THE UNIVERSITY OF MELBOURNE

COMP90024 Cluster and Cloud Computing

Group 23

Yalin Chen (1218310)
Qianchu Li (1433672)
Abrar Hayat (1220445)
Jie Yang (1290106)
Yadvika Jayam Nagaraj Yadav (1406716)

Contents

1 Introduction

This project aims to develop a Cloud-based system for analyzing social media data in Australia, leveraging various cloud computing techniques. The system utilizes couchDB for storing and processing Twitter data, and VMs on the Melbourne Research Cloud to enhance the Mastodon Harvester's scalability. To efficiently manage and package the application, deployment techniques such as Ansible and Docker are employed.

Based on the data provided, we explore happiness levels across Victoria by analyzing data from Twitter and Mastodon. The report will provide detailed instructions on how to deploy and use the system. It will explain the system's architecture and design, outlining the components and their interactions. By understanding the system's structure, users will be able to make the most of its capabilities.

Furthermore, the report will elaborate on the data processing methods employed in the system. It will describe how the Twitter and Mastodon data is collected, cleaned, and prepared for analysis. The utilization of NLP emotion and topic models will be highlighted, showcasing how these models are applied to classify the emotions and topics expressed in the social media posts.

Additionally, the report will present various scenarios that are relevant to the analysis of happiness levels in Victoria. These scenarios will demonstrate how the system can be used to gain insights into different aspects of happiness based on social media data. The report will discuss the interpretation of the results and provide recommendations for further analysis or actions based on the findings.

2 User Guide

2.1 System Deployment

In this section, we will outline the process of deploying the entire system from scratch. For clarity, we will first provide detailed instructions for manual deployment using commands and wizard. Additionally, for software deployment, an Ansible automated deployment script will be presented to make the deployment scalable and efficient.

2.1.1 Infrastructure Deployment

In this project, we built our web application on the Melbourne Research Cloud (MRC). The resources assigned to us consists of 8 virtual CPUs and 500GB data volume. In our implementation, we allocated three 2-core instances as our formal servers and two 1-core instances as test servers. Besides, we attached 100GB data volume to each of them. Since the configurations of formal servers are identical, we will briefly walk through the setup process of single instance and the same applies to others.

MRC provides users with handy wizard to manage instances and other resources. It also supports OpenStack API, which enables users to access cloud service in a programmatic manner. In our implementation, we utilize wizard to setup instances.

Before we initialize the instance, we need to create a key pair for establishing SSH connection and configure the security groups for exposing customized ports. To create a key pair, we can simply navigate to `Compute/Key_Pairs` and click "Create Key Pair". After that, we need to download the private key to our local machine. Then we can use the following commands to restrict read and write access of private key to only the owner and super user, and establish the connection to remote server.

```
$ chmod 600 path_to_private_key.pem
$ ssh -i path_to_private_key.pem username@ip_address
```

We also need to expose some ports of the servers to external users. To achieve this, we can navigate to `Network/Security_Groups`, click "Create Security Group" and add the port numbers into customized security groups. For better organization and management, we created a dedicated security group for each port listed in Table 1.

To initilize instances, we need to navigate to `Compute/Instances`. After clicking "Launch Instance" a configuration window will pop up. We can then configure the instance with following parameters:

- **Details**: instance name can be customized, test_01 in this case.
- **Source**: NeCTAR Ubuntu 22.04 LTS (Jammy) amd64.

Port Number	Purpose
22	Allows SSH connection from remote users
80, 443	Allows HTTP and HTTPS queries from remote users.
5984	Allows access to CouchDB Fauxton web administration interface
8888	Allows access to Jupyter notebook from remote users
4396	Enables CouchDB to use Erlang Port Map Deamon to find other nodes
9100-9200	Enables CouchDB nodes to connect with each other
8080	Allows developer to debug back-end API from remote device
3000	Allows access to Node JS Backend Harvester for debugging
3001	Allows access to final web service from remote users

Table 1: Customized Security Groups

- **Flavor**: uom.mse.2c9g.
- **Network**: qh2-uom-internal.
- **Security** Group: include all the security groups in Table 1.
- **Key Pair**: Select the key pair we defined before.

By clicking "Launch Instance", we completed the instance setup. We can repeat the initialization process to get multiple instances.

2.1.2 Software Deployment

The software deployment involves three components: CouchDB cluster setup, harvesters deployment and web server deployment. In our workflow, we have automated all of these components using Ansible playbook.

To setup a CouchDB cluster, we can following the pipeline:

1. Install the CouchDB from the package distribution system, snap in this case.
2. Modify the configurations in `local.ini`:
    - Change bind_address to 0.0.0.0, which bind the cluster interface to all IP address available in the virtual machine.
    - Create the admin and password for login authorization
    - Create the UUID and make it consistent across all nodes in the cluster. They need this UUID to identify the cluster when replicating.
    - Create the httpd secret and make it consistent across all nodes in the cluster. This is to minimize the overhead of cookie requests when nodes are communicating with each other.
3. Modify the configurations in `vm.args`:
    - Replace the node name, which is `couchdb@127.0.0.1` by default, with `couchdb@vm_ip_address`. The name should be a unique identifier in the cluster and the ip address should be valid by which the node can be accessed.

- Define a port range from which the CouchDB node will randomly select a port to communicate with other nodes. In this case, we deploy only one CouchDB node in a virtual machine, so we can specify a certain port for communication. However, to avoid potential conflicts, we specified port range 9100 - 9200.

4. On one node, include all other nodes into the cluster. Notice that there are no master node or servant node in CouchDB cluster. After configuration, all the nodes act equivalently.

We provided Ansible scripts to handle all the configurations mentioned above. We can modify the node IP address in `vars.yaml` file. Then, by running the roles: `setup-install`, `setup-couchdb` and `setup-cluster`, a CouchDB cluster will be created from scratch.

Our Mastodon harvester is a single python script. However, it utilized multiple heavy packages such as torch and transformers for data washing and processing. In order to make our system environment neat and easy to manage, we wrapped the Mastodon harvester into docker container. It is noticeable that we have two different type of harvester. One is used to collect toots from Mastodon server and another is to collect trending words from both Twitter and Mastodon server. The reason we do not encapsulate the later harvester is that it was implemented using JavaScript and share the same dependencies with front-end program. Since it is also a singe file script and do not need to scale, we can put it into our global environment and start it using Ansible scripts.

To deploy Mastodon harvesters as docker container, we should follow the pipeline:

- Create a Dockerfile which derives python official image and install all required packages. Set the entry point to harvester script.
- Send the Dockerfile and script to remote virtual machine, install docker and build the docker image using Dockerfile locally.
- Create a docker container from the image we built before and run it with arguments regarding to the Mastodon server we collect the data from.

We also provided Ansible scripts and Dockerfile to automate this process. Before running Ansible scripts, we need to define the Mastodon server's url, access token, host virtual machine and container name in `vars.yaml` file. Then, by running role `setup-harvest` we can deploy scripts to target vritual machines.

There is no specific pipeline to deploy the front-end and back-end servers since they are organized well. The entry points for them are both of script format and we can run them with single command line. The deployment is also handled by Ansible scripts, which first copy the project from local machine to remote servers and run scripts from the correct directory. The corresponding roles are `setup-backend` and `setup-frontend`.

