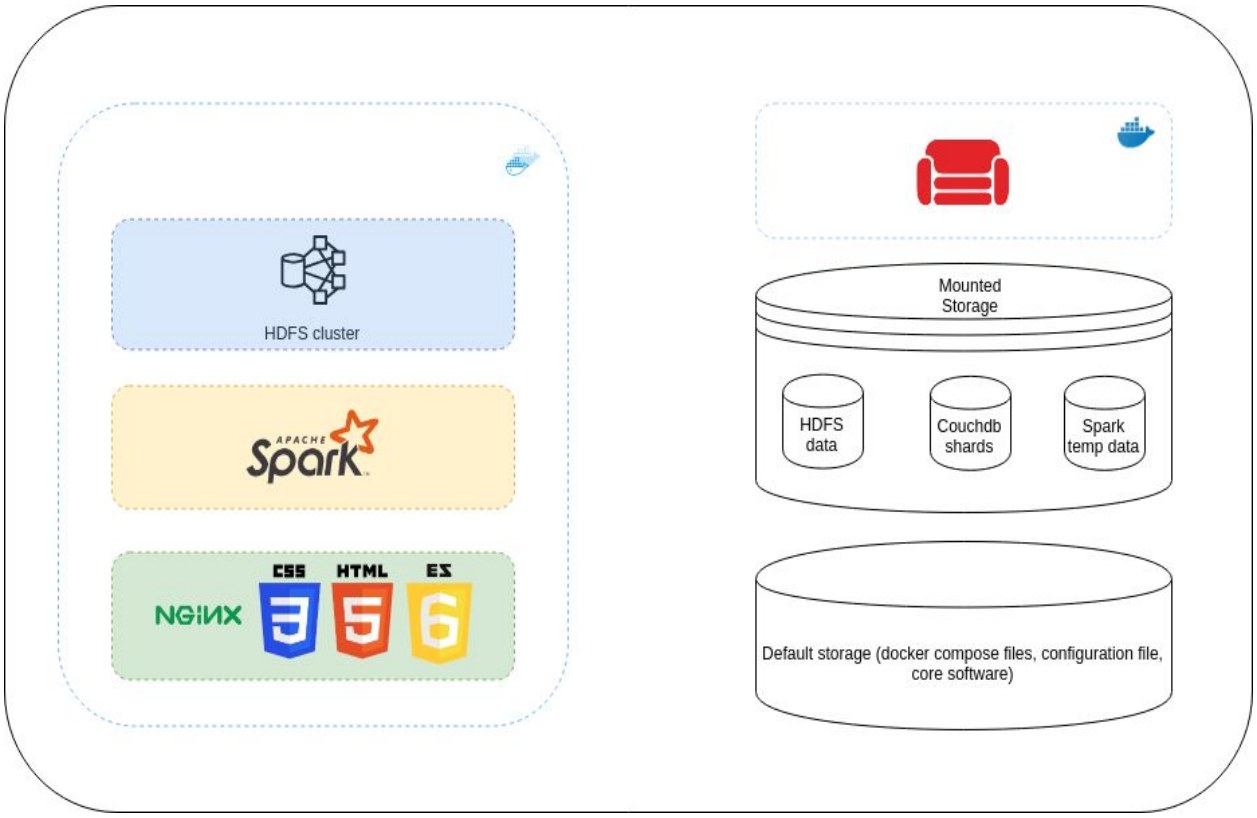


Figure 2: Internal architecture of the slave node

In each slave instance, there are HDFS data node, Spark worker a Nginx service running in swarm mode and a CouchDB container running in a separated container. The HDFS data blocks, Spark generated data and CouchDB data shards are bind mount to `/mnt` directory, where 70 GB external storage is mounted at.

The master instance is slightly different that Hadoop Name node and Spark Master are installed and the data only includes the user uploaded data. The user data includes MapReduce software written by the user which is to be uploaded to HDFS and some data fetched from Aurin. The user data directory can be mounted to Hadoop container so that the user can easily upload the data to HDFS without executing docker cp commands.

