THE UNIVERSITY OF MELBOURNE
FACULTY OF COMPUTER SCIENCE & FACULTY OF DATA SCIENCE

THE UNIVERSITY OF
MELBOURNE

Cluster and Cloud Computing

Assignment 2

Group 68 report

Student:	Jiashu Wu	1041990
	Junjie Wang	1045793
	Ruixiang Tang	1298221
	Haodong Gu	1039081
	Changwen Li	1360219

MELBOURNE, MAY 2023

Contents

1 Abstract

This project presents an analysis of social media data to determine factors influencing happiness in Victoria using cluster and cloud computing techniques. These variables include income levels, crime rates, the number of sports facilities, and so on. The study involves deploying various components including a CouchDB cluster, Mastodon Harvester, backend and front-end applications, all encapsulated within Docker containers on top of MRC with automatical deployment using Ansible. The analysis employs rich data sources and a detailed pre-processing stage to clean and prepare data. Finally, the analytical results provide key insights into the correlation between socioeconomic factors and the happiness index of Victorians, it also demonstrates the effectiveness of cloud and cluster computing technologies in large-scale Social Media Analytics.

2 User Guide

2.1 System Deployment

Ansible deploy mrc -> deploy couchdb cluster –> harvester –> backend –> front end –> user invocation

2.1.1 Deploy MRC Instance

First, editing the values in `ansible/build_mrc/host_vars/mrc.yaml` to decide the name of the instance, the image the instance is going to use, the volume the instance is going to attach and the security group is going to add. Then, using the script ansible/build_mrc/run-mrc.sh will deploy this instance with details specific in mrc.yaml in MRC.

2.1.2 Deploy CouchDB Cluster

Couchdb cluster is configured manually. It is done by first install the couchdb in the instance via couchdb-cluster/install.sh (in the virtual machine), during the installation choose the couchdb will be used as clustered database. Three instances with three different couchdb databases are created on three different hosts and they are used to form a cluster with three nodes via the user interface: http://172.26.135.221:5984/_utils#setup. In this node (172.26.135.221), add the other two (172.26.134.190 and 172.26.130.7) as nodes. After these steps, a clustered database is created. Note when creating the instances for couchdb database using the ansible script introduced above, all ports are opened (add in security group rules) to allow communications between different nodes (Detailed steps are shown in the video demonstration).

2.1.3 Deploy Mastodon Harvester

Editing the file in ansible/mastodon/src/mastodon_info.json to specify the mastodon server to harvest data, the access token, the couchdb server used to store processed data and the name of database these data are going to store. Add the host IP address you want to run the mastodon harvester to the ansible/mastodon/hosts file under [COMP90024]. Run ansible/mastodon/run-mastodon_harvester.sh will install all the dependencies including couchdb if they are not installed. After all the dependencies are installed the mastodon harvester will automatically start harvesting, processing and storing the data in the target database (see video demonstration).

2.1.4 Deploy Back-End server

The backend deployment aims to use Ansible and Docker to set up the Flask application automatically on an MRC instance.

Firstly, a Docker file is written to describe the image for the Flask application, which includes setting the Python parent image, adding the local files to the remote working directory in the Docker container, installing any needed packages specified in the `requirement` file, then exposing port 5000 outside this container, finally setting the command to run after the container launches.

Secondly, an Ansible playbook `deployBackEnd.yml` is prepared for recording the tasks for the deployment process: 1. Ensuring a clean state for deployment, the existing Docker container should be removed. 2. Copying the application source code, Dockerfile and package requirement to the remote server. 3. Building a Docker image, running a new container using the image created, and publishing the application on port 5000.

All the above steps can be done by running: `./runBackEnd.sh` in the terminal.

2.1.5 Deploy Front-End application

Docker is used as a container for react app. Under the front-end/ run command: `sudo docker build . -t dockerized-react-app` will build the image based on the Dockerfile and run: `sudo docker run -p 3000:3000 -d dockerized-react-app` will start the react app at port 3000.

We use Nginx as the web server. To configure Nginx, users first need to start editing the default configuration file by typing `vim /etc/nginx/sites-available/default`. Then rewrite the configure file with the following setting in Figure 2.1. Please modify the `ip address` and `port`. Then restart Nginx service by `systemctl restart nginx`.

2.2 End user invocation/usage

Users can access a website to access the analysis results. Users only need to type in the IP address (172.26.134.240) to access the website. There are three web pages on the website, which are the **Home** page, **Map** page and **Mastodon** page. The entries of these three pages are embedded in the navigation bar.