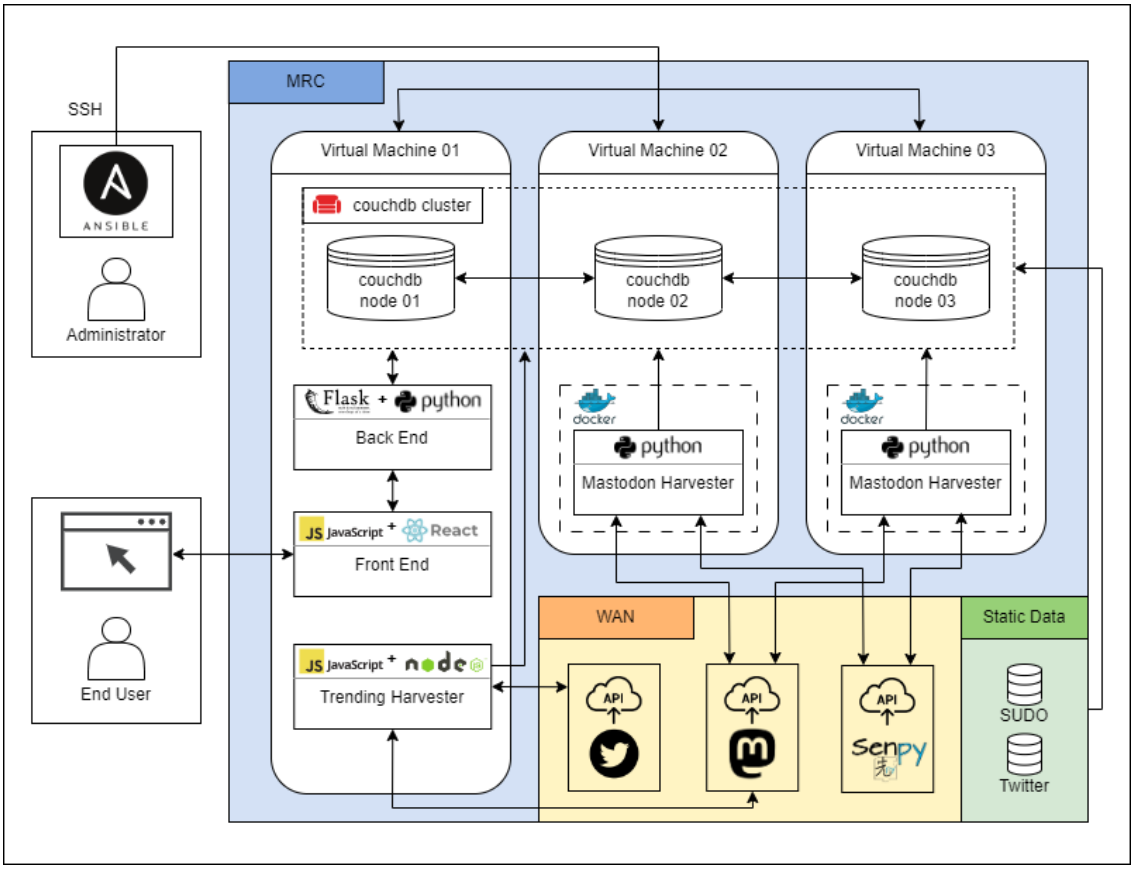


Figure 1: System architecture

3 System Architecture and Design

3.1 Overview

In our implementation, we distributed our system across three virtual machines on MRC. Figure 1 illustrates the overview of our system architecture. We deployed a CouchDB node on each of the virtual machines and connected them to from a cluster. The web servers, including front-end and back-end, are running on the Virtual Machine 01 and open the web service on port 3000. Two Mastodon harvest are running on Virtual Machine 02 and Virtual Machine 03 separately. They collect data from different Mastodon servers, utilize Senpy API to wash data and store the results in the same database. Trending harvester is running on Virtual Machine 01 and share the same environment with front-end server. It collects data from Twitter API and Mastodon server and stores the data in a separate database. Administrator can control the deployment of all these components from remote host using Ansible scripts. Besides, SUDO data and Twitter data are collected and uploaded to the database manually. By adopting this architecture, we can take advantage of distributed system to improve efficiency and failure tolerance of our system. We can also scale our system by including more virtual machines since virtual machines are generally of the same configuration and deployments are handled by Ansible in an automatic manner.

3.2 MRC

Our entire system is built on Melbourne Research Cloud (MRC), which provides us cloud computing resources and user-friendly dashboard. Through setting up multiple virtual machines on the cloud, we can deploy our system in a distributed manner without the requirement of acquiring extra physical devices, which gives us great flexibility in resource assignment. We can experiment on various architecture designs with very low cost and eventually select the best one. Besides, the network across virtual machines is organized by MRC adequately. One can access service of virtual machines via IP address and port number, and VM owner can manage accessibility by configuring security groups. In our development, snapshots provide us with a convenient system recovery. Whenever we make a system error by mistake and cannot undo it, we can utilize the snapshot prepared in advance to restore our system. This saves us a lot of effort in comparison with building up system from scratch. And it also facilitate system to scale up. MRC also provides us with a dashboard to manage our resources. As we discussed in user guide, we can create an instance within few steps using wizard. And several advanced features such as API access are also supported in dashboard. It serves as a good interface between users and MRC.

However, there are still some minor limitations in MRC. It does not seem to support dynamic instance resource monitoring. We cannot get the CPU and memory usage through MRC, though we can implement it by ourselves. Another issue is about its dashboard builtin console. Sometimes we want to do some quick operations on the VMs, and it is troublesome to login VPN and access VMs using SSH. Dashboard builtin console is expected to be an ideal tool for such tasks. Nonetheless, the builtin console is always disconnected and cannot identify the current user.

3.3 CouchDB

Although CouchDB is compulsory in this project, it is indeed well-designed for distributed system. The main feature is that CouchDB can be deployed in cluster mode, which not only improves the memory efficiency but also minimizes the risk of single failure impacting the whole system. Specifically, CouchDB will generate replications for each database, and the replications will be partitioned into multiple shards. The shards will then be distributed to nodes in the cluster and the management system will ensure they will not receive duplicated ones. If the number of replicas is less than the number of nodes, each node will only stores part of the database and saves the memory space as the results. And if the number of replicas is greater than one, the cluster will be able to maintain a good data resilience. In our implementation, we have two replicas for each database and they are divided into six shards. This means each of the CouchDB nodes stores only 2/3 of a replication and the system can tolerant at least one node going offline.

Another crucial feature is **MapReduce**. When we are handling a huge amount of data, CouchDB mapreduce mechanism is utilized here to optimize query time for the backend. One of the significant benefits of the MapReduce mechanism is its ability to quickly process vast amounts of data by **Map and Reduce** those two main phases. During the map phase, it divides input data into smaller chunks and it takes an input pair and produces a set of intermediate key-value pairs. As these tasks are independent of each other, they can be executed in parallel across different nodes in a cluster. Following this, the Reduce phase applies a function to all values associated with the same key to yield the output. Moreover, between the Map and Reduce phases, a 'Shuffle and Sort' step occurs to make all the shuffled key-value pairs are sorted and send them to the same reducers. In this project,we use it to filter positive tweets, count tweets by area, emotion type and topic type and filter the latest harvested Mastodon data, and three main application as following:

- **Positive emotion count View:emotionpos_count** identifies post with positive emotion in Twitter and Mastodon with the sentiment either 'marl:Neutral' or 'marl:Positive' and the emotion either 'wna:joy', 'wna:awe', or 'wna:amusement', if all these conditions are met, view emits the key of post count with the value 1. "ReduceByKey" function is then used to summarise the post count of having positive emotions, and by dividing this count by the total number of posts, the percentage of positive posts weighted in the overall dataset can be calculated.