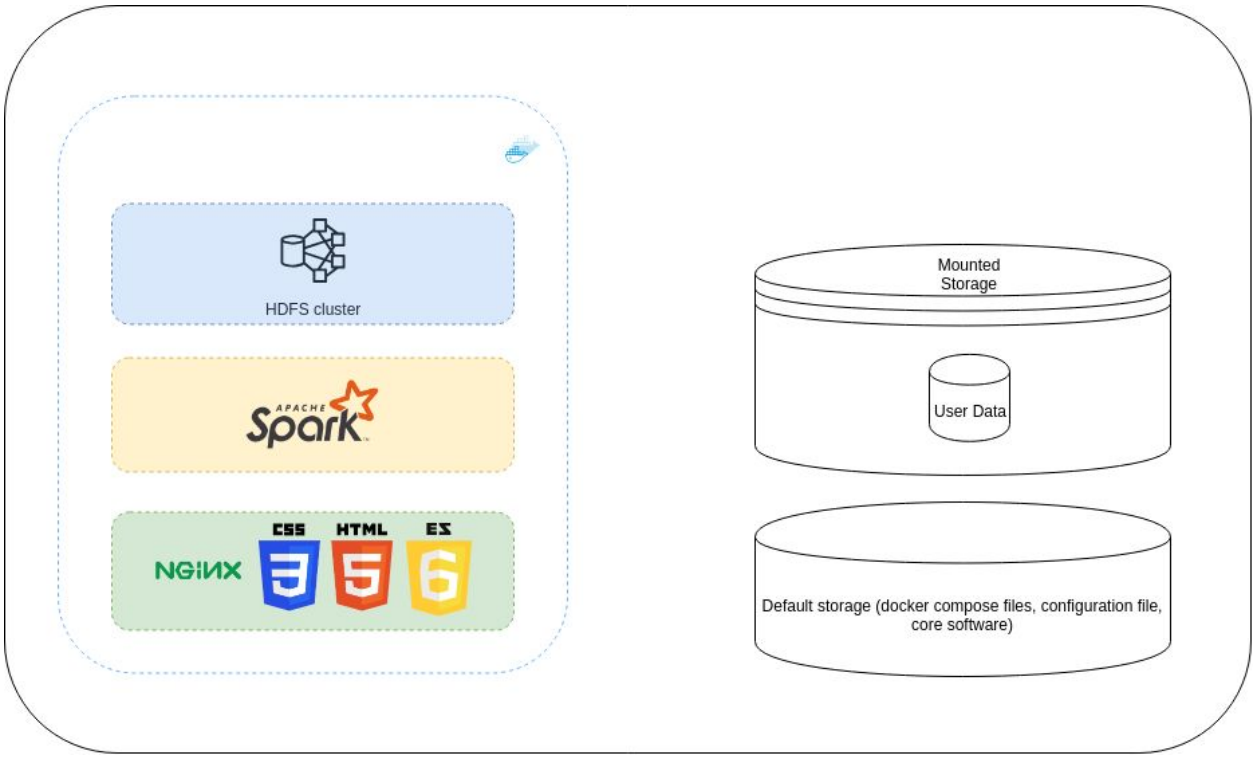


Figure 3: Internal Architecture of the master node

Both master and slave instance has a default 10GB storage mounted at the root directory, which stores small files like deployment scripts and configuration files.

4.3 Docker Swarm VS Separated Containers

The services can either be deployed in swarm mode or just plain docker container. The swarm mode has benefits of name resolution and ability to scaling up/down application dynamically and updating the service without losing availability. By using swarm mode, developers do not have to worry about the IP addresses of physical machines. Instead, they just have to carefully set up

customized networks, service alias, hostnames and use correct hostnames in configurations. Developers are free to select ingress mode, which is featured with a load balancer or host mode, which provides more flexibility on customized Addons, like third-party load balancer or reverse proxy.