2.2.1 Map Page

On the Map page, users can view the suburb map of Victoria. The colour of a suburb indicates the average sentiment of the tweets posted in that suburb. Users can choose an

```
server {
    listen       80;
    server_name  domain name;

    location / {
        proxy_pass http:// $(ip address) : $(port);
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Figure 2.1: Nginx Setting

influential factor and see the suburbs with high and low values of the influential factor. Users may also see the scatter plot and correlation value by clicking the button **View Analysis**. Users may also view the distributions of different topics discussed in tweets that were sent in the suburb by clicking the button **Suburb Twitter Topics**. An example of how to access the analysis of income and the corresponding suburbs is shown in Figure 2.2.

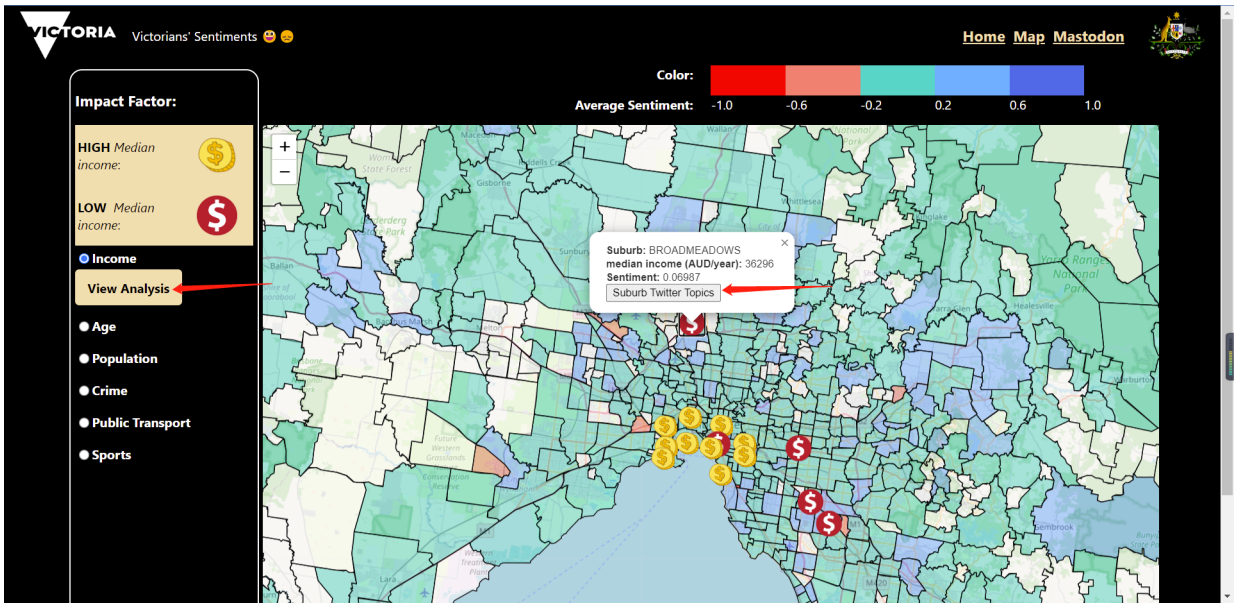


Figure 2.2: Example of Income Analysis in the Map Page

2.2.2 Mastodon Page

The **Mastodon** page presents the data and analysis of Mastodon posts. Users can see the total number of Mastodon posts crawled by the Mastodon harvesters. They can also view the newest Mastodon post crawled by the harvester. There are three types of charts presented on the **Mastodon** page. The first chart is a bar chart showcasing the proportions of different topics in Mastodon and Twitter posts. The second chart is a boxplot chart that demonstrates the sentiment value of different topics. The third chart is a pie chart, showing the proportions of topics within different sentiment intervals. Users need to choose a sentiment interval in the dropdown box.

3 System Architecture

3.1 Design Diagram

This project builds up a complicated system with multiple components, utilizing multiple technologies and frameworks. The architecture is made up of the following components, including Ansible self-deploying module, Mastodon harvester, CouchDB cluster, backend instance and front-end instance. Figure 3.1 demonstrates the overall architecture.

We use Ansible to create and deploy all MRC instances directly. All system configurations of MRC instances are implemented by Ansible, such as security group setting and volume mounting. We also use Ansible to deploy two Mastodon harvesters and CouchDB database.

The Mastodon harvester is a Python crawler that crawls Mastodon post data from Mastodon servers. Within each Mastodon Harvester, two third-party NLP tools are used to process the raw data crawled from the Mastodon servers. We use a pre-trained model from Hugging Face which understands topics discussed in the Mastodon and Twitter post data. NLTK is used to obtain the sentiment value of the Mastodon post data.

We use a cluster to hold the CouchDB databases. There are three instances within the cluster. Two of them act as mirroring replicas, and CouchDB will manage the data replicating automatically. The cluster database architecture ensures availability even when two of the instances fail, which enhances fault tolerance.

We use React to build our front end with Nginx. Using Nginx allows users to access the ip directly without giving access to an additional port, which contributes to better security. We build the backend with Python Flask to provide RESTful APIs.

All components are encapsulated in docker. We choose this architecture to make use of the advantages of each component. In the following sections, we will introduce the

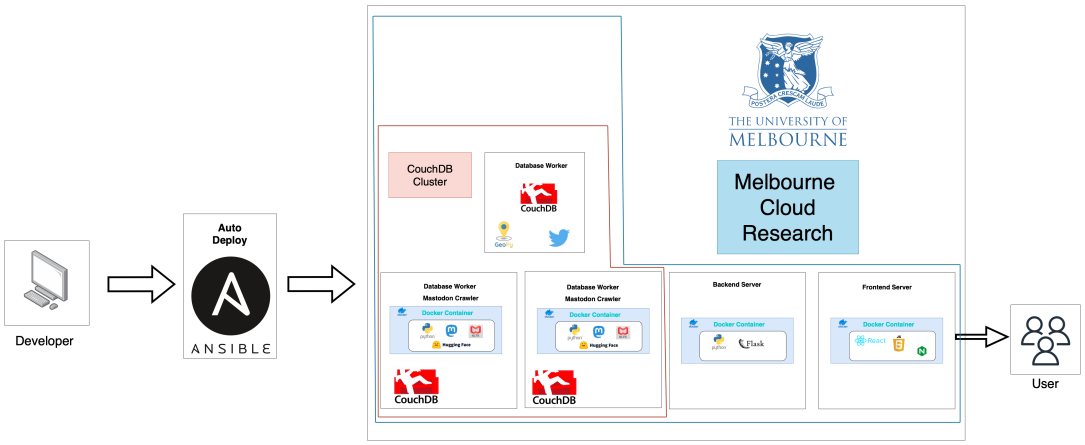


Figure 3.1: Example of Income Analysis in the Map Page

components in more detail, and explain how/why we chose to use them.

3.2 Unimelb Research Cloud

3.2.1 Pros

We are allocated 8 cores and unlimited RAM to complete our project. The Research Cloud offers a scalable infrastructure that can be customized to meet our needs to complete the project which means we are able to allocate these resources on need. For example, we allocate more computation power to the front-end due to high demand for rendering maps and less to the back-end.

The MRC allows for easy scalability, allowing us to quickly and dynamically adjust our computing resources based on demand without the need to invest in and manage a physical machine. We can add a new node easily if the couchdb cluster is not able to handle the current workload.

The MRC supports collaboration among researchers by providing tools for sharing and accessing data, code, and resources. It enables us to work together on the same virtual machine which gets access to the same resources, regardless of our physical location which enhances productivity and knowledge exchange.

The MRC offers access to high-performance computing resources, enabling us to process large datasets, run complex database queries, and perform computationally intensive tasks. This removes the restriction that our personal devices may not be powerful enouch and we can investigate more complex scenarios using the high-performance cloud computing platform.

3.2.2 Cons

The MRC heavily relies on internet connectivity. If the internet connection is slow, unstable, or unavailable, it can affect the accessibility and performance of cloud services. Offline access to applications and data without connecting to the university of Melbourne network is impossible.

The MRC is accessed through the internet which means if anyone knows the IP address of the machine, they can try to access this machine and if they can compromise the authentication process they will be able to access all the content in this machine. For example, our database may be compromised if we are using simple usernames and passwords since everyone in the university of Melbourne network can know there is a database on a specific IP address.

We are only able to use the image the MRC provided which means that we may not be able to use the latest systems or the latest version of certain software if MRC does not provide. In addition, if the MRC is down for any reason, we have no control and the only thing we could do is contact the administrator and hope it will be resolved as soon as possible.