- **MapReduce View:geo_emotion_count** is created to accelerate the summarizing tweet count process for 5.2.2 map, in the map step, we take full_name, bbox and emotion type of tweet as key and count each tweet as 1 for value, next, "ReduceByKey" is called on the output of the previous stage, this method will count up the tweet count according to unique combination of keys. Therefore, we will get the tweet count for each emotion in each place, then forward this result for later spatial matching with LGA.

- **MapReduce View: top5_topic_positive _post_5d & latest_wordcloud_result** play the quite important role in constructing 5.2.4 wordclouds, especially for Mastodon. The first view is created to select the positive post within top5 topics, moreover, it only keeps the posts created within 5 days of query time. So when the pre-process scheduler can read this view to do word tokenization to avoid the redundant data. After the result stored in the database, we would only need the latest result for front-end request, so the second view is used here. By doing this, the front-end respond quickly and the word cloud can show smoothly.

- **Bar chart View: class_each_emo & different_pos_emotion** summarises the number of different topics within each emotion type and the count of post for each

emotion in Twitter and Mastodon for 5.2.5. By creating two views that identify posts with the three types of positive emotion and positive sentiment. One view emits the key being the classification topics and the value being an object with the emotion type as the key and the value as 1, another view emits the key being emotion type and the value of 1. By applying a the reduce function "ReduceByKey", the counts for each emotion and topic are aggregated. Thus we can return these summaries to efficiently select the top 6 topics classified for each type of emotion displayed in the bar chart.

3.4 Back-end Server

We've established a RESTful back-end server, using Flask Web Framework, to retrieve data from a CouchDB database, structured using map and reduce functions. The back-end server has a RESTFUL architecture which means it exposes all of it's data via JSON responses and hence any other system can integrate with it in a very loosely coupled manner and is completely platform independent. Deploying the server on a Python-based framework was a strategic choice, given Python's strong data analysis capabilities.

There are two kinds of processing tasks performed based on the data type. For the static data like Twitter datawe utilized mapreduce view, and directly fetch the view result from database. For the harvested data, we pre-processed the data and stored the updated results back to database and combining with mapreduce to facilitate the response process.

The back-end server exposes quite a lot of features via it's endpoints. All of the API documentation are provided in 8.3. The */getGeoData* endpoint exposes all the processed data from the SUDO dataset. This endpoint provides finely defined geographical polygon coordinates spread out across various locations in Victoria, Australia. Along with each of these polygons, there are multiple attributes and their values. These values can be visualised on any map to represent whatever the user decides to visualise.

The next endpoint *word_cloud/param* exposes data based on a param which is an attribute name. The data here is classified based on the top 5 topics News & Social Concern, Diaries & Daily Life, Sports, Film Tv & Video & Music and Celebrity & Pop Culture. This data is basically a collection of keywords that are trending now in Twitter and Mastadon. The data is initially harvested from the social media sites and is then classified. The pre-process scheduler tasks are done at midnight to count words frequencies and stored result back to database, which is ready for exposure to the endpoint. This data can be visualised in multiple ways and even used in other data analysis tasks.

The three other endpoints *scatter_plot/param* , *bar_chart_data*, and *chart_data_mastodon*

expose data that is ready for plotting on any scatter chart or any bar chart respectively. The scatter plot exposes data related to Positive Tweet Percentage against attributes which include Median Income, Median Age, Mortgage Stress Percentage, Unemployment Rate and other SUDO attributes. The bar chart data includes data harvested from both twitter and mastadon and exposed data that relates the emotions *Emotion, Awe and Joy* to various sources of Entertainment such as Sports Diaries & Daily Life News, Music, Celebrity & Pop Culture and so on. These plots are very loosely coupled and can easily be visualised on almost any modern visualisation tool.

3.5 Front-end Website

We needed to visual all the data that we were processing. Data is only meaningful if we can visualise them in a meaningful and presentable manner. Hence we choose the most popular front-end framework React JS. React JS has been one of the most preferred front-end frameworks because of how fast it can be up and running and also because of its developer community, for which there are many libraries which are ready to use as *React Components* which saves us the trouble of implementing everything from scratch. React Apps be deployed as a static website as a Single Page Application (SPA) served statically from about any back-end server which is able to serve static content. Hence scaling these kind of web apps is very convenient hence it will requires us to do is to replicate the build across multiple servers serving only static content.

We designed our site to look and feel very modern and also made sure to use the best practices in terms of rendering content. Because we deal with huge volumes of data for visualisation, we only render components when necessary to save save time unnecessary loading time.

We move on to the design and the features of our front-end application. Starting with the landing page, we have kept it a simple and interesting introduction to our analysis. From the intro page, we are taken to the Trending in Twitter page (*/trending* route) where we have a word cloud that we have generated from the harvested twitter and mastodon data. This displays the most trending topics that people are talking about lately and we can filter it to also be the trending topics of the current day. We have also implemented a feature where when click on any of the words on the word cloud, we would be taken to the specific trending topic in Twitter. The word cloud component that we used React Wordcloud, is fully responsive and has very subtle interactive animations which is very nice to look at.

Next, we have the Happiness in Victoria page, which contains a mapbox and scatter plot visualisation. The data that we receive from the *getGeoData* and *scatter_plot/param* endpoints in the backend server is visualised here. The map that we are using is a variant of the package Mapbox GL on which we visualise all the geolocation specific polygons by

displaying the densities of the different attributes via a heat map divided into 7 classes and is colour coded with the legend and the class range values shown on a separate table. Below the map, on the same page we have a few selected attributes to to choose from which plots the Percentage of positive tweets made vs the selected attribute like Mean Income. We used the React Chart 2 JS library for visualising this scatter plot which is also very interactive and is also responsive.

We move on to the next page which is the Topic of interest page. Here we have data displayed on Word Clouds using the same library we used in the same trending page. The word clouds are classified into various different topics based on data harvested from out alternative harvesters both on twitter and mastadon. The sources of data are exposed on two different endpoints *word_cloud/T_topic_name* and *word_cloud/M_topic_name* from twitter and mastadon respectively and hence we render the word cloud component based on the selected data source. On this page, we also have bar charts plotted on categories of entertainment and how they contribute to different happiness metrics which we display using the package Recharts for responsive and visually aesthetic bar charts. We also have the option to select a datasource here from the endpoints *bar_chart_data* and *_chart_data_mastadon* for twitter and mastadon data respectively.

Finally we have the Twitter Stream page where we leverage the use of a twitter stream listening to requested Twitter accounts from a NodeJS backend server. This is the same Node JS server running which collects both twitter and mastadon trending tweets by running a cron job using the node-cron package which runs at a specific time of the day. We listen to accounts once a user has requested to stream the tweets of a specific page once from our website. After that we keep listening to all the tweets and store them in our database. All of our Twitter related streaming and harvesting trends is based on Twitter API V2 and the Mastadon Social Api V1

3.6 Deployment

We utilized Docker and Ansible in our implementation in our system deployment. They both provide us with great convenience in system development and scaling. In our implementation, Docker works solely on Mastodon harvesters while Ansible works on entire system.

