THE UNIVERSITY OF MELBOURNE

Cluster and Cloud Computing COMP90024

Social Media Analytics Report

GROUP 64

ZHI HERN TOM *1068268*

HAONAN ZHONG *867492*

NI ZHANG *1081143*

YUMENG CHEN *1079520*

ZIYU QIAN *1067810*

Abstract

This report describes the design and implementation of a Cloud-based solution developed for harvesting and processing social media data from Twitter and Mastodon. The implemented system comprises multiple instances running in tandem on Melbourne Research Cloud (MRC), paired with an integrated CouchDB database for the storage of Mastodon toots, Twitter tweets, and data from the Spatial Urban Data Observatory (SUDO).

Furthermore, the report outlines the development of several analytical scenarios leveraging the MapReduce capabilities offered by CouchDB for comprehensive social media analytics. This study presents an innovative approach to harnessing the power of cloud technology and distributed systems in managing and analyzing social media data.

The source code for this project is available and can be found at the following GitHub repository URL:
`https://github.com/COMP90024-2023-SM1/Assignment-2-Australia-Social-Media-Analytics-on-the-Cloud`

The implemented webapp can be found at the following URL:
`http://172.26.129.235`

The setup explanation video can be found at the following URL:
`https://www.youtube.com/watch?v=8U8SuxAJn3U&t=6s`

The webapp demonstration video can be found at the following URL:
`https://youtu.be/d7fVR742AUM`

Contents

1 Introduction

The explosive growth of social media has led to an unprecedented availability of digital data, offering researchers an immense resource to explore various sociocultural phenomena. This project takes advantage of this phenomenon, focusing on Australia-based Twitter data and global Mastodon data to examine scenarios revolving around Christianity, depression, and the Russia-Ukraine War, all through the lens of social media users.

Twitter is a mainstream social media platform that allows users to chat and share their views and opinions on various topics. On the other hand, Mastodon, a decentralized social network platform, is favored by those looking for more advanced privacy features. By fetching data from multiple Mastodon servers, we aim to gain a more nuanced understanding of the discussions surrounding our target scenarios.

For our study, one **JSON** format Twitter dataset was provided, which contains approximately 52,533,742 tweets. Each **JSON** object of the dataset encapsulates core attributes that describe a given tweet. Please refer to section 5.1 for additional information on how the Twitter data was cleaned and preprocessed. We have also implemented scripts for collecting Mastodon toots using Mastodon API. Further notes on harvesting and processing Mastodon Toots can be found in section 5.2. The implemented system incorporates CouchDB database technology for data storage and enhanced MapReduce functionality. Detailed information about the CouchDB setup can be found in section 5.4.

2 System Architecture & Design

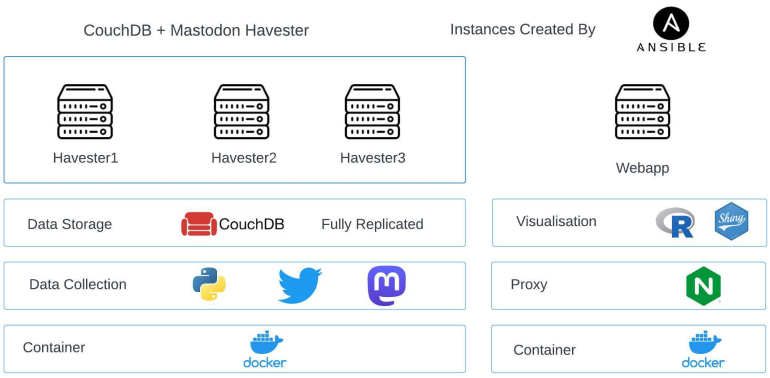

Figure 1: System Architecture

2.1 Key Components

2.1.1 Resource Allocation on MRC

The allocation of our instances on MRC [1] is based on the table below, which demonstrates the utilization of CPUs, memory, and the setup of the volume attached to each one.

Instance name	Number of VCPUs	Memory Size	Volume Size
Harvester 1	2	9GB	150GB
Harvester 2	2	9GB	150GB
Harvester 3	2	9GB	150GB
Webapp	2	9GB	50GB

Table 1: Instance Information

2.1.2 CouchDB

CouchDB [2] was the database solution for storing and analyzing **JSON** data collected from Twitter and Mastodon. This choice was based on several factors. Firstly, CouchDB is a document-oriented DBMS and provides native **JSON** support, allowing for the direct storage of data without the need for complex transformations. Secondly, CouchDB's web-ready property plays a crucial role in a web application's data and logic tiers. By eliminating the need for marshaling and unmarshaling data objects, it simplifies the computation process in the web app layer. Additionally, CouchDB's built-in replication and synchronization capabilities provide data availability and consistency across multiple instances, which is crucial for our project's requirements. The MapReduce functionality offered by CouchDB allows for efficient querying and analysis of large volumes of collected data, enabling us to extract valuable insights.

Our cluster was implemented with a trio of nodes, where **harvester1** was designated as the master node. Docker was utilized for configuration and connectivity, and the cluster was set up with full replication to ensure data redundancy and availability. The master node is responsible for processing query requests, while **harvester2** and **harvester3** serve as replica nodes, maintaining synchronized copies of the data. This configuration enables high availability, ensuring system reliability and fault tolerance to some extent.

2.1.3 Docker

Containerization simplifies the setup process and ensures consistency between the production and deployment environments by allowing virtual instances to run on a single host

operating system. Each container is responsible for holding the application and its related binaries while sharing hardware resources. This approach becomes especially valuable when working with multiple operating system distributions, as it provides manageability and streamlines the deployment process. In our implementation, we leverage Docker to launch the three nodes of the CouchDB cluster, as well as the web application and Nginx. The adoption of Docker and Docker Compose reduces the effort and time involved in setting up and configuring the system components, and thus it enhances efficiency and facilitates easier deployment,

2.1.4 Harvester

The two nodes **harvester2** and **harvester3** are responsible for fetching data from Mastodon servers and we use the Python library **Mastodon.py** to fetch toots. We fetched both legacy and streaming data from the **mastodon.social** server and **mastodon.world** server.

2.1.5 Web Application

The web application is built using **Shiny** [3], which is an R package that enables users to build interactive web interfaces straight from R, while the **Highcharter** package allows us to add more advanced interactivity into our visualizations to display additional information based on user selection. The implemented web application retrieves data from the database, renders the web pages, and visualizes the data. In addition, the Nginx reverse proxy acts as an intermediary between the users and the web application server. It receives incoming requests from users and forwards them to the appropriate server hosting the R Shiny application. This allows users to access the web content seamlessly without being directly connected to the application server.

3 User Guide

3.1 Prerequisites

Obtain the OpenStack RC file1 from the **Melbourne Research Cloud (MRC)** [1] as depicted in Figure 2 and place it in your current directory.

3.2 Instance Creation and Configuration

The system is equipped with automated deployment capabilities using **Ansible** [4] scripting.