Designing a service in swarm mode has some overheads, for example, testing can be more complicated, the developer should provide a pre-built docker image, etc., therefore, it would be better to deploy apps in swarm mode only if they can take the advantage of it. When choosing from swarm mode or traditional container mode, the criterion can either be the necessity of dynamic scaling of the service, or whether the service is stateful or stateless. For instance, it is quite simple to deploy a stateless application with the swarm, such as a static website. Their deployment process is quite predictable, and it does no side effects to the host system to scale up/down. While a stateful service like database or HDFS would be more difficult to deploy using swarm.

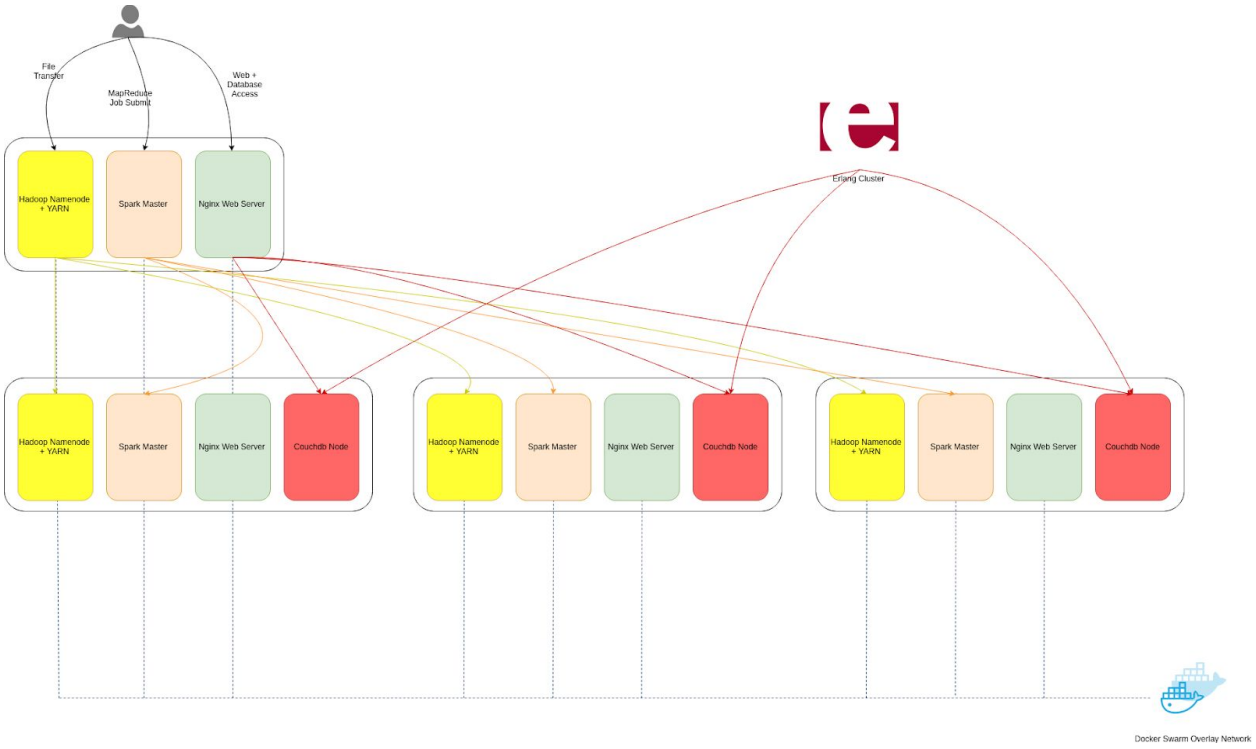


Figure 4: System Architecture

4.4 CouchDB Cluster

The CouchDB cluster is deployed on three slave instances. The CouchDB is more like a peer to peer architecture instead of master-slave architecture, because there is nothing like “master” node in a CouchDB cluster. The official document suggests the number of node in the cluster to be multiple of the number of replicas, so the number of replicas is set to be three, and the number of shards is set to 8, which is a default value.