Docker is a tool developed to setup a consistent development environment across different devices. It aims to provide users with an environment similar to system image while does not do any modification on current system kernel layer. Thus it is actually a lightweight container isolated from system environment. In our implementation, we wrapped the harvesters with docker container because we need to scale the harvesters and the dependencies of them are complex and heavy (we used AI tools in data processing). To

run a program in docker container, we first need an docker image. We can deem docker container an instance of docker image and different containers run independently. Since our harvester is implemented using Python, we download the python official image. Following that we can build our own image by defining a Dockerfile that adds customized dependencies, copies the scripts to work directory and specifies the entry point. Then by running the container derived from our image, we can scale our harvester easily.

Ansible is another tool that facilitates the system deployment. In addition to Ansible core program, its key components consist of host inventory, playbooks, modules and connection plugins. To control the remote servers from local machine, we need first add their detailed information in host inventory. In our implementation, as shown blow, we need to define group name (in square brackets), host name, IP address and port for SSH connection, system user and private key. Note that SSH port and SSH user can also be defined in `ansible.cfg` file.

```
[server]

server_01 ansible_ssh_host=172.26.130.99 ansible_ssh_port=22
ansible_ssh_user=ubuntu ansible_ssh_private_key_file=test_key.pem

server_02 ansible_ssh_host=172.26.128.217 ansible_ssh_port=22
ansible_ssh_user=ubuntu ansible_ssh_private_key_file=test_key.pem

server_03 ansible_ssh_host=172.26.130.239 ansible_ssh_port=22
ansible_ssh_user=ubuntu ansible_ssh_private_key_file=test_key.pem
```

Since multiple commands are required to setup our system, we need playbooks to manage all these commands. In our implementation, we assigned each sub-task such as cluster setup and front-end setup a individual role. Thus, we can run our playbook regarding specific tasks. Modules are basically atomic functions available to manipulate the remote servers. They act as Linux native commands but actually they were executed as Python scripts. This is also the reason why the remote servers only need Python in Ansible deployment. As to connection plugins, they take responsibility of maintaining the communication between Ansible and remote hosts. The main benefit Ansible bring us is scalablity. When we want to duplicate our system to include more servers, we only need to create instances on MRC, and run ansible scripts from local machine, then we will have a ready-to-use system with CouchDB cluster and all packages installed. Then after data processing, we can also use Ansible to deploy harvesters and web server.

4 Data Processing

4.1 Twitter

We use Twitter huge data from Assignment and select the data with geographical information (full_name) within Victoria, there are around 200k tweets in overall. The involved processing as following:

Emotion Analysis: In our analysis process for Twitter data, we examined the emotions expressed in tweets, particularly the positive ones. To accomplish this, we utilized the `Senpy API`[1] with the 'emotion-depechemood' endpoint, which identifies three types of positive emotions: 'joy,' 'awe,' and 'amusement.' Furthermore, it is worth noting that the emotion 'awe' can be perceived as negative. Hence, we also utilized another endpoint 'Senpy' of the same API to determine the sentiment of tweets: 'positive,' 'neutral,' or 'negative.' By combining both assessments, we extracted tweets that exhibited the three types of positive emotions while ensuring that their sentiment was not 'negative.' This approach allowed us to accurately identify and consider only those tweets that were validly categorized as positive.

Topic Classification: To delve deeper into the underlying reasons behind these identified positive emotion tweets, we employed another 'tweet-topic-21-multi'[2] model. This involved word tokenizing the posts and performs topic classification, which classifies posts into 21 distinct topics. Through this process, we were able to identify 19 unique topics within our positive tweets. This classification enabled us to gain additional insights into various topics that evoke positive emotions among Twitter users.

Spatial Join: Given that the geographic information in tweets often does not correspond precisely with Local Government Area (LGA) names and typically shows smaller locales or towns, we utilize the **spatial join** function from the GeoPandas package to align the bounding box (bbox) of the tweet with the LGA polygon area. Therefore, the majority of tweets are accurately allocated to their respective LGAs, enabling us to summarise the count of positive tweets per area. For the remaining tweets that do not align with an LGA through spatial join, we've constructed an LGA location list sourced from Wikipedia. This assists in matching the full name mentioned in the tweet with the corresponding LGA, ensuring that we appropriately locate as many tweets as possible.

Hashtag Extraction: to construct the topic word cloud, we tap into the hashtag field in Tweets, and count hashtag frequencies and preserve the top 100 hashtags for each topic type from the top 5 types. The result is stored back in the database for future request.

[1] https://github.com/gsi-upm/senpy

[2] https://huggingface.co/cardiffnlp/tweet-topic-21-multi

4.2 Mastodon Haverster

The servers we harvest are:

- https://theblower.au/
- https://mastodon.au/
- https://aus.social/

Harvesting Process: Implementing a method within the listener class to process each incoming post or status update. Content was extracted to find the classified sentiment, emotion, and topic using the same models as for Twitter data. Stores the processed data with the added classification information in the CouchDB database. And run the Mastodon stream using the defined listeners to continuously listen for and process incoming posts.

Scheduled Pre-process: to draw the word cloud like Twitter, since the hashtag in Mastodon is not that informative, we use **work tokenization** and remove stopwords for post context to construct words frequency dictionaries. More importantly, in order to expedite front-end requests and enhance efficiency, we schedule a processing task at midnight, which is facilitated by **Flask-APScheduler** then the result stored in the database for future requests. This approach not only optimizes the request time but also ensures the data is updated and refreshed regularly, providing the most current insights for users.

4.3 SUDO Data

In this project, used SUDO datasets are as following:

- Abs_data_by_region_pop_and_people_lga_2011_2019
- Crime_stats_offences_recorded_offence_type_lga_2010_2019
- Sep_qrt_2021_smhd_lga_unemp_rates
- Family_violence_child_court_afm_rate_lga_jul2013_jun2018
- phidu_education_lga_2016_19
- Phidu_health_workforce_lga_2018
- Phidu_housing_transport_lga_2016_20

Those datasets cover the attributes of median age, median income, population density, unemployment rate, crime offenses count, education level and health workforce, etc. for each Local Government Area (LGA). After we locate the tweets into each LGA as mentioned above, we can explore the correlation between those SUDO attributes with the positive tweet rate.