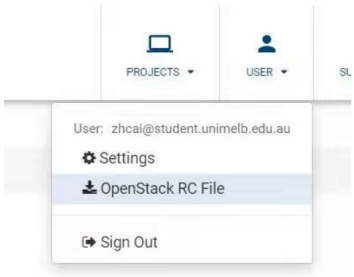

Figure 2: OpenStack File from MRC

Here is an example installation of **Ansible**. Ensure that **pip** is installed, and once you have pip, it is easy to install **Ansible**:

```
1 $ python3 -m pip -V
2 $ python3 -m pip install --user ansible
```

The next step is to generate a (RSA) key pair that would allow us to get access to the instances. There will be two files named "default.key" and "default.key.pub". Afterwards, the content of the public key will be uploaded to MRC when instances are created with Ansible. The private key will be then used to log in via SSH to VMs after they are launched.

```
1 $ ssh - keygen -t rsa -f default.key && chmod 600 default.key
```

Now we are all prepared and can proceed to run the scripts below. Ansible will then create the instances.

```
1 $ cd ansible
2 $ ansible-playbook -i inventory.ini myplaybook.yml
```

3.3 Harvester Initiation

To fetch, preprocess, and load legacy and streaming Mastodon data from each server to Couchdb, replace <mastodon-folder-path>with:

- mastodon_legacy_social
- mastodon_legacy_world
- mastodon_streaming_social

- mastodon_streaming_world

Upload web scraping code to MRC:

```
$ scp -i default.key -r <mastodon-folder-path> ubuntu@<ip-address>:~
```

Login to MRC:

```
$ ssh -i default.key ubuntu@<ip-address>
```

Create docker and run the code:

```
$ folder_name=<mastodon-folder-name>
$ cd $folder_name
$ sudo docker build -t $folder_name
$ sudo docker create --restart=always
    --network=host --name $folder_name $folder_name
$ sudo docker start $folder_name
```

Then proceed to run *twitter/main.py* to preprocess and upload Tweet data to CouchDB.

3.4 Web Application Initiation

```
ubuntu@webapp: cd ~/app/webapp
ubuntu@webapp: sudo docker-compose up --build
```

The web application will be housed on the local port 3838. However, clients will be able to access the application from outside the instance using port 80 through Nginx, which will then forward the requests to the web application.

4 Ansible - Deployment Automation

The shell script, dubbed *run-mrc.sh*, executes the *main.yaml* playbook using *private.key* as the SSH private key.

4.1 Instance Creation

In the phase of instance creation, five specific roles are necessary:

1. Role *openstack/common*: This role is responsible for the installation of pip and openstacksdk on the control node.

2. Role *openstack/volume*: This role is tasked with volume creation.

3. Role *openstack/security-group*: This role is responsible for establishing the rules for security groups.

4. Role *openstack/instance*: This role is tasked with creating instances on MRC.

The availability zone, named *melbourne-qh2-uom*, remains unchanged for all instances and volumes. The OS image utilized is NeCTAR Ubuntu 20.04 LTS (Focal) amd64. The network named qh2-uom-internal is used, and it restricts access solely from the University Local Area Network.

4.2 Security

In order to ensure secure communication and efficient coordination within the cluster, we have implemented a comprehensive security framework that includes the establishment of a dedicated security group called "group-64-2." This security group plays a crucial role in maintaining the integrity and confidentiality of the cluster's communication channels.

The security group "group-64-2" is designed to facilitate seamless and controlled communication among all instances within the cluster. By defining and configuring strict inbound and outbound rules, we can carefully manage the flow of network traffic and permit only authorized connections between instances. The ports we open, such as 5984 and 9100, can be targeted for potential exploits. Therefore, encrypting traffic that passes through these ports is vital. For instance, we can utilize the Transport Layer Security (TLS) protocol to provide communication security over a computer network. Using certificates for encryption also plays a crucial role in enhancing security.

Another important aspect to focus on is ensuring that we only use secure, encrypted connections for intra-cluster communication. This will further help to protect sensitive data in transit.

4.3 Volume

Our configuration illustrates the allocation of volumes on MRC with varying sizes. It outlines four volumes: **webapp**, **harvester-1**, **harvester-2**, and **harvester-3**. **Webapp** is set to use 50 storage units, which is typically smaller due to the nature of its contents, mainly codes and libraries, that don't require much storage space. The other three volumes, **harvester-1**, **harvester-2**, and **harvester-3**, each have 150 units allocated, to handle heavier workloads such as storing large data sets. Together they add up to 500 units, maximizing the available storage.

Additionally, the design of these volumes offers a critical advantage in terms of data reliability and fault tolerance. In case of instance failure, these volumes can simply be detached and remounted to a new instance, effectively preventing data loss and ensuring the continuity of operations. This strategy provides an efficient solution for storing data and maintaining resilience in our digital operations.

4.4 Virtual Machine Configuration

After creation, the following three roles are executed for all instances:

1. **setup/common:** Facilitates the installation of essential packages and dependencies, including Docker, Docker Compose, Pip, and Couchdb, within the Ubuntu environment on Melbourne Research Cloud (MRC).

2. **setup/couchdb:** This task will run a script named **couchdb-setup.sh**. It automates the deployment and setup of a CouchDB cluster with version 3.2.1 using Docker. It pulls the CouchDB image, creates a data directory, and creates a Docker container for each node in the cluster. It then enables and adds the nodes to the cluster, finishes the cluster setup on the master node, and finally checks the cluster status.

3. **setup/mount:** Automates the process of setting up an XFS file system, specifically on the '/dev/vdb' device. It manages the installation of necessary packages, creation of a directory '/data', mounts the device, and configures the fstab file to ensure the mount is persistent across reboots.

5 Data Delivery

5.1 Twitter

Twitter is a popular social media platform that allows users to send and receive short posts known as "tweets." Twitter offers an interactive environment where people exchange thoughts, news, or personal insights. The platform boasts a global user base of approximately 396.5 million, including approximately 5.8 million Australian users. However, Twitter has faced backlash over concerns related to user privacy, misinformation, and online harassment. Consequently, other privacy-focused social media platforms like Mastodon are gaining traction.

5.1.1 Preliminary Twitter Data Analysis

The project utilizes Twitter data sourced from the **Australian Data Observatory (ADO)**, totaling 63.2 GB. This dataset encompasses tweets published in Australia over a six-month span, from February 10, 2022, to August 10, 2022. Out of the entire volume of 52,533,742 tweets, only 3,233,659 tweets, which include geolocation information, are pertinent to this study. As a result, tweets without geolocation data are excluded from our analysis.

5.1.2 Preprocessing Tweets

Leveraging the power of the **Message Passing Interface (MPI)** [5], we significantly accelerated our tweet data preprocessing procedure. **MPI** allowed us to distribute the computation across multiple cores. By decomposing the large volume of tweet data into smaller and more manageable chunks and processing them simultaneously, we significantly reduced the total processing time. After retrieving each tweet, preprocessing is done to streamline its structure and content before storing it in a CouchDB database. Figure 3 shows a sample dictionary of a single unprocessed tweet.