3.2.3 Image creation and deployment

NeCTAR Ubuntu 22.04 LTS (Jammy) amd64 is chosen and used as the image of all our instances. All the virtual machines are deployed by the Ansible script. As a result, if any of the instances is not working, it can be quickly resolved by re-deploying a new instance using the ansible script.

3.3 Cluster CouchDB Component

3.3.1 Map Reduce

For working with large data sets, Twitter data in this case, MapReduce should be used to create pre-calculated views, which can significantly improve query performance and response time. With MapReduce, the results of complex calculations on large amounts of data can be stored as views, which can be accessed directly when querying. Avoiding the calculations in real-time, and getting the data directly from the pre-calculated and stored results, which greatly reduces response time. However, maintaining MapReduce views will cause storage expenses, and they will update whenever the original data changes.

Some non-trivial MapReduce function is deployed. For example, to calculate the proportion of Twitter topics within each sentiment range, the Map function puts the sentiment

of each doc into four ranges and emits one, and then Reduce function is used to sum it up for each interval. In addition, as a large amount of Twitter data needs to be processed by MapReduce, CouchDB splits the data into multiple parts and executes MapReduce on each part independently. So, `rereduce` function is used for calculating the sum of the Sentiment score for each Suburb.

3.3.2 Fault-tolerance

Through the process of data replication among the cluster's nodes, clustered CouchDB instances provide solid fault tolerance. This replication strategy makes sure that data is mirrored across several instances, providing redundancy and protecting against potential failures or outages. When a CouchDB cluster is set up, each node actively replicates its data to other nodes. Each node acts as both a source and a target for replication because it is a bidirectional process. As a result, any modifications or updates made to a document in one node automatically spread to the other nodes in the cluster.

3.4 Mastodon Harvester Component

To obtain the most recent data from the Mastodon server, the use of development keys is a requirement for this task. By using the Mastodon StreamListener in Maston API with the development key, it becomes possible to retrieve the latest toot data from the Mastodon server. However, not all the information obtained from the crawled toot data is relevant to our analysis. Therefore, additional processing of the retrieved toots is still necessary. The specific steps involved in this processing are outlined in Section 4.2.7.

3.5 Back-End Component

The backend server is implemented using the Flask web framework and serves as an API for the CouchDB database then provides to the frontend for visualization. It contains above 20 routes for supporting several scenarios, each first performing a JavaScript MapReduce function to process the original data of the database and then creating views to speed up subsequent query operations. As the major operation of the backend in this Assignment is to deal with the GET request, the lightweight and flexible web framework Flask is used, which is also well-supported for developing RESTful APIs[4].

3.5.1 ReSTful design

Representational State Transfer web services is a way of providing interoperability between computer systems on the Internet[2], which relies on the HTTP communication protocol and the functions and data can be accessed by URL. A unique URL identifies a kind of information that can be accessed, such as a database record, an application object and so on. REST represents the URL in the form of `/user/name`, then followed by the HTTP methods, such as GET, POST, and so on. In this assignment, the endpoints provide based on different scenarios, such as `/topics_proportion_tweets`. RESTful API benefits from stateless, which can handle each request independently and make the design simpler. Also, it can improve the responding speed and reduce the server load by utilizing caching. In addition, REST supports any coding language that can send HTTP requests. However, RESTful also has some disadvantages, such as high load on the network[3], and requires the client to poll the server for an update in real-time scenarios. In the case of showing real-time Mastodon messages, RESTful can lead to delays, and frequent requests will overload the server. Some technology like WebSockets can provide better real-time performance than REST APIs.

3.5.2 Fault-tolerant

In the Flask application, Cross-Origin Resource Sharing is used which allows Web applications to access resources in other domains. In addition, the server stores the connections to three CouchDB instances as shown in Figure 3.2, it will try to get the target database from one of these instances, and if it fails on one instance, it will try the next. This design provides load balancing and fault tolerance capabilities that improve system performance and fault tolerance.

3.6 Front-End Component

3.6.1 React

React is a prevalent front-end framework that is utilized in this project as well. Compared to traditional JavaScript, React provides better web page rendering performance with virtual DOM (Document Object Model) and more declarative programming syntax. Incorporating with SCSS (a higher-level version of CSS), React can build web pages with beautiful layouts, high efficiency and relatively short development time.

To draw the suburb map of Victoria, `react-leaflet` is used. To display the boundaries of Victoria even when the map is zoomed out, we overlaid the boundaries of each suburb