5 System Functionalities and Scenarios

5.1 System Functionalities

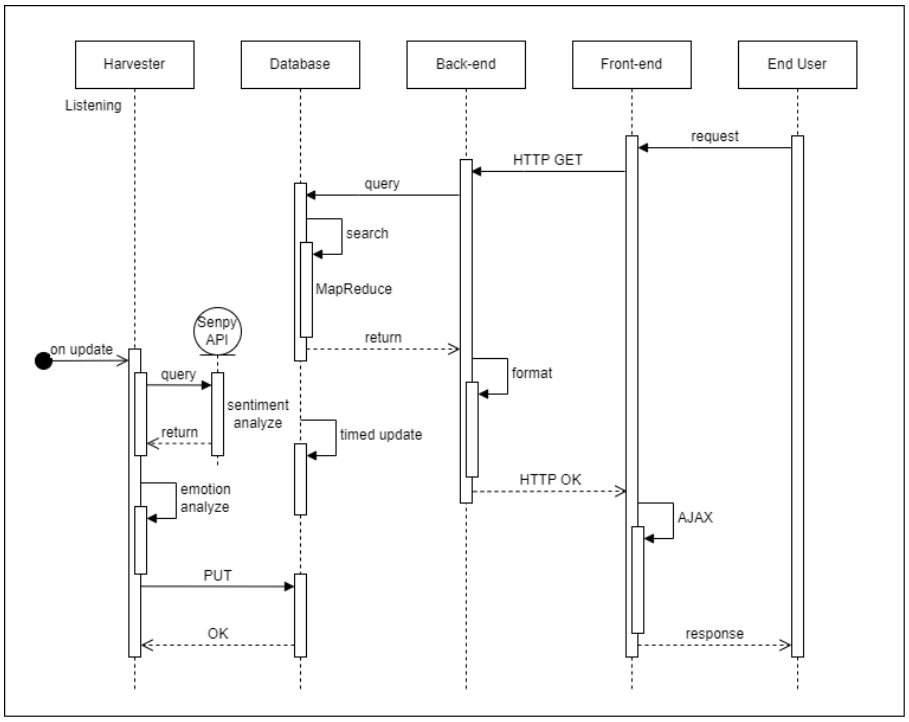


Figure 2: System Functionalities

5.2 Scenario

Happiness, a state of being often characterized by feelings of joy, satisfaction, and fulfillment, can be a somewhat elusive goal with varying definitions from one individual to another. In general, it is associated with positive emotions and overall life satisfaction. Leveraging social media, we aimed to explore the emotional landscape in the state of Victoria during the turbulent years of the COVID-19 outbreak, from 2021 to 2022. Our gauge for happiness was the rate of positive posts, and we define positive tweet consist of awe positive, joy and amusement emotion type. We have used multiple data visualization tools maps, scatter plots, bar charts, and word clouds to gain insights into the emotional landscape and understand how different factors might have influenced happiness levels in Victoria. The whole analysis is broken into 2 main parts:

1. **Happiness across Victoria**, was explored through happiness levels in different between LGA in Victoria and what demographic attributes (SUDO) impact Happiness levels. The overall analysis outcome indicates that approximately 66% of the analyzed posts was determined to be positive tweets. This percentage is considered significant and appropriate, suggesting that a substantial portion of the data was categorized as expressing positive emotions.

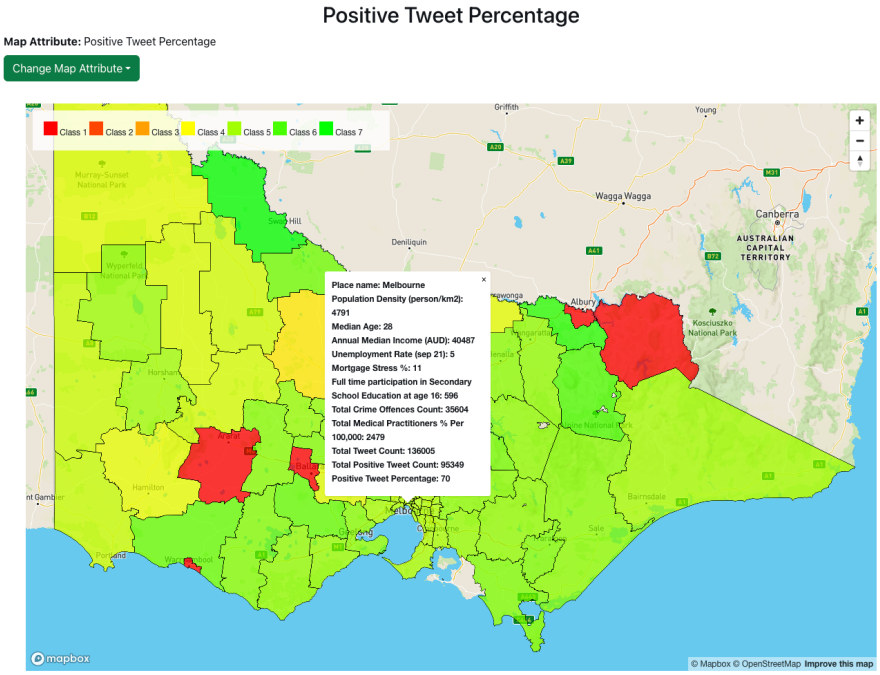
2. **Topic of Interests** through the investigation into what topics people discuss about when expressing positive emotions, and examined whether there were any discernible patterns between the posts in Twitter and Mastodon. When examining the type of positive emotion within each classified topic, we found that all positive emotions were predominantly associated with the "news__social_concern" topic. This topic had the highest percentage, accounting for approximately 31% of the tweets, also approximately 30% of Mastodon posts. These findings suggest that people might feel positive when they encounter positive news, inspiring stories, or social progress. Furthermore, the topic "diaries__daily_life" was identified in approximately 23% of the tweets, also indicating high relevance to positive emotions, due to users often like to share their daily experiences, personal achievements, or moments of gratitude and happiness on social media platforms such as Twitter. Additionally, for Mastodon, the second highest classified topic was "other_hobbies," accounting for approximately 18%, suggesting that users on Mastodon frequently engage in discussions and share positive emotions related to various hobbies or interests rather than their daily life compared to Twitter. All these topic encompasses a broad range of content related to personal lives, routines, and reflections, suggesting its lead to a greater expression of positive emotions.

5.2.1 Trending in Twitter

A live (latest) Twitter trending wordcloud provides immediate insights into what topics are currently popular or noteworthy in Victoria, reflecting the pulse of public sentiment and conversation. When users hover over the word, the word frequencies are shown and on clicking the word, we redirect users to the related tweets on Twitter. This feature can also help users in identifying commonalities between the topics discussed in positive tweets (topic of interests section) and those trending more broadly. This could help to understand the factors contributing to positivity during specific periods and the potential influence of popular topics on public sentiment.

Figure 3: Trending Wordcloud

5.2.2 Map



Map Class Values

Class	Value(s)
1	Values between 0 to 16
2	Values between 16 to 33
3	Values between 33 to 49
4	Values between 49 to 66
5	Values between 66 to 83
6	Values between 83 to 100
7	Greater than 100

Figure 4: Map

We've utilized choropleth maps to illustrate the distribution of various attributes across Victoria. The default attribute is the percentage of positive tweets, calculated by dividing the count of positive tweets by the total tweet count. Additionally, users have the option to select static data for each Local Government Area (LGA), encompassing emotion, sentiment, and SUDO data. These values are divided into seven classes, facilitating the comparison of differences between regions. Furthermore, users can click on each region to delve into the SUDO statistics for more detailed insights.

Our findings revealed that approximately 70% of the tweets in our dataset originated from Melbourne, the largest city in Victoria. The rate of positive tweets from this area was around 70%, placing it in class 5. This isn't the highest, but still a reasonably good score. The surrounding areas displayed a similar positivity rate of about 65%. However, the areas of Whittlesea, Nillumbik, and Wyndham notably had rates below 50%. Each of these areas offers a unique lifestyle with its blend of urban and rural elements. However,

the lower rates of positive posts from these areas could be influenced by a range of factors. These factors could include economic issues, local policy impacts, socio-cultural factors, or even the specific demographics of the people using social media in these regions. Further research would be needed to pinpoint the specific reasons behind these lower rates. Interestingly, we observed that rural areas boasted a slightly higher positivity rate of around 75%, hinting at a potentially higher degree of happiness among the rural population compared to urban dwellers. However, we must consider that this could be influenced by the smaller sample size from these rural regions, possibly skewing the data.