The fields chosen for more detailed examination include: `"id"`, `"created_at"`, `"lang"`, `"public_metrics"`, `"hashtags"`, `"tokens"`, `"text"`, `"sentiment"`, and `"places"`. Furthermore, we'll derive the **Greater Capital Cities (GCC)** code from the `"full_name"` field for comparison with **SUDO** data discussed in a later section. Notably, the **GCC** code pertains to the Australian Bureau of Statistics' term **Greater Capital City Statistical Area (GCCSA)** [6], which examines the socio-economic extent of each of the eight Australian state and territory capitals.

We assign tweets to their corresponding **GCCSA**s using a multi-step process:

1. We first use an Australian location dataset *sal.json*, which contains detailed information about suburbs in each GCCSA. This dataset serves as our reference for mapping tweets to their respective GCCSAs.

2. To be considered for further analysis, a tweet must have a `""full_name"` field that follows the specific format: "suburb, state". This format is essential for accurate mapping to the GCCSAs. As a result, there're 2,843,498 tweets with a valid format.

3. Once a tweet passes the format check, it's then assigned two unique GCC codes corresponding to the state in the tweet's `"full_name"` field. For instance, a tweet tagged with "Abbotsbury, New South Wales" is linked with two GCC codes: `"1gsyd"`

```
{
  "id":"1491725646330220544",
  "key":[
    2022,
    2,
    10,
    "1294802004892463104",
    "1248056147266768897",
    "1491725646330220544"
  ],
  "value":{
    "tags":"",
    "tokens":"The|only|interesting|WAG|ever|seen|was|the|arse|end|dog|and|had|balloon|attached"
  },
  "doc":{
    "_id":"1491725646330220544",
    "_rev":"1-f253da993aa8a0bc8ba846d2f40257c5",
    "data":{
      "author_id":"1248056147266768897",
      "conversation_id":"1294802004892463104",
      "created_at":"2022-02-10T10:48:03.000Z",
      "entities":{
        "mentions":[
          {
            "start":0,
            "end":14,
            "username":"AssExtranious",
            "id":"860758238"
          },
          {
            "start":15,
            "end":28,
            "username":"theheraldsun",
            "id":"35466620"
          },
          {
            "start":29,
            "end":38,
            "username":"walkleys",
            "id":"24840047"
          }
        ]
      },
      "geo":{
        "place_id":"01864a8a64df9dc4"
      },
      "lang":"en",
      "public_metrics":{
        "retweet_count":0,
        "reply_count":0,
        "like_count":0,
        "quote_count":0
      },
      "text":"@AssExtranious @theheraldsun @walkleys The only interesting WAG I've ever seen was on the arse end of a dog and it had a balloon attached",
      "sentiment":0.08333333333333333
    },
    "includes":{
      "places":[
        {
          "full_name":"Melbourne, Victoria",
          "geo":{
            "type":"Feature",
            "bbox":[
              144.593741856,
              -38.433859306,
              145.512528832,
              -37.5112737225
            ],
            "properties":{

            }
          },
          "id":"01864a8a64df9dc4"
        }
      ]
    },
    "matching_rules":[
      {
        "id":"1491447910617485312",
        "tag":""
      }
    ]
  }
}
```


Figure 3: Sample Unprocessed Tweet

and `"1rsyd"`. This step allows us to fine-tune our mapping to coincide with the state-level geographical data.

4. The final assignment of a GCC code to a tweet only occurs when both the suburb name and the state in the tweet's `"full_name"` field match entries in our *sal.json* location dataset. For example, if "Abbotsbury" is listed under the GCC code "1gsyd" in our location dataset, then any tweet with "Abbotsbury, New South Wales" in its `"full_name"` field will be tagged with `"1gsyd"` as its final GCC code.

This step-by-step process ensures accurate and precise mapping of tweets to their respective GCCSAs, allowing for detailed, location-specific analysis. As a result, a sample preprocessed tweet saved to CouchDB is shown in Figure 4.

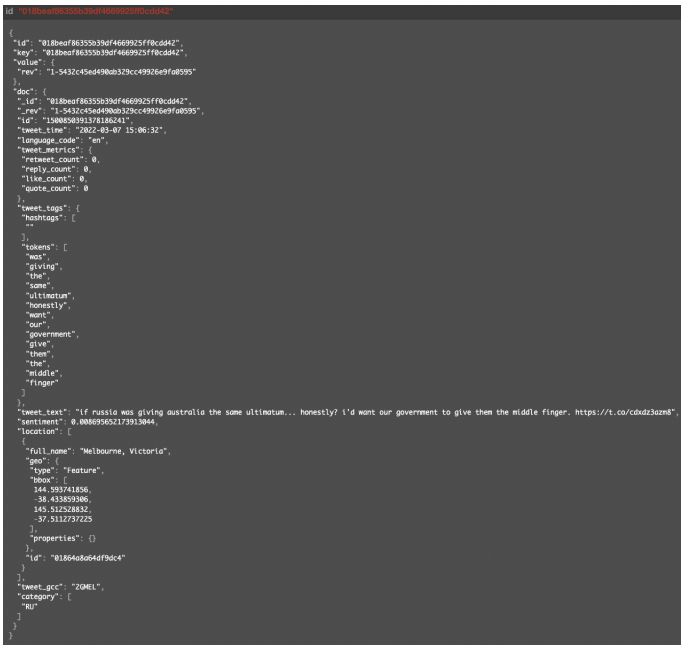

Figure 4: Sample Preprocessed Tweet on Couch DB

5.2 Mastodon

Mastodon is a decentralized, open-source social networking service with blogging services similar to Twitter. Mastodon servers are administered independently and establish their own policies and moderation standards. Compared to Twitter, Mastodon servers offer more granular privacy settings and a content warning feature. As a result, some users turn to Mastodon as an alternative to mainstream social media platforms such as Twitter due to concerns about privacy, advertising, and content moderation policies.

5.2.1 Harvesting Toots

We have chosen 2 servers to harvest data from: **mastodon.social** - the flagship instance of Mastodon run by the creator of Mastodon, one of the largest and most well-known instances, and **mastodon.world** – a generic Mastodon server for anyone across the world to use. From each server, we harvested both legacy and real-time posts (we've ensured that the timelines don't overlap), this will allow us to extract sufficient data related to our scenarios for further analysis.

The Mastodon API doesn't support fetching legacy data from a specific time frame, in turn, we have decided to harvest 1,000,000 legacy toots going backward from the date of harvest from each server. The number of legacy toots to fetch, "total_limit", can be ad-

justed in *./harvester/mastodon_legacy_social/mastodon_couchdb_access.py* and *./harvester/mastodon_legacy_world/mastodon_couchdb_access_world.py* according to the amount of legacy data one wishes to harvest starting from the current date. Two parameters "start_date" and "end_date" determine the date range for legacy data extraction, this can be adjusted in *./harvester/mastodon_legacy_social/main.py* and *./harvester/mastodon_legacy_world/main.py*.