CouchDB instances are deployed in simple docker containers which communicate over the internet using instance IP addresses. The reason is that the CouchDB communicates over Erlang-Native cluster, which is not functional with `/etc/hosts` files as well as DNS service built in a docker swarm. Besides, CouchDB cluster does not benefit from the ease of scaling up by using docker swarm. The database services are stateful and are quite difficult to dynamically scale. According to the official document, there is a need to moving around data shards before doing modifications like removal of nodes to a cluster.

4.5 Hadoop & Yarn

Hadoop cluster is deployed as Master-Slave architecture, which is necessary. The name node on the master instance monitors the state of the entire system, receiving block reports from three data nodes located on the slave instances. The NameNode is also the entry point for all the Hadoop users, which checks the status of HDFS, Yarn and upload/download files to HDFS.

Hadoop cluster is deployed in swarm mode, so name node and data node can communicate docker overlay network. The advantage is that the configuration can be reused over different physical instances with different IP addresses. All the nodes in the cluster are deployed in host mode, so it does not take advantage of routing mesh and load balancing, and it should not. The placement of nodes is fixed to maintain the consistency of the distributed file system.

4.6 Spark

Spark has the exact same topology as the Hadoop cluster, which puts the master on the master instance and workers on slave instances, one on each. Spark can be deployed as a standalone cluster which uses spark built-in cluster manager or shares a YARN resource manager with Hadoop.

The Yarn Resource Manage provides more advanced features in task scheduling and security, it also brings support in HDFS seamlessly which manages the source code automatically when developers submit their jobs. The standalone mode schedules the tasks in a simpler way, that is, launching executors evenly on all the nodes. Another disadvantage of standalone mode is the lack of support for pySpark cluster mode, which introduces communication overhead between driver and client.

In this project, Spark standalone is used, because there are two facts that make it quite difficult to deploy it with Yarn. The first one is that, in order to integrate HDFS into job submission, it is necessary to add extra name resolution entries in spark nodes, which resolves Hadoop node hostname to its actual IP address. This is because when a client sends remote uploading requests to Hadoop cluster, the cluster firstly returns a resource location, which contains the

hostname of a certain data node. This is simply done by adding 4 extra lines in /etc/hosts file containing the IP address of instances and mapping it into the container. The second fact is that in order to make sure that the resource application mechanism works properly, there must be valid connections to the resource manager at a random port. In the last step, the host is resolved to the public address, where most of the ports in Yarn service are shielded by docker. One way to resolve this is to use subnet IP addresses of Yarn nodes in docker overlay network instead of public IP, however, the subnet addresses cannot be fixed in an overlay network. In this case, the deployment becomes too complicated because 4 spark nodes need to be manually configured. Besides, there is an unresolved issue of the BlockTransferService not being able to find a valid port to bind, which stops drivers/executors from being launched.

4.7 Web Service

The web service is deployed as replicated nodes which utilizes the routing mesh of docker swarm for load balancing and nginx for reverse proxying. The copy deployed on the master instance is almost identical to the ones on slave instances. The only difference is that the nginx server in master instance serves as a reverse proxy for CouchDB nodes except for front end server. Therefore, the web server works like this:

- If a request on port 80 comes in, it is passed through routing mesh and sent to any available container (controlled by Docker swarm load balancer)
- If the request is to master instance address on port 5984, it is always accepted by master instance and is redirected to one of the database nodes using IP hash algorithm. This is because we want to maintain the session for all the accesses from the same IP. (Controlled by Nginx load balancer)
- If the request is to slave instance address on port 5984, it is directly handled by the target server.

4.8 Exception Handling

4.8.1 CouchDB Cluster

Every time a CouchDB node is shut down or exit unexpectedly due to certain reasons like docker upgrading, docker service restart or disk for docker daemon is out of space, it always tries to restart. However, it is not guaranteed that the cluster is reformed and everything is synchronised automatically. When that happens, all the database nodes are still available but data is no longer consistent. However, it is not necessary to explicitly sync the database when consistency is not critical in the application, the database reforms the cluster as long as some

changes are made to them, they can be synchronised by incremental replication mechanism. Therefore, as long as there are live nodes in the cluster, the cluster is available and is easy to be recovered. The conflicts resolution and synchronization tasks are automatically handled by CouchDB using incremental replication mechanism.