5.2.3 Scatter Plot

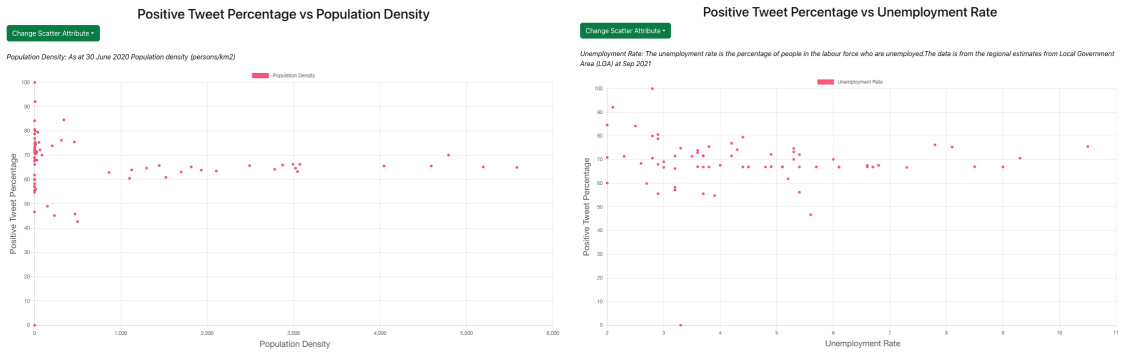


Figure 5: Scatter Plot

In our front-end, scatter plots are used to visualise the relationships between SUDO features and positive tweet rate and user can select the SUDO features to explore.

SUDO Attributes	**PearsonCorrelation** (with positive tweet %)	**P-value (90% confidence level)**
median income	-0.1582	0.1906
median age	0.0613	0.6136
mortgage stress %	0.1086	0.3707
unemployment rate	**-0.1964**	**0.0843**
education level	-0.1428	0.2383
crime offences count	**-0.2002**	**0.0965**
total medical practitioners % per 100,000	0.0479	0.6958
Population density (person/km2)	**-0.2320**	**0.0532**

We can find that the positive tweet percentage only has correlation with population density, unemployment rate and crime offenses count with p-value¡0.1. Even though the correlation is not that strong, we still look deeper into it:

- **Higher population density, lower positive rate** It indicates that in areas with

a higher population density, there tends to be a lower rate of positive posts shared on platforms like Twitter. This could be due to several reasons. High population density often comes with increased noise, traffic, and competition for resources, which can cause stress and reduce happiness. These conditions might negatively affect the overall mood of the community and thus influence the positivity of the content shared on social media. However, we need to be careful if there is a bias from the small sample size and the findings might be skewed potentially.

- **Higher unemployment rate, lower positive rate** It implies that when more people are unemployed, there are fewer positive posts being made on platforms like Twitter. This could be because unemployment is often associated with financial insecurity and stress, which can negatively impact a person's well-being and outlook on life. These negative feelings might then be reflected in the content people share on social media, resulting in a lower rate of positive posts.

- **Higher crime offenses count, lower positive rate** It suggests that when the crime rate is higher, there are fewer positive posts being shared on platforms like Twitter.This could be due to the impact that crime has on a community's sense of security and well-being. High crime rates can create feelings of fear, anxiety, and uncertainty, which might affect the overall positivity of the content shared on social media.

5.2.4 Wordcloud

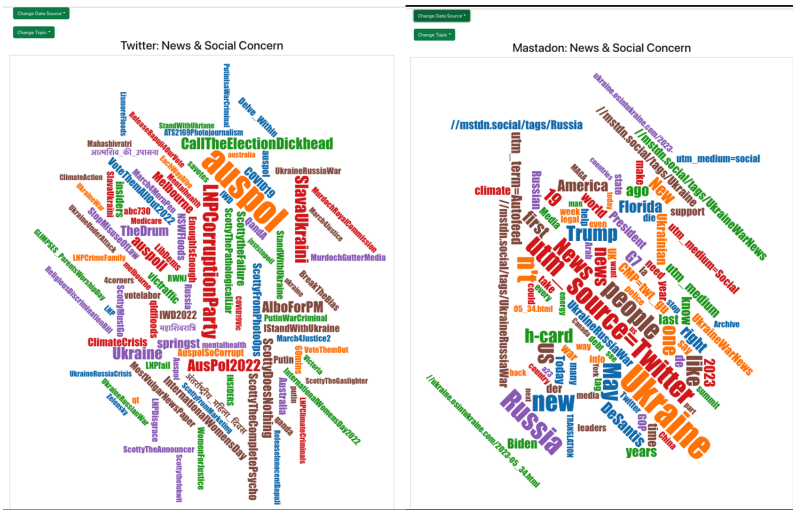


Figure 6: Wordcloud - News & Social Media Topic

Once we've identified the positive tweets, our next step is to uncover the topics that interest people most and what primarily contributes to their happiness. To achieve this, we classified both tweets and Mastodon posts into various topics, generating word clouds based on tag and word frequencies. Users can select the topic type and data source to do some comparisons.

It suggests that the majority of posts revolve around news and social concerns, with topics such as Ukraine, Russia, and climate being prevalent on both Twitter and Mastodon. With tags being available in Twitter data, we were able to extract additional information, like auspol and LNPCorruptionParty. Those political words might be associated with negative connotations, but they were expressed in a humorous manner, leading the model to classify them as positive, it is also a limitation.

When we look into file tv & video & music topics, we also observed disparities in the content across these platforms. For instance, on Twitter, the most frequent words were MAFS and BTS, whereas on Mastodon, YouTube links were most common, accompanied by words related to music such as Metal, Band, and Album. This suggests distinct conversation trends on these two platforms might due to the timeliness.

5.2.5 Bar Chart

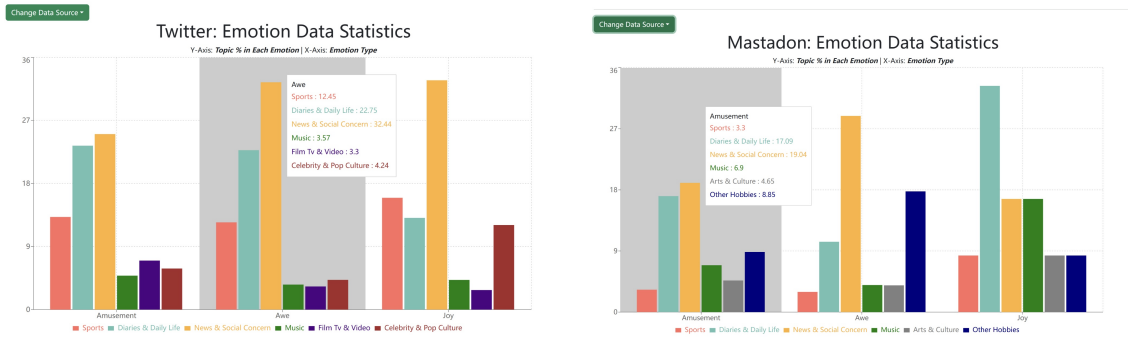


Figure 7: Barchart showing top 6 topics classified for each emotion

The Bar chart evaluates the relationship between different topics and the three various positive emotions, used to find insights into what kind of topic and type of positive emotion people posted about. The Bar chart seen in Figure 7 displays the percentage of the top 6 topics classified within each emotion. Calculated through:
$\frac{\#of\ topic\ classified\ in\ individual\ emotion}{total\ individual\ emotion\ count} \times 100.$
Hovering labels were included in the visualization, clearly indicating the calculated percentage for each individual topic within each type of emotion. The top 6 topics for Twitter data are: 'news &social concern', 'diaries& daily life', 'film tv &video', 'celebrity& pop culture', 'music', and 'sports', And for Mastodon: 'sports', 'diaries & daily life', 'news &social concern', 'arts& culture', 'music', 'other hobbies'. Overall the top classified topic of these positive tweets shows reasonable results, as all topics seems to be aligning with people's happiness revolving around entertainment. Though 'news_&social_concern' distinctly seen as of the word 'concern' in general do not highly associate with positive emotions, hence considerations should be taken in interpreting this topic and suggests this to be a possible concern.