Real-time toots were streamed using the Python library **Mastodon.py** [8] which facilitates interactions with the API directly. A **StreamListener** is defined for each server to stream toots as they are posted. In our set-up, **harverster2** is responsible for harvesting legacy data, and **harverster3** is responsible for harvesting real-time data. We initiated **harverster3** right after **harverster2**, this will ensure that there is no overlap between data fetched by the 2 harvesters and that it will give us a continuous timeline of toots.

5.2.2 Preprocessing Toots

As each toot is fetched, some preprocessing is done to simplify the structure and content before saving it to a CouchDB database. A typical return dictionary for a single toot is shown in Figure 5. We extract selected fields including "id", "created_at", "content", "tags", and "language". An additional field "toot_category" is also created after each toot is retrieved, which indicates if a toot can be used for any of the 3 scenarios we have chosen for our analysis. Figure 6 shows a simplified toot saved to CouchDB after extracting relevant fields, re-formatting datetime, and using **html.parser** from the **bs4.BeautifulSoup** library to process toot content.

5.2.3 Rate Limit Handling

The Mastodon API has a rate limit of 300 requests per 5-minute time frame. To handle this, we have implemented a measure to detect when the Mastodon API rate limit has been exceeded and then pause the script until the rate limit resets, at which point it continues making requests. More specifically, upon receiving a HTTP response, the script checks if the status code is **429**, which indicates that the rate limit has been exceeded. In such a case, it retrieves the **X-RateLimit-Reset** header from the response which indicates when the rate limit will reset. It then calculates the time difference between this reset time (converted to a **datetime** object) and the current time to determine how long the script should pause before sending further requests. A notification message is printed, detailing how long the program will wait. Finally, the script is put to sleep for the duration of the calculated waiting time plus an additional second to ensure it does not hit the rate limit immediately

```
mastodon.toot("Hello from Python")
# Returns the following dictionary:
{
    'id': # Numerical id of this toot
    'uri': # Descriptor for the toot
        # EG 'tag:mastodon.social,2016-11-25:objectId=<id>:objectType=Status'
    'url': # URL of the toot
    'account': # User dict for the account which posted the status
    'in_reply_to_id': # Numerical id of the toot this toot is in response to
    'in_reply_to_account_id': # Numerical id of the account this toot is in response to
    'reblog': # Denotes whether the toot is a reblog. If so, set to the original toot dict.
    'content': # Content of the toot, as HTML: '<p>Hello from Python</p>'
    'created_at': # Creation time
    'reblogs_count': # Number of reblogs
    'favourites_count': # Number of favourites
    'reblogged': # Denotes whether the logged in user has boosted this toot
    'favourited': # Denotes whether the logged in user has favourited this toot
    'sensitive': # Denotes whether media attachments to the toot are marked sensitive
    'spoiler_text': # Warning text that should be displayed before the toot content
    'visibility': # Toot visibility ('public', 'unlisted', 'private', or 'direct')
    'mentions': # A list of users dicts mentioned in the toot, as Mention dicts
    'media_attachments': # A list of media dicts of attached files
    'emojis': # A list of custom emojis used in the toot, as Emoji dicts
    'tags': # A list of hashtag used in the toot, as Hashtag dicts
    'bookmarked': # True if the status is bookmarked by the logged in user, False if not.
    'application': # Application dict for the client used to post the toot (Does not federate
                   # and is therefore always None for remote toots, can also be None for
                   # local toots for some legacy applications).
    'language': # The language of the toot, if specified by the server,
                # as ISO 639-1 (two-letter) language code.
    'muted': # Boolean denoting whether the user has muted this status by
             # way of conversation muting
    'pinned': # Boolean denoting whether or not the status is currently pinned for the
              # associated account.
    'replies_count': # The number of replies to this status.
    'card': # A preview card for links from the status, if present at time of delivery,
            # as card dict.
    'poll': # A poll dict if a poll is attached to this status.
```

Figure 5: Return Toot Dictionary

after it resets. After the sleep duration, it resumes the process with the **continue** command.

The above measure is only implemented when retrieving legacy data and not streaming data. There are 2 reasons for this approach, firstly, we rarely hit the rate limit while harvesting real-time toots as compared to continuously harvesting readily available legacy toots, and secondly, real-time toots are harvested using **Mastodon.py**, which automatically handles rate limits in the exact manner as we did for legacy data by using a default "wait" mode, i.e., once a request hits the rate limit, **Mastodon.py** will wait until the rate limit resets and then try again.

5.3 Tweet/Toot Classification

In order to determine whether a tweet/toot is relevant to any of the 3 scenarios we have chosen to analyze (Religion - Christianity, Depression, Russo-Ukraine War), we constructed an English keyword dictionary for each case. Each of the English keywords then gets translated into 11 other languages that include: simplified Chinese, Arabic, Vietnamese, Punjabi, Greek, Italian, Filipino, Hindi, Spanish, Japanese, and Korean. Our approach is to

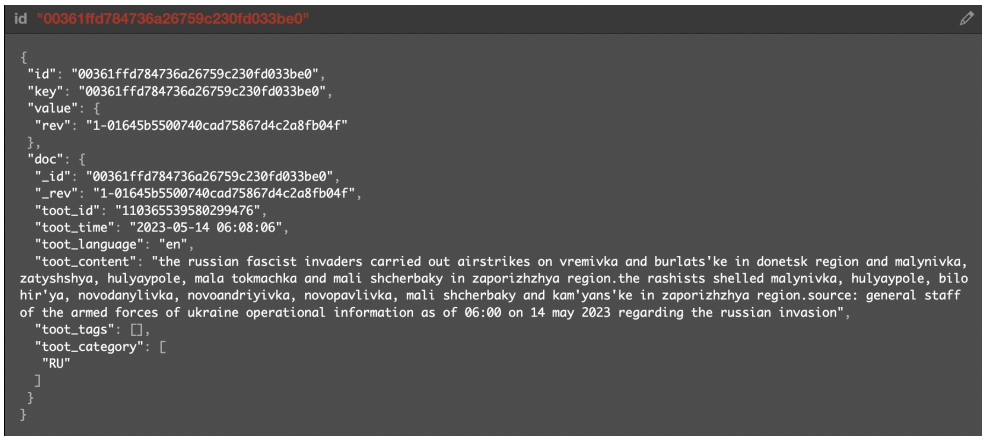

Figure 6: Simplified Toot on CouchDB

search the entire content string, and if any relevant keyword exists, then we tag the post as ”Religion”, ”Depression”, or ”RU” to support further analysis. After translation, we removed keywords of less than 3 characters to avoid accidentally tagging posts as relevant due to a frequently appearing keyword that might only have meaning in one language. We also converted entire content strings to lowercase characters to ensure accurate keyword matching. Each post can be tagged with a maximum of 3 labels since we believe it is possible for the 3 scenarios to overlap, e.g., it would make sense to tag ”I just can’t take it anymore, I feel so lost and exhausted, may the lord please help me.” as both ”Religion” and ”Depression”.