```
hosts = ["172.26.135.221", "172.26.130.7", "172.26.134.190"]
# create a list of couchdb connections
couches = [couchdb.Server(f"http://{admin}:{password}@{host}:5984/") for host in hosts]
print("Connected to couchdb.")

± rex *
def get_db(name):
    for couch in couches:
        try:
            return couch[name]
        except Exception:
            continue  # if exception occurs, try next couch connection
    raise Exception("No valid couchdb connection.")
sport_facility_suburb_db = get_db('sports_facility_suburb')
public_transport_db = get_db('transport')
employment_db = get_db('employment')
income_db = get_db('income')
```

Figure 3.2: Connecting to CouchDB cluster

as a polygon. The shape of the suburb is obtained from the Victorian government website.

The scatter chart, bar charts and pie charts used for data analysis are drawn with `react-chart`, while the boxplot chart is drawn with `react-plotly`.

The data used in the web pages, including data for each chart, is obtained from the Flask backend. The HTTP request is sent asynchronously so that users can still interact with the website while waiting for the backend server's response. On the Mastodon page, the website sends a request to the backend server asking for the current total number of crawled Mastodon posts and the newest crawled Mastodon post. An effect hook is used to achieve this by setting a dependency of time variable that refreshes every second.

3.6.2 Nginx

Nginx is used as a web server. Nginx allows access from clients to the port of a website. There are potentially two advantages to Nginx. The first one is that it can map a domain name to an IP address. The second advantage is that it provides better security without granting access for clients to access the port of the front-end program.

3.6.3 Fault-tolerance

The front-end component improves security by ensuring robust programming. The error could occur if the format of the backend data is not the same as the program decided. For instance, the website is crashed if there is a program using the data of tweet topics while the raw data do not have a key to store topics. Thus, we use some try-catch

and if clauses to deal with incorrect data.

3.7 Containerization

According to the lecture, though virtualization brings lots of benefits, such as application isolation and horizontal scalability, it causes some resource costs. The operating system and binaries in each virtual machine can lead to duplication across VMs, which consumes server processor power, memory, and disk space. So, these drawbacks will restrict the number of VMs a server can support. On the other hand, Containerization allows multiple virtual instances to share a single host operating system, along with its associated drivers, binaries, and libraries, which minimizes resource waste.

3.7.1 Docker

In this project, the system consists of different operations performed by smaller individual deployed services (crawler, backend, frontend). Hence, a low-overhead virtualization technique is needed here, and one of the lightweight virtualization technologies [5], Docker is deployed on top of the MRC instance. It benefits our system in many different ways:

Firstly, as the application runs in an environment isolated from the local OS, there will be no conflict between different environments and no interaction with each other. So, all team members can debug in a consistent environment.

Secondly, Docker supports quick deployment and scaling after writing the Docker file and requirement files. In addition, after writing the Ansible playbook, it can further make deploying an application faster and more efficient. This is important for Web applications that need to scale quickly to cope with changes in load.

Next, two mastodon harvesters are deployed in this project, and they are benefitted from the portability and scaling of the Docker. Because Docker containers can run on any system that supports Docker without any modifications, it makes container creation and configuration simpler and more repeatable.

For future improvement, some Container Orchestration Tools can be used to manage the availability and scaling of containers, such as Kubernetes and Docker SWARM. These can also improve development efficiency, reduce manual operations, and ensure consistency from every deployment.

3.7.2 Fault-tolerance

Although the docker provides an isolated environment for the application from OS, the Docker container is still running on a host instance, and the Docker container running

in it will stop running if the instance shutdown. However, one of the advantages of using Docker is that it easily packages an application and its dependencies together, so that if one of the instances shut down, it can be recovered quickly by starting a new container to replace it, greatly reducing the time to restore service.

4 System functionalities

4.1 Data Sources

The data used in this research were obtained from *SUDO*[1], *AURIN*[2], *Data VIC*, *Mastodon* and a *huge Twitter file*. In total, six data files were used for this assignment: the Victorian Income File, the Victorian Employment File, the Victorian Crime File, the Victorian Age & Population File, the Victorian SA2 Suburb Shape File, and the Huge Twitter File.