For all emotion, 'news_&social_concern', 'diaries&daily_life' and 'sports', are the top three popular topics in Twitter data. 'celebrity&pop_culture' category contributes a notably high percentage in joy compared to the other two emotions, suggesting that tweets primarily exhibit joy when discussing celebrities, which could be attributed to fans expressing positive emotions and sharing their admiration for celebrities and pop. The joy emotion also has a higher 'sports' related topic percentage over the 'diaries_&_daily' topic contrasting to the other two emotions, suggesting that people feel more absolute joy in sports than feeling the other two emotions, awe and amusement. This finding suggests that people experience more intense joy when it comes to sports, potentially due to factors such as passion, achievement, emotional connection, and positive experiences associated with sports.

In Mastodon similar relationships are seen. For the joy emotion 'diaries&daily_life' seems to be higher in percentage than 'news_&social_concern' different from the other emotions. Suggests that users of the platform are more likely to associate happiness with personal experiences, activities of daily living, and self-expression than to follow news or social issues, which may be influenced by the platform's user statistics, content preferences, and the interactive nature within the Mastodon community. Additionally, it is notable that the 'music' topic shows a particularly higher percentage in the joy emotion. Music is often associated with personal preferences and enjoyment, which can evoke joy and elicit a sense of pleasure and satisfaction in individuals. In contrast to Twitter data, Mastodon posts seem to have a large percentage of topics classified as 'other hobbies'. This can be attributed to Mastodon's decentralized nature of the platform, which attracts niche communities and cultivates diverse interests. This results in a wider range of interests and niche topics discussed on Mastodon, resulting in a greater representation of "other hobbies" in the data.

Overall, the observation that joy consistently exhibits different patterns of behavior of classified topics compared to the other two emotions (awe and amusement). Conceivably because that joy is a more varied and subjective emotion, influenced by personal experiences, personal preferences and cultural factors. The uniqueness of this joy pattern may stem from the different ways in which individuals express and perceive joy, resulting in different associations and topics of discussion.

5.2.6 Twitter Stream

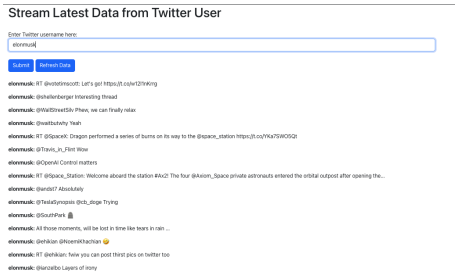


Figure 8: Latest Twitter User Data Stream

This section is designed to allow users to directly receive and browse the latest tweets from their interested Twitter users. It serves as an active bridge for more interaction with the Twitter platform. In addition, it enhances the data source diversity and adds a new facet as a data harvester, creating a more engaging and information-rich user experience. We also store these live tweets on our database so that we can process it later based on a given scenario.

6 Issues, Challenges and Limitations

6.1 Error Handling and Solutions

6.1.1 Architecture

As we discussed in section 3, our system is designed in a distributed manner. One of the main benefits of the distributed system is failure tolerance. Though our database can tolerant at least one random node going offline, there exists a restriction in our whole system: virtual machine 01 cannot fail. The reason is that we deployed our front-end and back-end server on that virtual machine and there is no running copies on other servers. However, if we catch the failure in time, we can manually start our web servers from another virtual machine. Since the program deployments are handled by Ansible, it will take seconds to re-launch our service.

Another issue is about CouchDB cluster. To scale the CouchDB (include more nodes), our current solution is to rebuild the cluster from the start. CouchDB actually supports adding or removing nodes from existing cluster and re-organize shards. However, this process involves too much IO operations and error handling which is a heavy work to automate using Ansible. We would left this work for future improvement.

6.1.2 data / couchdb

In this task, the data volume is big. Therefore we need to ensure our database have enough memory space for storage. We installed CouchDB from Snap, and once we modify the data storage directory, the CouchDB will report an error that no database found and hanging in a uninitialized state. The reason is that Snap restrict the packages under its control strictly. Any interaction with resources outside should be processed via Snap API. For disk access, out of safety concern, Sanp only scan pre-determined directories to search for target resources. Since root directory is not in these hardcode directories, we should be careful to mount the volume in correct directory. In our implementation, we simply mount the volume on the CouchDB current data directory and do not change default directory anymore.

A challenge encountered during our classifying process of positive emotion data, the sentiment methods all proved to be difficult to classify tweets exactly accurate, hence positive tweets identified cannot be purely positive. We handled this limitation by combining the classification results of two **Senpy API** algorithms to increase the possibilities of correctly identifying positive tweets.

6.1.3 backend

In word_cloud endpoint, the ideal result is to get the latest result from the latest_wordcloud_result view. However, when there is no latest record created at the query date, the endpoint will try 10 times to search if there is a result within previous 10 days and return the latest available records. In this case, if there is some issues occur the pre-process scheduler, we can still get the data for presenting wordcloud. A very similar approach is also employed for the harvesters in the node JS server where it recursively gathers results until a certain threshold is met.

6.1.4 frondend

Since most of the website relies on fetching data from various different servers, we have gracefully handling fetch error showing an error message to the user, requesting the user to try again later. A generic fetch error message is logged as follows:

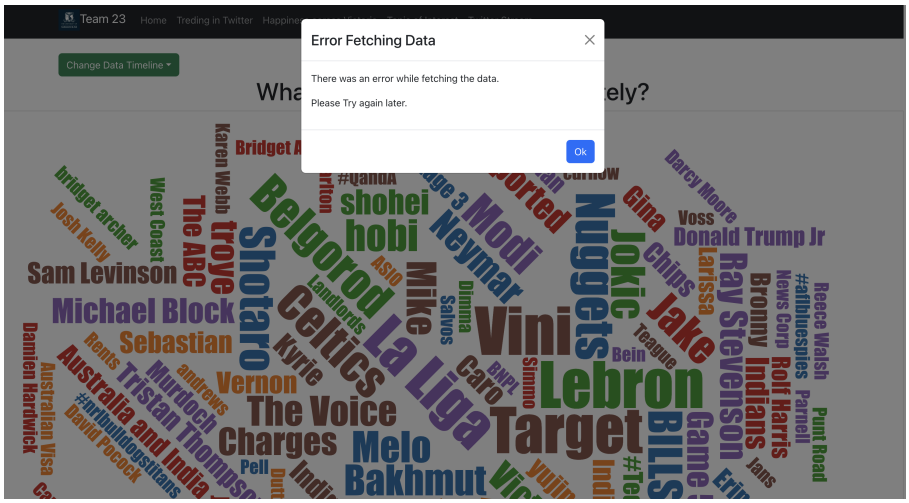


Figure 9: Fetch Errors on front end

6.1.5 docker / ansible

The most errors in Ansible are about deploying couchdb cluster. After we include all nodes into the cluster, we expect to see the membership to be:

```
{
  "all_nodes": [
    "couchdb@server1.com",
    "couchdb@server2.com"
  ],
  "cluster_nodes": [
    "couchdb@server1.com",
    "couchdb@server2.com"
  ]
}
```

where the elements in `all_nodes` and `cluster_nodes` should be identical. However, when using Ansible to automate the deployment, there is always an extra node named "couchdb@127.0.0.1" in `cluster_node`, which makes the clustering fail. After checking, the reason is that in our first implementation, we created admin and password before we changed the name of the node. There is no problem if we do this manually because CouchDB will try to start once installed but eventually failed without a valid admin. Before we finish configuration in `local.ini` CouchDB will reach the timeout. Nonetheless, Ansible script is fast and it can finish configuration in `local.ini` within the timeout duration. Then the CouchDB will start with default name "couchdb@127.0.0.1" and leave it at "cluster_nodes".To solve this problem, we put the configuration of admin and password after all other settings. Then the CouchDB will not start before the node name being changed.