4.8.2 Hadoop Cluster

Hadoop handles the exception by setting up a secondary name node, which periodically backups the FSImage and edit logs from the primary name node, which stores the Metadata of the HDFS. If the name node is down by accident, it can be recovered from the checkpoint in FSImage and edit logs in most cases. If one of the data nodes is down, the data in HDFS is still available if the replications of data blocks are correctly placed, and the system can be recovered by restarting the cluster.

4.8.3 Spark

The Spark manager can still schedule the jobs if one or more worker nodes are down, but there is large performance degradation. The Spark cluster in this project does not maintain any states, therefore, it is quite straightforward to restart the cluster without worrying about data loss.

4.8.4 Web Service

The webservice is automatically placed by the docker swarm manager. If the worker node in the swarm is down, the other nodes will have more replicas of service running on them to maintain the total number of live replicas. If the swarm manager is down, the entire web service can be simply restarted because it is stateless.

5 Analysis

5.1 Spark

5.1.1 Spark Overview

We used Apache Spark which is an open source cluster computing framework for big data processing.

Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance. Compared with traditional way doing parallel and distributed processing, we don't need to worry about the challenges such as critical path problem, reliability problem, equal split issue, single split failure issue and aggregation of results. Thus, we can focus on the code logic.

MapReduce is the key feature that enables us to perform distributed and parallelized processing on large data sets. It includes two distinct steps: map and reduces. Resilient Distributed Dataset(RDD) is the fundamental data structure in Spark. Map step simply turns one form of RDD into another form which contains the key features in the dataset. The output of a Mapper is the input to the Reducer. The reducer aggregates the intermediate results from mappers and turns them into a smaller set which is the final result.

5.1.2 MapReduce

In this project, we are using pyspark, which is a powerful python API for spark. There are many functions available for the MapReduce task.

Taking measuring polarity(a measurement for the positiveness of a tweet) for each polygon for example. First of all, all collected twitters within the CouchDB are linked to spark using cloudant. We then cast the data into a RDD, this is the starting point for all Map Reduce functions. Through the map method, we can turn each tweet into a key value pair. The key is the sa2 code for the polygon the tweet lies in and the value is the polarity of the tweet text analysed by the sentimental machine learning model. Next, ReduceByKey is called on the output of the previous stage. This method will aggregate the total polarity according to the polygon code. At last, we will get a polarity number for each polygon, representing the positiveness for the region.

All the other analysis performed on the data such as wrath rate, tweets density follows the same routine.

5.2 Twitter

5.2.1 Harvesting

For tweet harvesting, we have used 3 sources of data (~810k tweets overall):

1. The database provided in the Assignment 2 Specification (~190k tweets)
2. Big twitter from Assignment 1 (~420k tweets)
3. Twitter Stream API and Twitter User Timelines (~200k tweets)

All of the harvesting methods are implemented in python with libraries: TwitterAPI[1] and CouchDB-Python[2], the former one allows us to obtain the tweets, and the latter one allows us to save tweets to the database as documents. For flexibility, the database authentication, Twitter API authentication, and the bounding boxes of Victoria & Australia are all saved in JSON files and read by the harvesting programs.

For the database provided in spec, the first tweet source, we have used RestFul API to obtain the tweets in Melbourne from 01/06/2017 to 31/05/2018. The tweets were saved to the volumes of instances first, and then save to CouchDB after preprocessing. Similarly, for the big twitter json file from Assignment 1, we read the file line by line, and save each tweet to the database after preprocessing.

For Twitter Stream API and Twitter User Timelines, we have written a single multi-threaded program for getting tweets with both of these two APIs at the same time. In the program, we use one thread for harvesting the real-time tweets continuously & adding the users to a user list and another thread for getting the timelines of each user in the user list. Since there is a rate limit of 900 requests per 15-min window, we put 4 Twitter API credentials into a single list, and switch to the next credential whenever it reached the rate limit.