4.2 Data Pre-processing

For this project, data pre-processing was necessary to ensure that the data were correct, consistent, and analytically fit. The decision on data preprocessing was made in accordance with the subject matter we selected, which is what elements affect people's sentiments in Victoria. The detailed data preprocessing of each file will be discussed below.

4.2.1 Twitter Data

The analysis stage of the Twitter data entails utilizing several key features extracted from the extensive Twitter dataset. These features include the sentiment score, bounding box (bbox), language, and content. However, the mere acquisition of the raw data is inadequate due to the necessity of incorporating suburb names, considering that the frontend employs SA2 boundaries for map division. To fulfil this requirement, the Victorian SA2 Suburb Shape file from Data VIC is employed. Figure 4.1 provides a visual representation of the SA2 suburb shape file, which contains both the suburb name and its corresponding geometric polygon. To determine the specific suburb associated with each tweet's origin, a multi-step procedure is employed. Initially, the center point coordinate of the bounding box is determined. Subsequently, the imported **Point** function from the **shapely.geometry** library is utilized to convert the coordinate type to a **point**

[1] https://sudo.eresearch.unimelb.edu.au
[2] https://aurin.org.au/

	LC_PLY_PID	LOC_PID	DT_CREATE	LOC_NAME	LOC_CLASS	STATE	geometry
0	lcp6229215ba53f	locb0dcb52a6b55	2021-06-24	Abbeyard	Gazetted Locality	VIC	POLYGON ((146.81721 -37.09735, 146.81729 -37.0...
1	lcp386f2bcf9bce	locb9872f35df41	2021-06-24	Abbotsford	Gazetted Locality	VIC	POLYGON ((145.00235 -37.80723, 145.00350 -37.8...
2	lcp122c942a8fc9	loc8123ed12ea8d	2021-06-24	Aberfeldie	Gazetted Locality	VIC	POLYGON ((144.89830 -37.76465, 144.89790 -37.7...
3	lcp9f50bd795d16	loc1eb4a229104a	2021-06-24	Aberfeldy	Gazetted Locality	VIC	POLYGON ((146.39447 -37.71008, 146.39405 -37.7...
4	lcp59bf69caacb0	locb17be87767e1	2021-06-24	Acheron	Gazetted Locality	VIC	POLYGON ((145.75030 -37.24313, 145.75036 -37.2...

Figure 4.1: SA2 geopanda

type. A **for loop** is then implemented to iterate through the dataframe, employing the **.within** method to assess whether the centre point resides within the current polygon. Upon locating the polygon in which the centre point resides, the loop is terminated, and the suburb is appended to a dictionary containing the aforementioned features, thereby forming a comprehensive dictionary.

Understanding the content of tweets and being able to determine their topics is of great importance to our project. We are specifically interested in investigating various factors that may influence the happiness of individuals in Victoria. Therefore, having knowledge about the topics covered in tweets would be highly beneficial for our analysis. In order to determine the topic of each tweet, we utilize a pre-trained NLP model named `cardiffnlp/tweet-topic-21-multi` [1] from Hugging Face[3]. This model uses a TimeLMs language model that was trained on over 124 million tweets from January 2018 to December 2021. It is a multi-classification model that can classify a particular tweet into numerous topics. Because this model was created exclusively for the English language, we only took into account English tweets when analysing the data for this project. Below is a list of every potential tweet topic for `cardiffnlp/tweet-topic-21-multi`:

- arts & culture
- business & entrepreneurs
- celebrity & pop culture
- diaries & daily life
- family
- fashion & style
- film tv & video
- fitness & health
- food & dining
- gaming
- learning & educational
- music
- news & social concern
- other hobbies
- relationships
- science & technology
- sports
- travel & adventure
- youth & student life

This process is repeated for each tweet, allowing for the completion of the dictionary before moving on to the subsequent tweet. It is important to note that we utilised mpi4py to speed up the overall processing speed and that readline was the sole tool we used to

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

collect valid data from each tweet using the same shape, which significantly increased running speed. Upon the completion of processing a Twitter dataset, a prompt course of action is to expeditiously store the data in CouchDB, thereby mitigating the consumption of system RAM. This ensures that the processed data is promptly persisted in a reliable and efficient manner, without imposing unnecessary strain on the system's volatile memory resources. In the end, there are only 640,786 tweets that met our requirements.

4.2.2 Income Data

We extract three columns which are median income, mean income and SA2 suburb name from the data file called `ABS_-_Personal_Income_-_Total_Income__SA2__2011-2018.csv`. As all the data in this file are within Victoria, there is no need to filter the data. This data file is sourced from SUDO.

4.2.3 Population Data