After we add all nodes into cluster and the membership is correct, the cluster setup is still not able to finish. When we run command

```
curl -X POST -H "Content-Type: application/json" \\
http://admin:admin@0.0.0.0:5984/\_cluster\_setup \\
-d '{"action": "finish_cluster"}'
```

it will generate an error

```
{"error":"setup_error","reason":"Cluster setup timed \\
out waiting for nodes to connect"}
```

This is due to CouchDB internal bug. When we query CouchDB to finish cluster, it does not accept username and password as authorization schema. To bypass this error, we queried CouchDB for a session cookie and save it locally. Then we use cookie as authorization schema to finish the clustering and it works well.

Besides, when we use 'ansible.builtin.copy' to copy folders to remote servers, ansible performs extremely slow. It is hangs with neither exception nor error. We did not find out the reason but we can use 'ansible.builtin.synchronize' instead and it works well.

For docker, since it was not heavily used in our task, the only error occurred when we try to pass arguments to scripts but use CMD in our Dockerfile. After we replace CMD with ENTRYPOINT our script can accept arguements and we can delopy harvesters to collect data from customized servers.

6.2 Limitation/Future Direction

From our data analysis, we observed that the tweets classified under the three emotion types were not evenly distributed. This difference in categorization poses challenges for accurately capturing and categorizing tweets that convey specific positive emotions, hence interpretations of emotions made may contain errors or uncertainties. Notably, a very

small percentage (1.2%) of the positive tweets were labeled as 'joy' emotion. However, we may anticipated this outcome since 'joy,' is an extreme positive emotion, that can be likely covered by other positive emotions. Additionally, a significant percentage (75%) of tweets were identified as 'awe' emotion. Considerations should be taken into account that "awe" encompasses a wide range of emotions, and its prominence in the analysis can be attributed to its ability to capture a variety of subtle positive emotional expressions. We acknowledge that this limitation suggests the **Senpy API** model may hold biases towards different emotions, potentially resulting in the loss of valuable insights about diverse positive emotions.

Furthermore, when examining the type classified topic of positive emotion, our results revealed that the highest classified topic for both Twitter and Mastodon data was the "news and social concern" topic as mentioned in data analysing. This finding is somewhat surprising because it suggests that major positive emotions are not directly related to entertainments related topics, which positive emotions usually associate with. This raises concerns that the classification model used may have been trained on a dataset that contains a significant number of tweets related to "news and social concerns", thus exhibiting bias in the results. It is also possible that many users on Twitter prefer to discuss many 'concern' related topics in an ironic humorous manner, leading the **'tweet-topic-21-multi'** model to detect such tweets as having positive emotions, thus analysis could hold many false positive results. As a consequence, valuable insights into the topics that are genuinely associated with or evoke positive emotions again may be inaccurate. Hence, further investigation could be done by applying multiple classification models for topics and emotion classifying, then compare and contrast the performance of these different classification models, aiming to identify the optimal one enable to avoid model bias and increase data interpretations.

7 Teamwork and Contribution

7.1 Tools Used

Communication Channel: we used Discord to communication with each other and share some links and documentation. We hold meeting every week to catch up, the meeting is mainly on ZOOM but in person when we need to wrap up the project.
Work Collaboration: all our work are pushed on GitHub and every updates can keep on track.
Project management tool: we used here is **Trello board**, which can help us understand the task allocation and track each other tasks to make sure everyone is on the same track

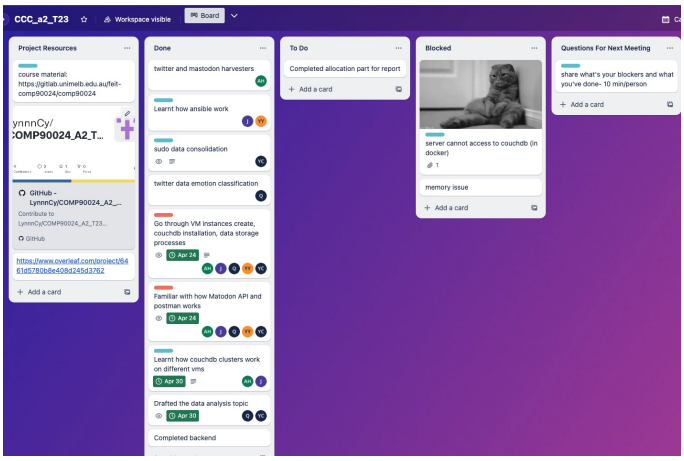


Figure 10: Project management tool

7.2 Contribution Table

Name	**Tasks Allocation**
Yalin CHEN	Data Processing (SUDO Data), Backend (map, scatterplot, wordcloud endpoints) development
Qianchu Li	Data Processing (NLP model application, Mastodon Harvesting), Backend (barchart endpoints) development
Abrar Hayat	Backend (Twitter and Mastadon NodeJS Scheduled Harvesters & Tweet Stream API application) & All Frontend work
Jie Yang	System Architecture Design, Deployment (CouchDB cluster, Ansible, Docker)
Yadvika Jayam Nagaraj Yadav	

8 Appendix

8.1 GitHub repo

https://github.com/LynnnCy/COMP90024_A2_T23/

8.2 Youtube Video Link

`https://www.youtube.com/watch?v=RfKDo405GzM`

8.3 Back-end Endpoints

Endpoint	Description	Method
/tweet_stats	Retrieves overall tweet statistics, including the percentage of positive tweets, total tweet count, and positive tweet count.	GET
/getGeoData	Retrieves geographical data from an analyzed SUDO Data.	GET
/scatter_plot/<param>	Generates a scatter plot showing the relationship between selected attributes and the percentage of positive tweets.	GET
/scatter_plot_note/<param>	Retrieves notes related to the scatter plot for a specific attribute.	GET
/bar_chart_data	Sends bar chart information displaying the positive tweet count and grouping them by emotion type and topic classification.	GET
/chart_data_mastodon	Sends bar chart information displaying the positive Mastodon post count and grouping them by emotion type and topic classification.	GET
/word_cloud/<param>	Retrieves word cloud data for a specified topic from either Twitter or Mastodon.	GET
Available Parameter Values for /scatter_plot/param and /scatter_plot_note/param		
median income	Median income data	
median age	Median age data	
mortgage stress %	Mortgage stress percentage data	
unemployment rate	Unemployment rate data	
education level	Education level data	
total medical practitioners % per 100,000	Percentage of total medical practitioners per 100,000 people	
crime offences count	Count of crime offenses	
population density	Population density data	