5.3.1 Religion Related

Keywords include: ”god”, ”jesus”, ”holy spirit”, ”bible”, ”prayer”, ”pray”, ”faith”, ”worship”, ”church”, ”mass”, ”sacrament”, ”eucharist”, ”communion”, ”baptism”, ”baptise”, ”baptize”, ”confession”, ”amen”, ”bless”, ”hallelujah”, ”catholic”, ”christian”, ”christianity”, ”salvation”.

5.3.2 Depression Related

Keywords include: ”sad”, ”lonely”, ”isolated”, ”hopeless”, ”helpless”, ”tired”, ”exhausted”, ”miserable”, ”anxious”, ”stressed”, ”overwhelmed”, ”worthless”, ”guilty”, ”suicidal”, ”suicide”, ”numb”, ”desperate”, ”broken”, ”lost”.

5.3.3 Ukraine-Russia War Related

Keywords include: ”russia”, ”ukraine”, ”putin”, ”zelenskyy”, ”kyjv”.

5.4 CouchDB Views

5.5 SUDO

The **Spatial Urban Data Observatory (SUDO)** [7] combines established open-source datasets and tools established by the MeG team with new data collections including social media data, national bushfire data, and air quality records. The data from **SUDO** is employed in a comparative analysis with the Twitter and Mastodon data, allowing for a location-based examination of the posts.

For the sake of maintaining consistency, all the data that we've utilized from **SUDO** are exclusively from the year 2016 and have been downloaded in CSV format. Unlike previous Twitter and Mastodon datasets, these CSV files weren't uploaded to CouchDB because there wasn't a requirement for a map-reduce operation. Instead, we stored these files on Github and directly loaded them into our **R Shiny** dashboard for use in our analysis.

5.5.1 Selected Datasets

1. **ABS - Data by Region - Population & People (GCCSA) 2016**

   Organisation: AU_Govt_ABS
   Description: The dataset from ABS Data by Region provides population statistics in 2016 based on GCCSA boundaries. It includes data on various demographic factors, sourced from both ABS and non-ABS collections.
   Selected Fields: Estimated Resident Population - Summary Statistics - As of 30 June Persons (no.), Religious Affiliation - Census Christianity (%)

2. **GCCSA-P13 Language Spoken at Home by Sex-Census 2016**

   Organisation: AU_Govt_ABS_Census
   Description: This dataset, sourced from the 2016 Census, provides information about the most common languages spoken at home, sorted by gender and based on 2016 GCCSA boundaries.
   Selected Fields: Speak English Only Persons, Total Persons

6 Results & Discussion

We presented our data analytics insights in a web application developed with **R Shiny**. Our web application consists of 4 tabs; a home tab for summary statistics and a separate tab

for each scenario. This section will outline our web application's capabilities and highlight key insights from our analysis of social media posts in relation to demographic data from SUDO whilst making a comparison between tweets vs. toots. Note that figures and graphs are the content displayed as of 22nd May 2023 and are subject to change as more live toots are harvested.

6.1 Summary Statistics

The summary tab begins by quantifying the total number of tweets and toots obtained, as shown in Figure 7, this figure will progressively increase as we fetch new toots.

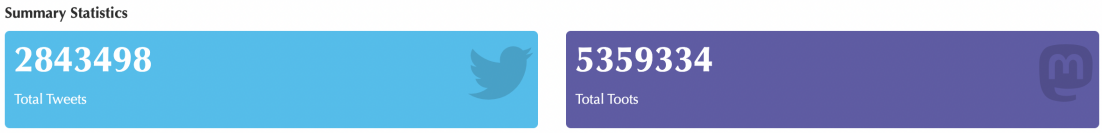

Figure 7: Summary Statistics of Total Number of Tweets and Toots

Figure 8 employs a heatmap to compare the distribution of tweet frequency with the population within each **GCCSA** across Australia. The population component is represented by the base heatmap in various shades of pink, while tweet frequency is symbolized by purple bubbles for greater capital cities and blue bubbles for rural areas. Interactivity is facilitated through hover-activated popups displaying the specific population or tweet frequency, and filters on the top right allow the selection of a specific date range as well as urban/rural adjustments. This heatmap illustrates a positive correlation, suggesting that tweet frequency tends to increase with population size.

The left of Figure 9 presents a line graph tracing the monthly accumulation of tweets from February 2022 to August 2022. Although February and August show reduced volume, this is attributable to incomplete data for these months (19 days in February and 10 days in August). Between March and July, tweet counts were relatively steady with a slight peak in May, averaging around 482,000 per month.

Concluding our visual summary in the home tab, the right side of Figure 9 presents a word cloud, summarizing the prominent themes embedded within the tweet content. The cloud reveals a dominance of positively connoted terms such as ”like”, ”good”, ”great”, ”love”, and ”well”. Other frequently appearing terms include common words in conversation such as ”one”, ”time”, ”now”, and ”people”.

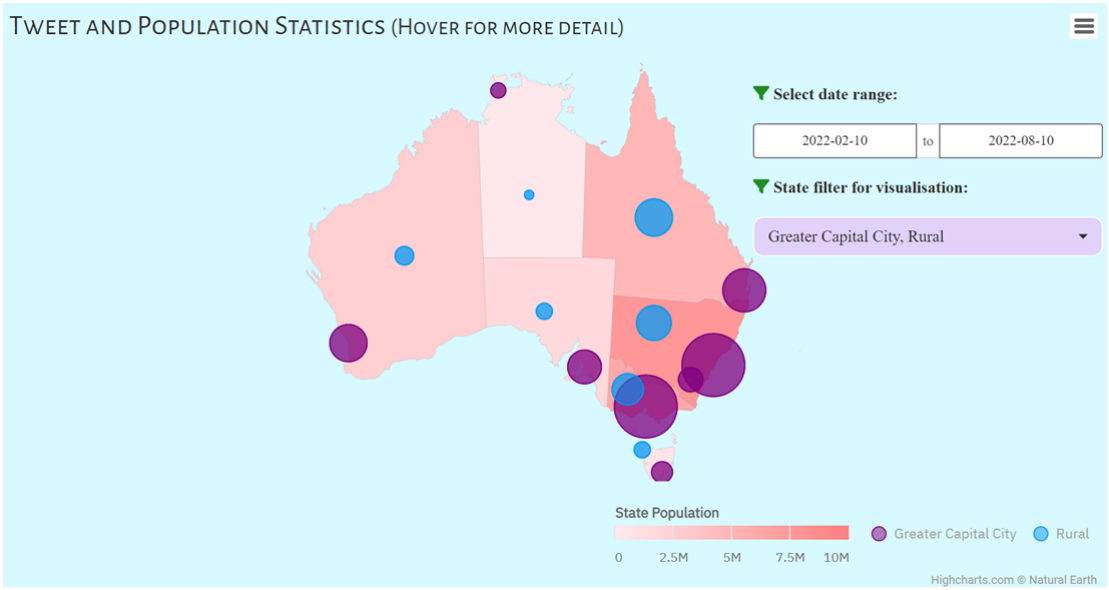

Figure 8: Heatmap of Tweets and Population Statistics

6.2 Scenario 1 - Religion

Christianity is the most prevalent religion in Australia, with 51.11% of Australians identifying as Christian according to data obtained from **SUDO**, this is shown in Figure 10. Christianity, like any other religion, can influence societal norms and individual behavior. With this in mind, we aim to study the relationship between Christianity proportion and the percentage of tweets/toots associated with Christianity across various **GCCSA** locations in Australia.