We abstract two columns from the data file called `ABS__Data_by_Region__Population_ __People__SA2__2011-2019.csv`. The two columns are SA2 suburb and the median age corresponding to that area. This data file is sourced from SUDO.

4.2.4 Crime Data

we obtained 7 columns from the data file called `csa_crime_stats_offences_recorded _offence_type_lga_2010_2019-548116567796959816.csv`. Among these 7 columns, 6 columns represent the offence count for different level of crime. We sum them up and use the left column called lga2 which indicate the suburb name as key. This data file is sourced from SUDO.

4.2.5 Public Transport Data

All of the columns that were initially contained in the dataset are still present in the data file that was used for the analysis, `Melbourne_Transport.json`. However, because the suburb name property is missing from the original file, geopy must be used to geocode the locations and determine the names of the relevant suburbs. It's crucial to remember that the `Melbourne_Transport.json` dataset was obtained from SUDO, guaranteeing its dependability and trustworthiness for this study's needs.

4.2.6 Sports Data

We sum the number of facilities according to the suburb name provided in the data file called `Victoria_Sport_and_Recreation_Facility_Locations_2015-2016.json` and make a dictionary using suburb name as key and number of facilities as value. The data file is sourced from Sudo.

4.2.7 Mastodon Data

In our data analysis, we mainly use toots collected from two Mastodon servers, namely `Mastodon Australia`[4] and `Mastodon.world`[5]. In order to harvest data from Mastodon servers, we are first required to create accounts on these servers and then get the API access tokens for toot harvesting. Similar to Twitter data, we will only analyze English toots on Mastodon servers.

We select five features for each harvested toot, which are `id`, `time_created`, `content`, `url` and `language`. There are a few important aspects to consider in this procedure. Firstly, the initial content of the toots that were collected is in HTML format. To make it more easily readable, we utilize the html2text library to convert the original content into a text format. Secondly, even though we selected the Australian mastodon server, we encountered some toots that were not in the Australian time zone. To address this, we employed the Pytz library to convert all the timestamps to the Australian time zone.

Additionally, In order to be able to compare and analyze the data more effectively with the Twitter data, we also implemented sentiment analysis and topic classification for each crawled toot. Here, the same topic classification approach mentioned in Section 4.2.1 is applied here. For achieving the sentiment score of each toot, we utilize `SentimentIntensityAnalyzer` in NLTK library to get the polarity score of each toot, there are 4 outputs in the polarity score, the one we used for representing the toot's sentiment score is the compound score, which is a normalization score and has ranges from -1 and 1.

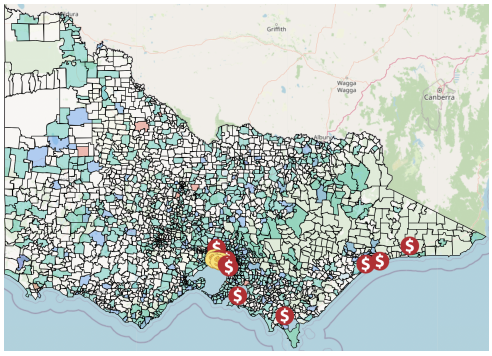
After all these processing, we send the toot data to our CouchDB server. Unfortunately, the amount of data in Mastodon is significantly smaller when compared to the vast number of tweets, and although we have used 2 harvester crawlers and keep running in the background, we have not been able to achieve a number of toots close to the number of tweets in the end (almost 150,000 by the project deadline).

[4] https://mastodon.au/
[5] https://mastodon.world/

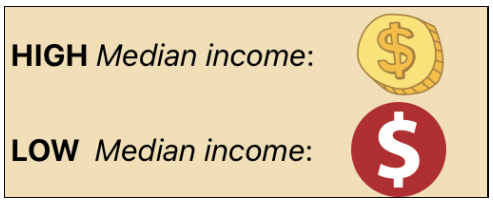
4.3 Analytical result

4.3.1 Scenario Analytical Result

The depicted Figure 4.2 illustrates a discernible trend, wherein the proximity to the central region of Victoria correlates positively with higher income levels among individuals. Conversely, remote areas exhibit comparatively lower income levels. Victoria's center correlates with higher incomes due to better access to economic opportunities and developed infrastructure. Remote areas, on the other hand, face limitations in proximity to economic hubs, resulting in lower income levels. Individual circumstances and industry specialization also contribute to income variations within regions.

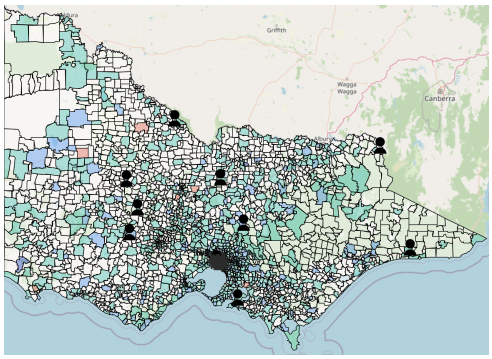


(a) Income map



(b) Logo Meaning

Figure 4.2: Victoria average income per suburb



(a) Population Map

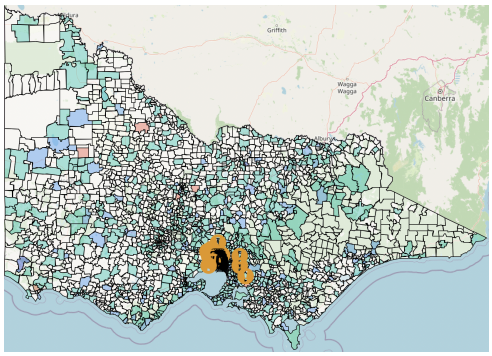
(b) Logo Meaning

Figure 4.3: Victoria population per suburb

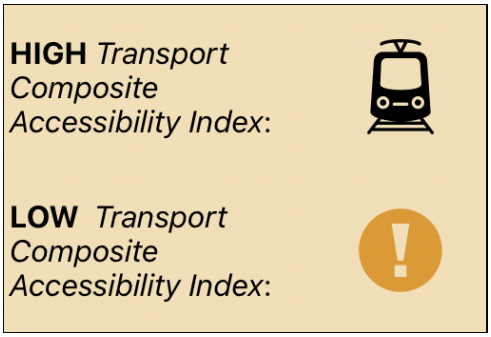
As can be seen from Figure 4.3, the closer to the center of Victoria, the district seems to have a larger population, and vice versa. it may because urban areas near the center often offer more amenities, job opportunities, transportation options, and cultural attractions.

These factors make these areas more attractive to people, resulting in higher population densities. Conversely, districts farther away from the center may have a more rural or suburban character, which leads to lower population densities.