Besides the three methods above, we have also implemented a program for using Twitter Search API, this program allows us to make a query to get 100 tweets in Melbourne, automatically preprocess the tweets, and save them to the database.

To avoid getting duplicated tweets, the document id of each tweet is set to “id_str” of the tweet when we are saving the tweets to the database. Whenever the harvester is trying to save a duplicated tweet to the database, this will raise an exception, this exception could be simply ignored and the duplicated tweet will then be avoided.

5.2.2 Sentiment And Emotion Analysis

In this project, we have used “Textblob[3]” for sentiment analysis, which classifies each tweet as either positive or negative. However, the sentiment analysis may not be enough for us to identify the Wrath tweets, so we have also used “Senpy”[4] API for emotion analysis. For each tweet, “Senpy” can provide the scores of 8 emotions: *Fear, Amusement, Anger, Annoyance, Indifference, Joy, Awe, Sadness,* then, we can classify the tweet to the emotion with the highest score. Here, we treat both anger and annoyance tweets as wrath tweets, and do both sentiment and emotion analysis to each tweet before saving them to the database.

---

[1] TwitterAPI: https://github.com/geduldig/TwitterAPI
[2] CouchDB: https://github.com/djc/couchdb-python
[3] Textblob: https://textblob.readthedocs.io/en/dev/
[4] Senpy: https://github.com/gsi-upm/senpy

5.3 Aurin

In this project, used Aurin datasets are as follow:

- SA2 Estimating Homelessness 2016
- SA2 Estimates of Personal Income - Total Income 2010-2015
- ABS - Data by Region - Population & People (SA2) 2011-2017
- SA2-G52d Industry of Employment by Hours Worked by Sex-Census 2016
- LGA-P40b Labour Force Status by Age by Sex-Census 2016
- SA2-P16a Highest Year of School Completed by Age by Sex-Census 2016

Those datasets cover the range of median income, hours of work per week, education level, population, unemployment rate, etc. of each area. The data from aurin have been processed to get the desired data, for example, we summed up the total number of people work over 40 hours each week, divided by the total number of people, and treated this value as the work overtime rate.

The result of tweet emotion analysis in each area can be analysed against all of those statistics. Once the correlation between tweet results and those statistics is found, the reason for people getting angry may be explored.

The selection of dataset is restricted to the Great Melbourne (gccsa_2016/2GMEL) area and the Statistical Area 2 2016 Boundaries are used as the principle to determine the boundary of each area.

6 Frontend Web App

6.1 Structure and UI Design

The frontend web app in this project is used to visualize the tweet sentiment analyze results to the users. In addition, users can interact with the frontend web to explore some interesting stories. The bootstrap 4 is used to build the UI of the web app (https://getbootstrap.com/docs/4.3/).

Briefly, the web has one HTML page and consists of three sections. The first section is “Home”. At the top of this section, there is a navigation bar. The navigation bar shows all the topic of the story which includes “Home”, “Wrath Story” and “Team”. User can use the navigation bar to quickly scroll the page to the desired section they want. For

example, If the user clicks the “Wrath”, then the page will scroll to the “Wrath” section. Below the navigation bar is the header section of the web app.

The second section is “Wrath Story” which tells a story of wrath. It begins with a text description of the story, then there is a visualization map. User can interact with the map by hovering the mouse over the map, and the map will show the information of each polygon in the map. For instance, if the user hovers the mouse over the polygon of ‘North Melbourne’, the map will show the information at the top corner of the map and the information may be “North Melbourne: tweets num: 100, Wrath tweets num: 10, Average Income: 20000”. Then if the user scrolls the web app, there is a pie chart which shows the total tweets count distribution for different types of tweets. In addition, there is a select button above the pie chart, and the user can select different suburb regions. For example, the user may select the “Southbank” region and in the “Southbank” pie chart, it indicates that “awe_tweet” accounts for the largest proportion. Below the pie chart, there is a word cloud image and a scatter chart.