An interesting observation from Figure 10 is that the percentage of Christianity-related toots is more than double that of Christianity-related tweets. This may be due to the fact that all tweets considered in our analysis are posted within Australia while Mastodon has users from all over the world. From Figure 11 we see that users from the United States and Germany account for approximately 50 percent of total users from the **mastodon.social** server [9], and both countries are shown to have a higher Christianity proportion than Australia as of 2016. Similarly, the United Kingdom and the United States combined give over 50 percent of total users from the **mastodon.world** server [10], and the average Christianity proportion across the two countries is also higher than that of Australia. This suggests that the higher percentage of Christianity-related toots is a result of more international users from countries where Christianity is more prevalent. In addition, Mastodon offering more granular privacy settings may also have an impact on this result, where users feel more comfortable making religion-related posts on Mastodon as compared to Twitter.

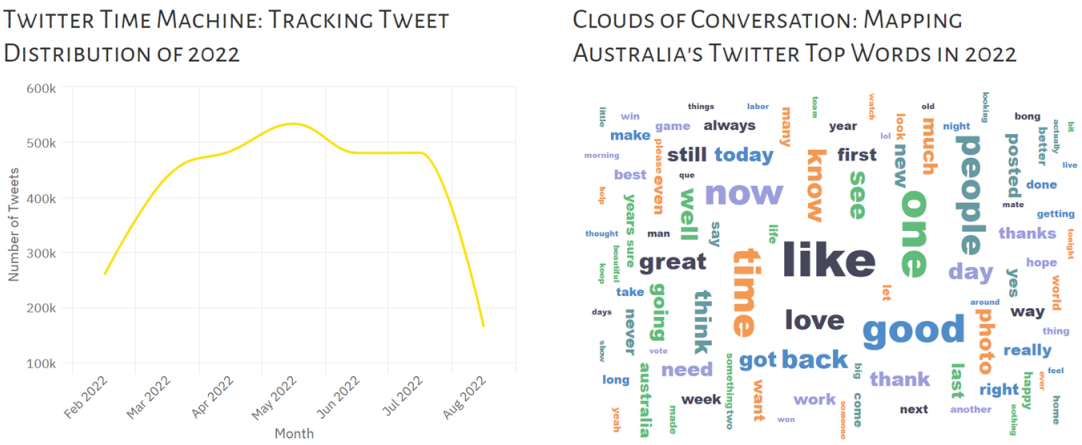

Figure 9: Monthly Trend of Tweet Volume, Word Cloud of Tweets

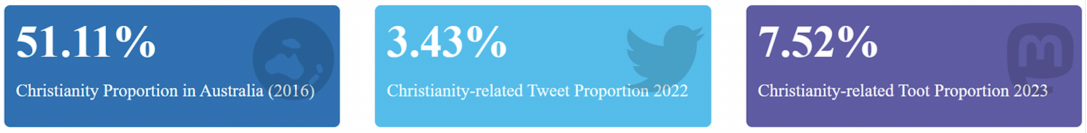

Figure 10: Summary Statistics of Christianity Proportion in Australia and Christianity-related Tweet/Toot Proportions

Having a closer look at tweets together with supporting demographic data from **SUDO**, Figure 12 displays the percentage of Christianity-related tweets layered on top of the percentage of Christian population at each **GCCSA** location. We can generally observe consistency between the two percentages, however, Hobart and Darwin display a notable discrepancy. Considering Northern Territory and Tasmania are two of the three states and territories with a small population under 600,000, we believe this discrepancy can be attributed to limitations due to the small sample size of tweets from these states

Figure 13 analyses the volume of Christianity-related tweets from February 2022 to August 2022. We see a gradual increase in volume from February to mid-May, then a small dip in June, then slowly rising again until our cut-off day in August. Below are some events related to the Christian Calendar that could explain this trend:

- **Lent and Easter**: The period from February to April includes the Christian season of Lent and the Easter holiday. Easter is a major event that often comes with a large amount of social media activity prior to and after the Easter holidays.
- **Whitsunday**: Whitsunday is celebrated 50 days after Easter Sunday, which would typically place it in mid-May.

	Country with most Users	*Country with 2nd most Users*	*Country with 3rd most Users*	*Country with 4th most Users*	*Country with 5th most Users*
mastodon.social User Proportion	United States 28.81%	Germany 18.89%	Turkey 4.90%	United Kingdom 4.34%	France 3.75%
mastodon.social Christianity Proportion	United States 73.70%	Germany 64.50%	Turkey < 1.00%	United Kingdom 41.00%	France 51.10%
mastodon.world User Proportion	United Kingdom 32.04%	United States 26.78%	Greece 9.20%	New Zealand 8.68%	Germany 3.71%
mastodon.world Christianity Proportion	United Kingdom 41.00%	United States 73.70%	Greece < 1.00%	New Zealand 37.20%	Germany 64.50%

Figure 11: Mastodon User Profile and respective Christianity Profile by Country

- **Feast of St. James and the Transfiguration:** These are both notable Christian feasts that take place in July and August.

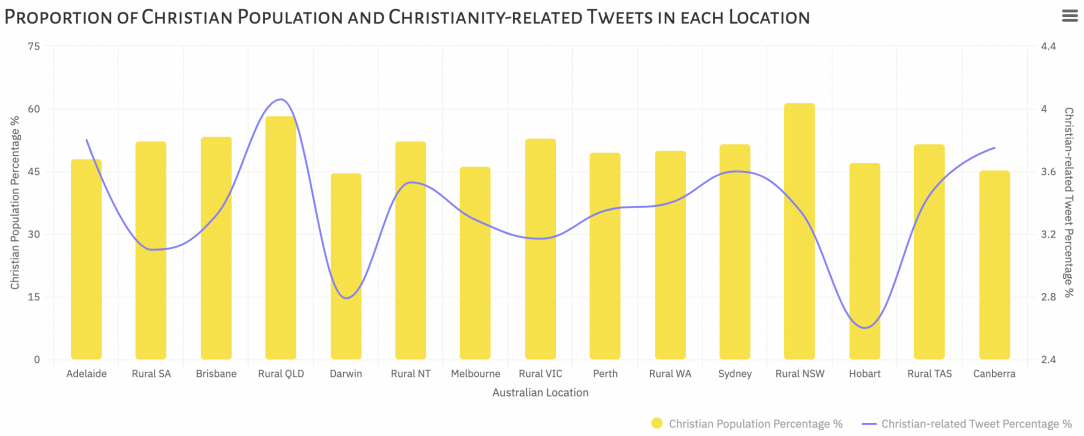

Figure 12: Proportion of Christian Population and Christianity-Related Tweets by GCCSA

6.3 Scenario 2 - Depression

Depression is a significant health issue in Australia, as well as in many other countries. According to the Australian Bureau of Statistics, approximately 9 percent of the population reported having a mood disorder such as depression at one point in life [11]. Many individuals suffering from depression turn to social media and online communities for support. This scenario aims to provide an in-depth analysis of depression-related tweets/toots.