It can be seen from the Figure 4.4, the closer to the center of Victoria, the more developed the public transport, and vice versa. This suggests that urban areas near the center tend to have better public transportation infrastructure due to higher population densities and the need for efficient mobility options. In contrast, districts farther from the center may have less developed public transport systems due to lower population densities and potentially less demand for extensive transportation networks.

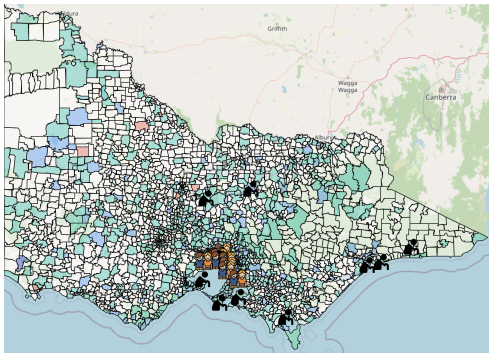


(a) Transport map

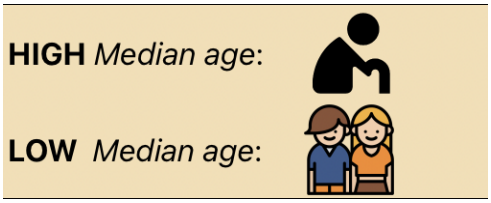


(b) Logo Meaning

Figure 4.4: Victoria transport composite accessibility index per suburb



(a) Age Map

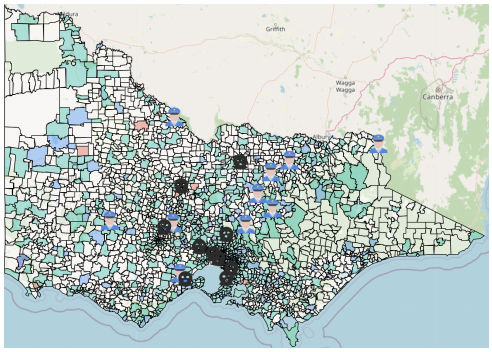


(b) Logo Meaning

Figure 4.5: Victoria average age per suburb

Figure 4.5 indicates that there is a concentration of young people in the center of Victoria, while the average age of people living on the outskirts of Victoria tends to be relatively higher. This observation suggests that the center of Victoria attracts a younger population, possibly due to factors such as employment opportunities, educational insti-

tutions, cultural attractions, and a vibrant social scene. On the other hand, the outskirts of Victoria may have a higher proportion of older residents, possibly due to factors such as residential preferences, availability of housing, and a quieter or more suburban lifestyle.



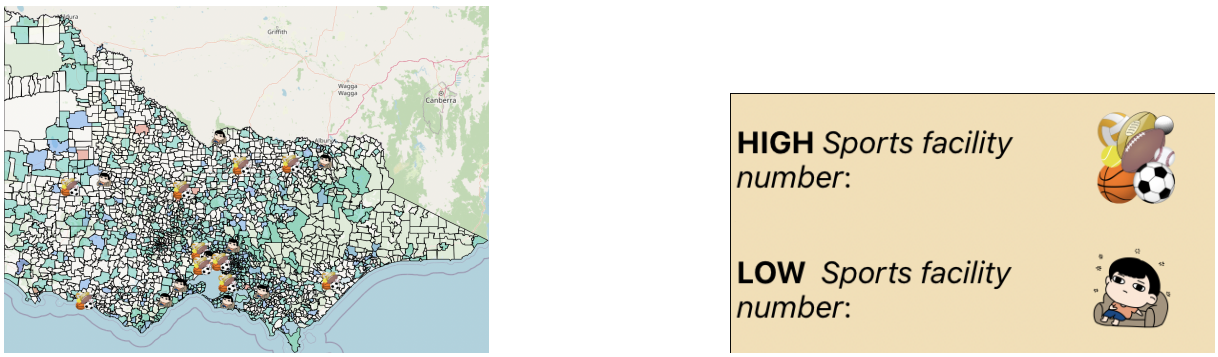
(a) Crime Count Map

(b) Logo Meaning

Figure 4.6: Victoria crime count per suburb

Figure 4.6 illustrates that the crime rate in the center of Victoria is higher compared to the fringes. This observation suggests that there is a higher incidence of criminal activities occurring in the central areas of Victoria. The reasons for this could include factors such as higher population density, greater economic disparity, higher levels of social unrest, or the presence of areas with higher concentrations of businesses and public spaces. Conversely, the fringes of Victoria, with lower population densities and potentially more residential areas, may experience relatively lower crime rates.

Figure 4.7 indicates that there are sufficient sports venues in most parts of Victoria. This suggests that the region has a well-developed infrastructure for sports and recreational activities, providing opportunities for residents to engage in various sports and physical fitness pursuits. The availability of sports venues throughout Victoria indicates a commitment to promoting an active and healthy lifestyle for its residents. This can contribute to the overall well-being and quality of life in the region by offering accessible recreational options for individuals of all ages.



(a) Sport Map (b) Logo Meaning

Figure 4.7: Victoria number of sport facilities per suburb

4.3.2 Tweets Analytical Result

In this section, the relationships between tweets and scenarios mentioned above will be mainly discussed.

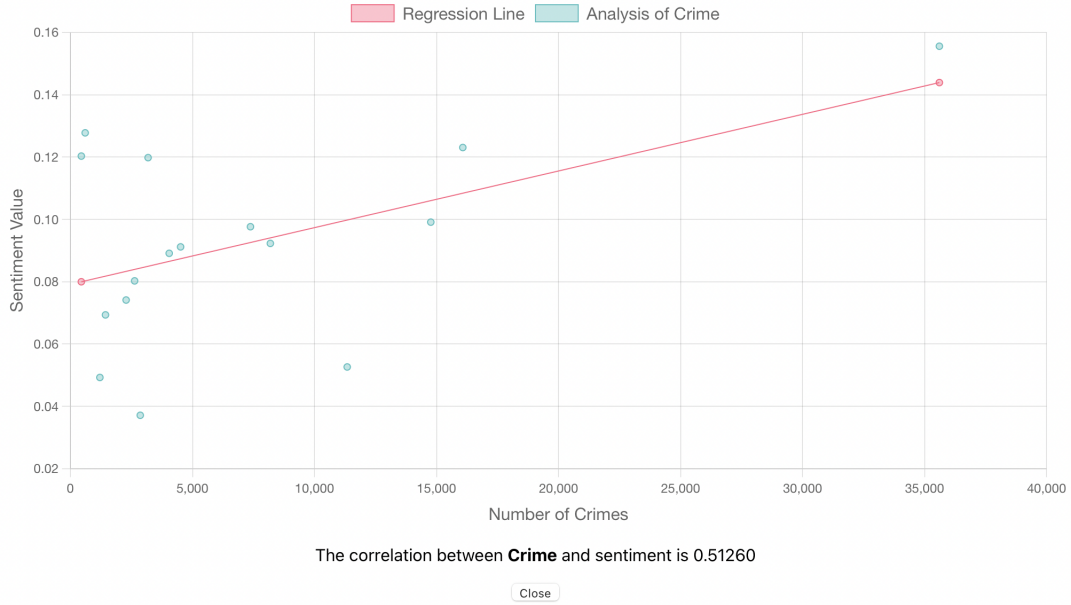


Figure 4.8: Correlation between number of crime and sentiment score

Figure 4.8 reveals a significant correlation between the sentiment score and the number of crimes (correlation coefficient: 0.51260). A higher incidence of crimes tends to generate feelings of insecurity and fear among residents, leading to a negative sentiment. This occurs because people become concerned about their well-being and the overall safety of

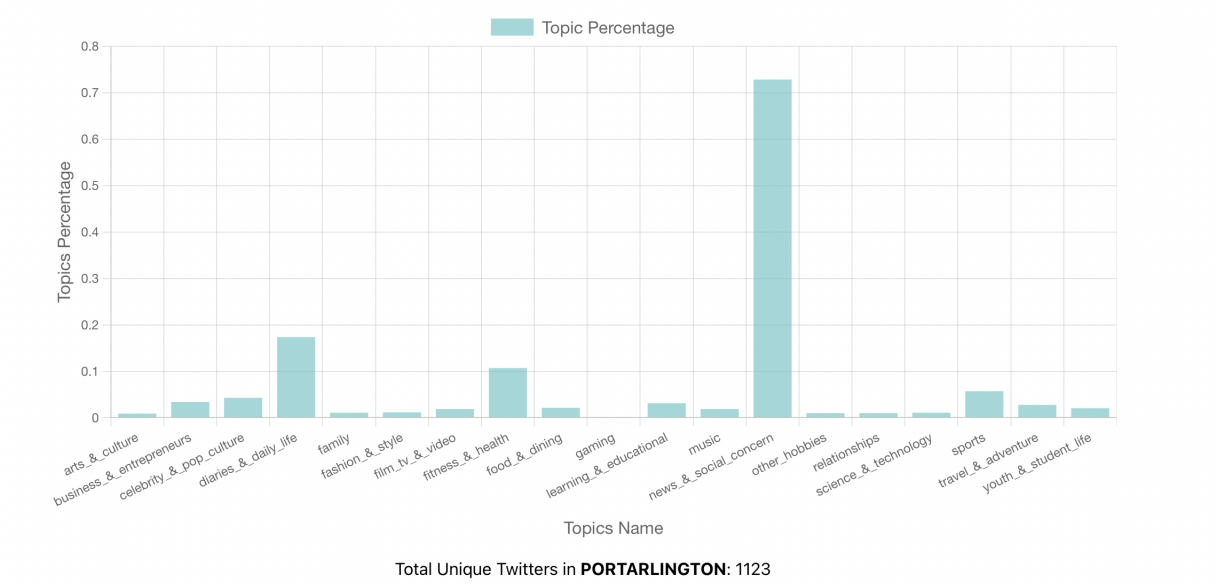


Figure 4.9: Twitter topics in high crime zone

their community. Moreover, high crime rates can erode trust in local institutions and law enforcement, as residents perceive a lack of safety and inadequate response to criminal activities. Consequently, negative sentiments towards authorities and a diminished sense of security can arise. Additionally, the impact of high crime rates extends to the quality of life in an area, restricting social activities, impeding economic growth, and diminishing residents' sense of belonging. These consequences contribute to a negative sentiment as individuals experience the adverse effects of crime on their lifestyle and opportunities.

From Figure 4.9, we can see that news & social concern are the top-mentioned topics that tweets sent from Protalington which is a high crime zone.

There is a negative correlation between age and sentiment score, which has value -0.15548. Older people may have experienced a broader range of life events, including personal challenges or losses, which can contribute to a more nuanced emotional outlook. These experiences can impact sentiment and potentially lead to a more balanced or less positive emotional response. Also, Older individuals may experience changes in their social networks, such as the loss of friends or family members, retirement, or a decrease in social interactions. These changes can impact their emotional well-being and potentially contribute to lower sentiment scores.

According to Figure 4.11, there is a correlation coefficient of 0.17910 between transport and the sentiment score. This indicates a positive correlation, albeit a relatively weak one, between the two variables. In other words, there is a slight tendency for higher sentiment scores to be associated with the transport scenario. A positive correlation suggests