The last section is “Team”. “Team” section lists the names of all the team members.

6.2 Interactive Maps

To build the map component, Leaflet library is used (https://leafletjs.com). The leaflet is an open-source JavaScript library. It has many mapping features and it is easy to use. The map component in the web app contains two layers. One layer is a base map layer and another layer is a GeoJSON layer. The base map layer is used to initialize the map and some parameters such as center and zoom are set. The center parameter a latitude and longitude coordinate and this coordinate will be on the center of the map. All the base maps used in this web app come from OpenStreetMap (https://www.openstreetmap.org). As in this project, we only focus on Melbourne, hence the center is set to a place in Melbourne. And the zoom range is set between 10-12.

Another layer is a GeoJSON layer, it parses a GeoJSON data and represents it on the map as polygons. On the GeoJSON layer, the user can interact with the map by using the mouse. When the user hovers over a polygon, the map will show some information about this polygon such as polygon name, number of tweets and number of wrath tweets, etc. To achieve this functionality, the GeoJSON data that the layer parsed need to contain all the information. Moreover, each polygon in the GeoJSON layer has different colours. Different colours in polygon indicate the different numbers of the median income.

In addition, there are also some markers with custom icons to indicate the location of some specific tweet. And if the user clicks the marker, a popup will appear and it will show the raw tweet text. For example, in the “Wrath” section, if the user clicks an “angry” icon marker in the map, it will pop up a raw tweet about “Wrath”.

6.3 Charts

To build the chart component, Chart JS library is used (https://www.chartjs.org/). Chart JS has many different charts includes: bar chart, line chart, scatter chart, etc. In the project, we use two charts which are pie chart and scatter chart. The web app gets the JSON data related to the charts from the CouchDB and then processes the JSON data to provide the information such as label names, nums of each tweet, types of tweets to the chart object in Chart JS library. In the pie chart, we compare the number of different type of tweets. Also, the selector button allows the user to switch region. To achieve this switching region functionality, each time we select a new region from the select button, we delete the related canvas element in HTML and then create a new canvas as well as a new pie chart object. The reason for this is Chart JS has an issue when changing the dataset for the same chart and it will need causes problem when hovering over the pie chart. Hence we need to delete and recreate the canvas element. In the scatter chart, we also have a select button which allows the user to select different feature. The scatter chart shows the correlation between the feature and the target.

6.4 CouchDB Connection

The web app connects to the CouchDB directly to get the AURIN dataset and the analyzed tweet dataset. To connect to the CouchDB, we use the jQuery ajax function to send an HTTP ‘GET’ request to the CouchDB. There is a CROS error occurred when the web app requests the data, and the reason is that the web app and the CouchDB have a different IP address. The problem is solved by changing the configuration on the CouchDB and also includes a header section in the ajax code for authorization. In addition, as the loading of GeoJSON costs some time, JQuery async function is used to parse and show the map successfully.

7 Melbourne Research Cloud

The cloud platform used in this project is quite easy to use. It's based on the OpenStack, so there is a bunch of documentation and community support available online. It only provides very functionalities like authentication, computing instances, security groups and mountable volumes, which makes it quite easy for learners to get started.

During development, it was found that the limitation of storage resources may cause some problems as the project proceed. For example, the default storage mounted at / is only 10 Gigabytes, where an operating system, development software and docker are installed. Software like docker is quite storage hungry, which easily takes tens of Gigabytes of memory. When the default volume is eaten up, some services like spark start to have abnormal behaviours, i.e., jobs cannot be executed properly. The only way to resolve this is to clean up the docker directory, uninstall it and restart the services, which is bad in user experience. Even though the docker directory (`/var/lib/docker`) is moved to somewhere else, say, `/mnt/var/lib/docker`, it may still take a lot of space from database and HDFS.

Appendix

Github

https://github.com/ssswww4444/The-Deadly-Sins/tree/master/harvesters

Service Deployment Demo Video

https://youtu.be/nX1Dweh4qHg

Front-End Demo Video

https://youtu.be/astrjribpGE