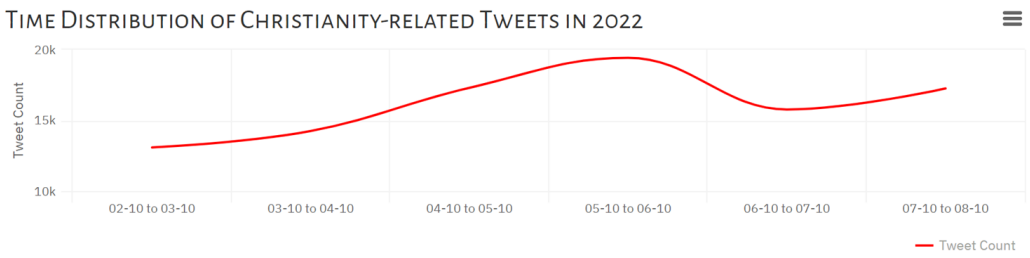

Figure 13: Volume of Christianity-Related Tweets

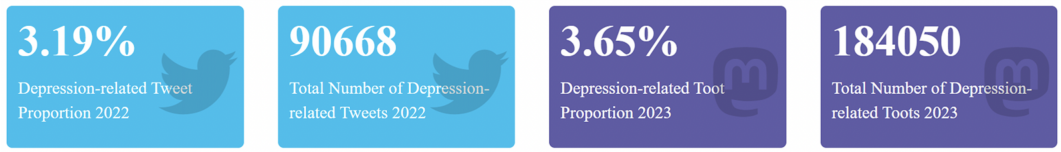

Figure 14: Summary Statistics of Depression-Related Tweet/Toot

From Figure 14 we can see that the percentage of depression-related tweets is slightly lower than that of depression-related toots. Once again, we believe this is due to more advanced privacy functions offered by Mastodon, which creates an environment with a better sense of security for individuals suffering from depression to share their stories or seek advice.

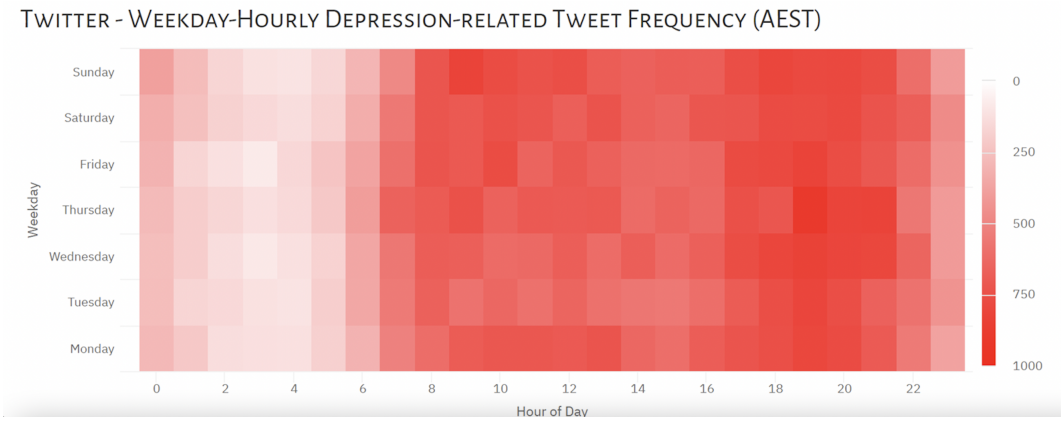

Figure 15: Weekday-Hourly Drepression-Related Tweet Frequency

We then look at the hour-of-day and day-of-week when depression-related tweets are made (Figure 15) as compared to toots (Figure 16), both are shown in **Australian Eastern Time (AEST)**. It can be observed that most depression-related tweets are made between 6AM - 10PM with a slight bias towards Monday and Sunday. We assume this is because 6AM - 10PM is when most people are active and Monday and Sunday are both associated with the reluctance of starting a new week. Interestingly, Figure 16 demonstrates an al-

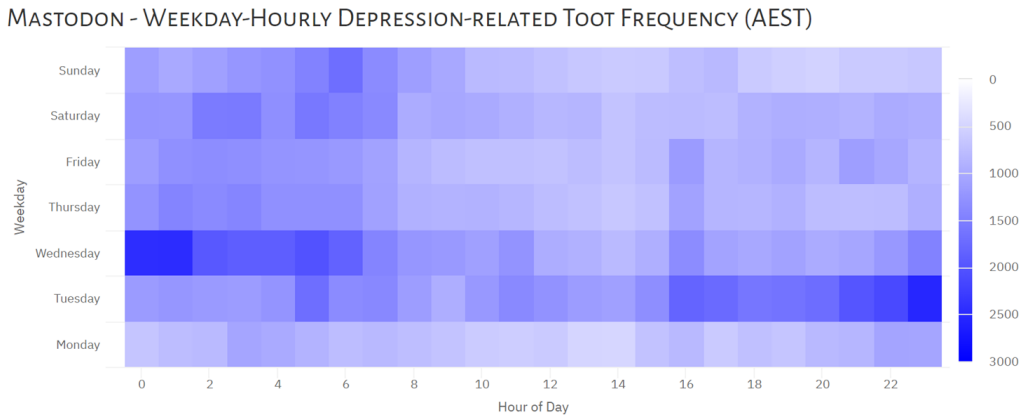

Figure 16: Weekday-Hourly Drepression-Related Toot Frequency

most opposite trend in hour-of-day and shows a large volume of depression-related toots on Tuesday and Wednesday instead. The difference in hour-of-day can be explained by Mastodon's user profile with most users based in the Northern Hemisphere. The spike in volume on Tuesday and Wednesday may be related to stress levels. Life in many countries in the Northern Hemisphere with a high percentage of Mastodon users such as the US and the UK tends to be more fast-paced compared to Australia, and Tuesday and Wednesday are in the middle of the work week when stress and workload might peak, leading to increased posts about depression.

6.4 Scenario 3 - Russo-Ukraine War

Russia Invaded Ukraine in February 2022, facing international condemnation and increased sanctions. This war has led to heated media debate and controversy on social media. In this scenario, we aim to understand how Australian Twitter users are posting as compared to international Mastodon users on this matter.