Figure 4.10: Correlation between Age and Sentiment score

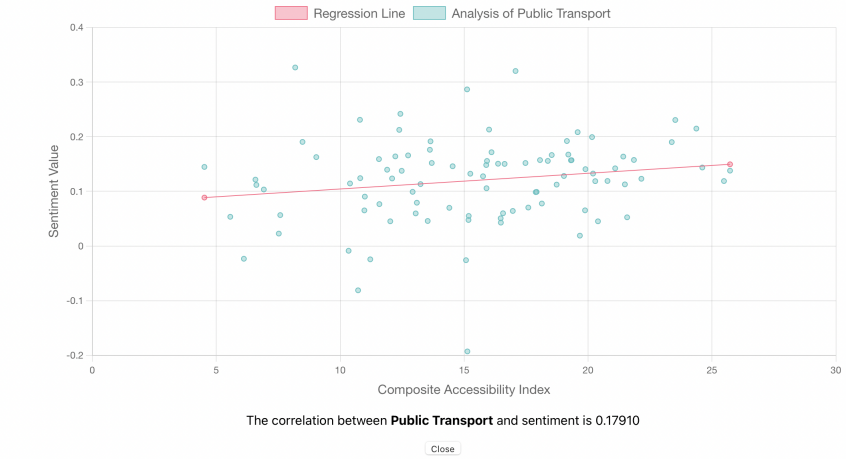


Figure 4.11: Correlation between Age and Sentiment score

that, on average, when the quality or availability of transport is better, there is a slight increase in the sentiment score. This could be attributed to several factors: Convenience and Accessibility, Mobility and Connectivity and Stress Reduction.

4.3.3 Scenario Mastondon

We analyze the proportions of various topics in data from Mastodon and Twitter, visually representing them with blue and red bars respectively in Figure 4.12. It is evident from the chart that both Twitter and Mastodon users predominantly discuss news and social concerns, followed by diaries and daily life topics. Notably, there is a significant difference in the percentage of posts related to sports, as Mastodon has very much fewer posts discussing this topic.

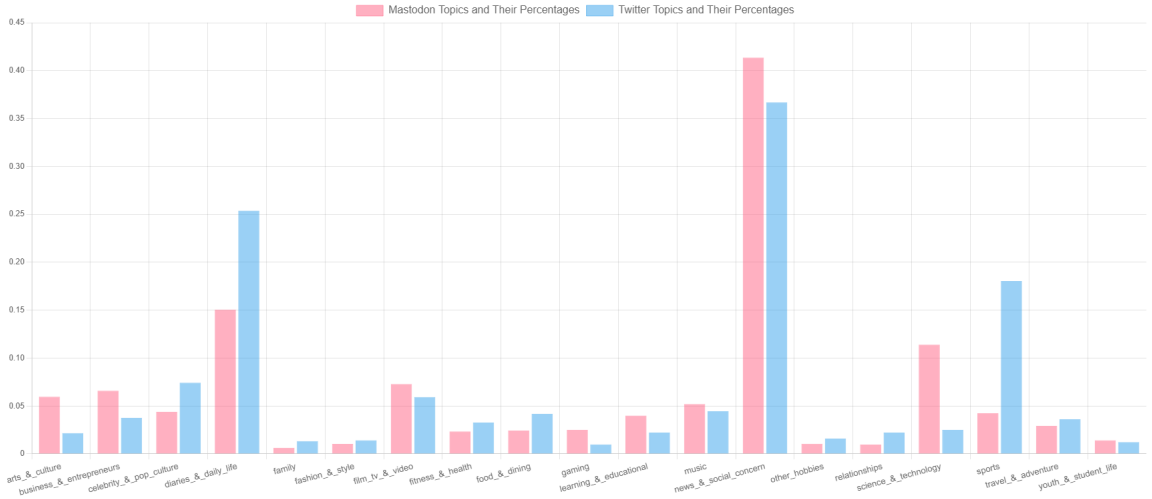


Figure 4.12: The proportions of different topics in Mastodon and Twitter data

In order to illustrate the distribution of sentiment scores for both tweets and Mastodon toots, we utilize boxplots. From Figure4.13, it is evident that the box representing Mastodon toots is larger, indicating a wider range of sentiment scores compared to tweets.

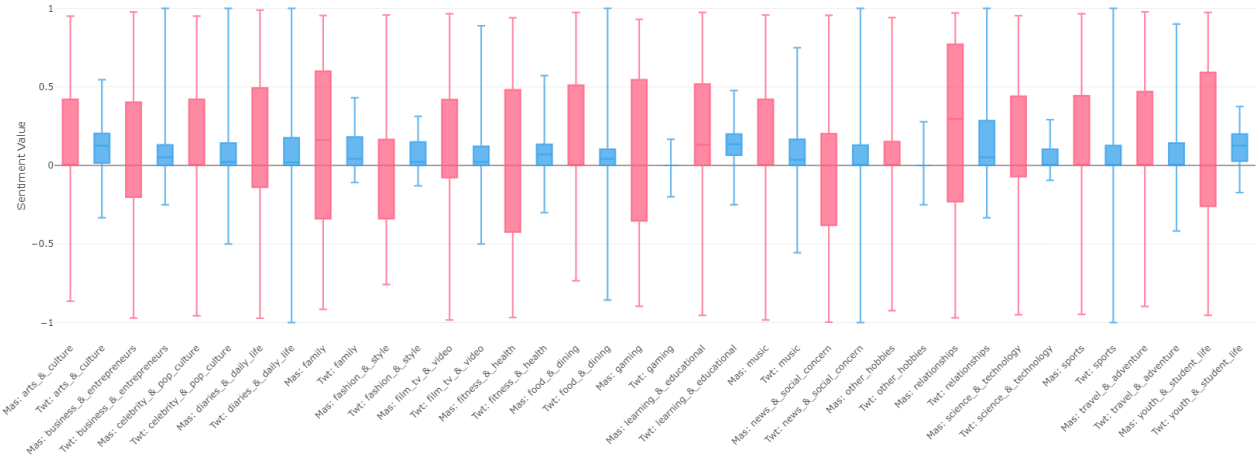
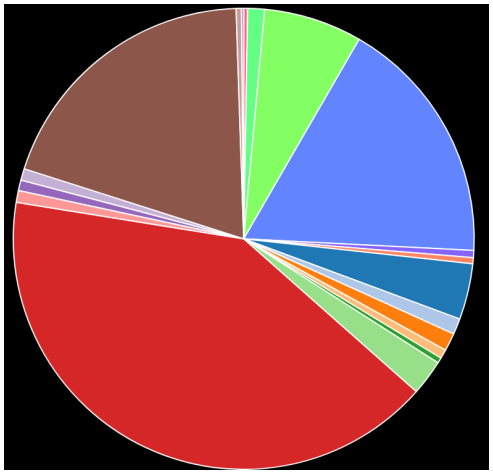
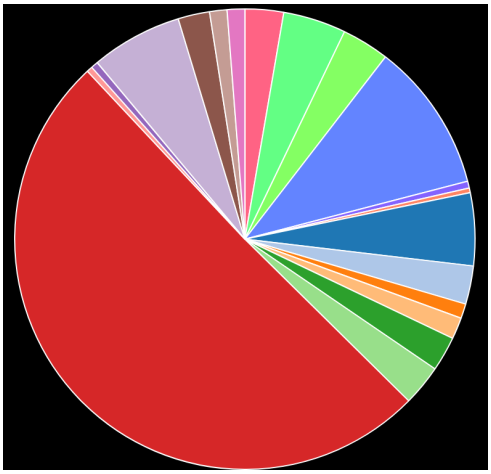


Figure 4.13: Sentiment value of different topics in Mastodon and Twitter data

Lastly, we utilize pie charts to examine the topic distribution within different sentiment score intervals. We divided the sentiment scores into four intervals, each with a step size of 0.5. For the purpose of clarity in our report, we only show the proportions of topics within the sentiment range of -1 to -0.5 and the sentiment range of 0.5 to 1.

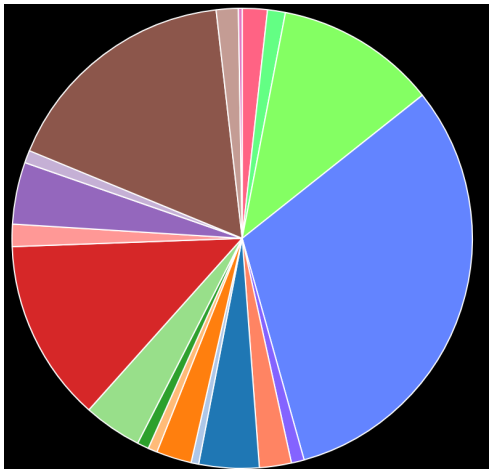


(a) Proportion of Twitter topics within sentiment range of -1 and -0.5

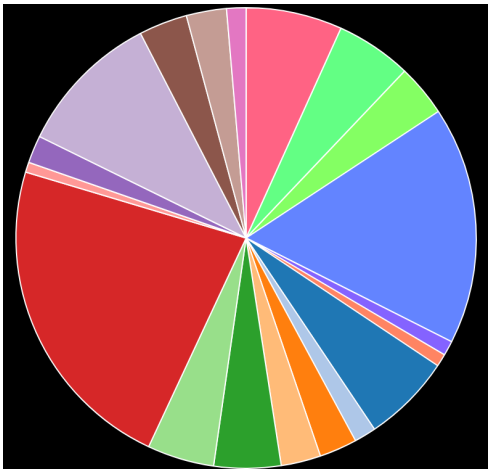


(b) Proportion of Mastodon topics within sentiment range of -1 and -0.5

Figure 4.14: Proportion of topics within sentiment range of -1 and -0.5



(a) Proportion of Twitter topics within sentiment range of 0.5 and 1



(b) Proportion of Mastodon topics within sentiment range of 0.5 and 1

Figure 4.15: Proportion of topics within sentiment range of 0.5 and 1

Figure 4.16: Content Topics

From Figure 4.14 and Figure 4.15, we can see that the distribution of topics varies across different sentiment intervals. Interestingly, an upward trend can be observed where higher sentiment scores correspond to an increased focus on daily life discussions. In contrast, there is a decline in the topic of news and social media concerns. This pattern is evident in both Twitter and Mastodon data.

To summarize, the choice of sentiment score intervals can significantly impact the distribution of topics. There seems to be a correlation between the content shared on social media and the emotions experienced by individuals. The data indicates that personal, positive experiences contribute to a higher level of happiness, while discussions surrounding news and social concerns may lead to a decrease in overall happiness.

5 Appendix

5.1 Team member contribution

Name	Contribution
Jiashu Wu	Data collection Data preprocessing CouchDB deployment CouchDB replication Perform data analysis
Junjie Wang	Data collection Data preprocessing Topic classification Sentiment analysis Mastodon harvester deployment CouchDB deployment Perform data analysis
Ruixiang Tang	Data collection Backend development with MapReduce Ansible for Backend Docker for Backend
Haodong Gu	Data collection Ansible for MRC Ansible for Mastodon Harvester Configure Couchdb Cluster Docker for front-end
Changwen Li	Data collection Frontend development

Table 5.1: Team member contribution

5.2 Video Link

The link for Ansible demonstration: https://youtu.be/VGHe7GJrFYQ

The link for Web App demonstration: https://youtu.be/jPuNLwwAvxo

5.3 Github Link

The link for Our GitHub repository: https://github.com/HaodongGU/comp90024-a2

References

[1] Dimosthenis Antypas, Asahi Ushio, Jose Camacho-Collados, Vitor Silva, Leonardo Neves, and Francesco Barbieri. Twitter topic classification. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3386–3400, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

[2] Xianjun Chen, Zhoupeng Ji, Yu Fan, and Yongsong Zhan. Restful api architecture based on laravel framework. In *Journal of Physics: Conference Series*, volume 910, page 012016. IOP Publishing, 2017.

[3] DV Kornienko, SV Mishina, SV Shcherbatykh, and MO Melnikov. Principles of securing restful api web services developed with python frameworks. In *Journal of Physics: Conference Series*, volume 2094, page 032016. IOP Publishing, 2021.

[4] PS Lokhande, Fankar Aslam, Nabeel Hawa, Jumal Munir, and Murade Gulamgaus. Efficient way of web development using python and flask. 2015.

[5] Amit M Potdar, DG Narayan, Shivaraj Kengond, and Mohammed Moin Mulla. Performance evaluation of docker container and virtual machine. *Procedia Computer Science*, 171:1419–1428, 2020.