Figure 17: Summary Statistics of Russo-Ukraine War Related Tweet/Toot

In Figure 17 we see that the percentage of Russo-Ukraine War related toots is more than double that of Russo-Ukraine War related tweets, this is likely caused by the amount of US Mastodon users that have heightened interest in this topic due to the US's involvement in this matter. Similar to previous scenarios, increased privacy will allow Mastodon users

to share their opinions on possibly controversial matters more openly. Next, Figure 18 illustrates the proportion of Russo-Ukraine War tweets and toots made in English versus other languages against the linguistic landscape of Australia as a whole. We can conclude that English tweets seem to dominate on this topic although a fair share of Australians speak a language other than English, while relevant toots are more diverse in the language used.

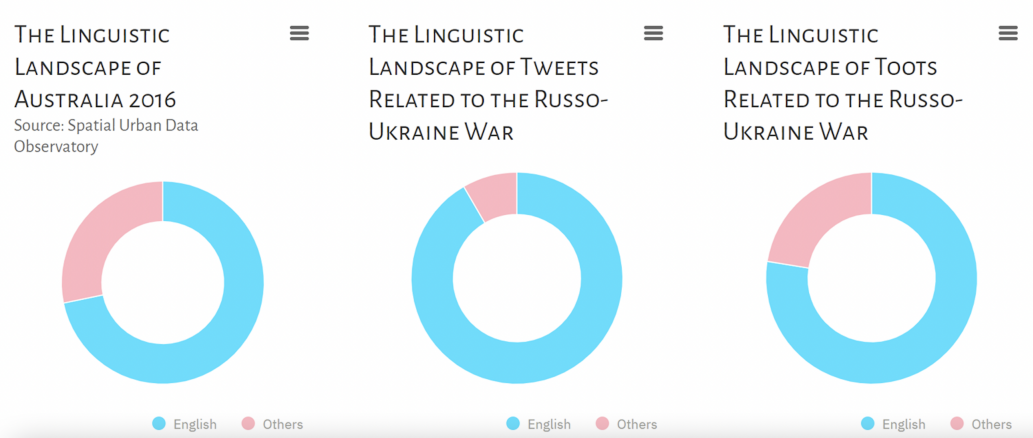

Figure 18: The Linguistic Landscape of Australia, Tweets and Tweets

Figure 19 analyses the volume of Russo-Ukraine War related tweets from February 2022 to August 2022. The peak in March logically corresponds to the spike in media attention after Russia invaded Ukraine on the 22nd of February 2022, which gradually died down after the end of March.

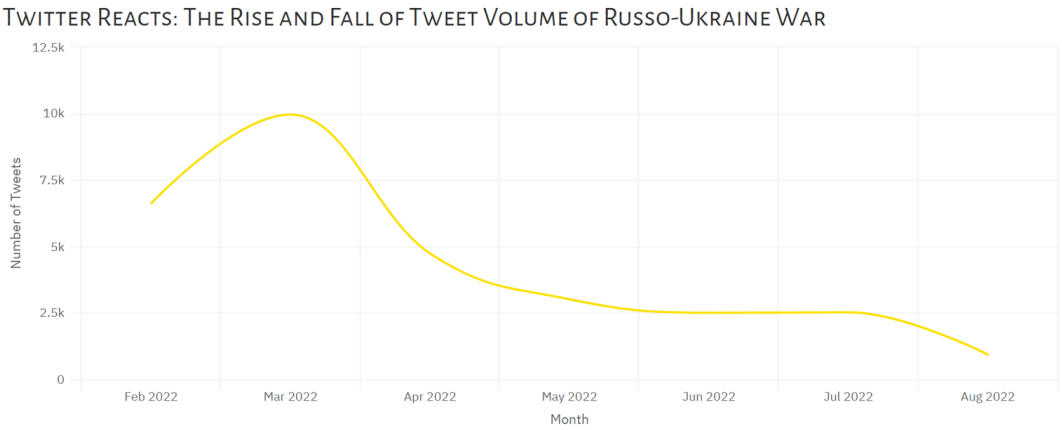

Figure 19: Volume of Russo-Ukraine War Related Tweets

7 Limitations

As some methods are overly computationally expensive or require a prolonged period of time to implement, we faced some limitations as outlined below:

- **Mastodon Timeline**: Ideally we would harvest legacy data all the way back to February 2022 to overlap with our Twitter timeline and make trend comparisons with respect to time. However, given Mastodon's rate limit, it would take months to crawl enough data to overlap with Twitter's timeline.
- **Tweet/Toot Classification**: Instead of using keyword dictionaries like our current implementation, we experimented with using a pre-trained language model such as **BERT** [12] to classify posts into one or more of our scenarios. We found that this multiclass classification process takes around 10-15 seconds for each post. While the accuracy of tweet/toot classification will increase using a computationally expensive model like **BERT**, this was simply not feasible for the volume of data we have, and we would require more resource allocation on **MRC** to implement this.
- **Demographic Data:** The lack of more recent data from the **SUDO** platform is a limitation in our analysis. To maintain consistency across analysis, we have employed **SUDO** data exclusively from the year 2016.

8 Conclusion

Our cloud-based social media analysis explored three scenarios, two of which are social aspects that influence the way both Australian and global users post (Religion and Depression), and the third scenario is a controversial political matter that has sparked intense media debate over the past year. On a broader level, our results demonstrate a positive relationship between population size and tweet frequency, a prevalence of positively connoted terms appearing in Australian tweets, and Mastodon users being more comfortable with their expression in all three scenarios. Due to the live nature of how Mastodon data is utilized in this project, our results are not fully replicable and are subject to current trends on the two Mastodon servers as of the date of implementation. For future work, the use of more advanced language models to perform post classification and sentiment analysis are areas to potentially focus on. It would also be insightful to compare Mastodon and Twitter posts over an overlapping period of time given the availability of data.

References

[1] "Melbourne Research Cloud Documentation." Melbourne Research Cloud Documentation. https://docs.cloud.unimelb.edu.au/ (accessed May 25, 2023).

[2] "Overview — Apache CouchDB® 3.3 Documentation." Overview — Apache CouchDB® 3.3 Documentation. https://docs.couchdb.org/en/stable/ (accessed May 25, 2023).

[3] "shiny package — RDocumentation." Home — RDocumentation. https://www.rdocumentation.org/packages/shiny/versions/1.7.4 (accessed May 25, 2023).

[4] "Ansible Documentation." Ansible Documentation. https://docs.ansible.com/ (accessed May 25, 2023).

[5] "MPI for Python — MPI for Python 3.1.4 documentation." MPI for Python — MPI for Python 3.1.4 documentation. https://mpi4py.readthedocs.io/en/stable/ (accessed May 25, 2023).

[6] "Greater Capital City Statistical Areas." Australian Bureau of Statistics. https://www.abs.gov.au/statistics/standards/australian-statistical-geography-standard-asgs-edition-3/jul2021-jun2026/main-structure-and-greater-capital-city-statistical-areas/greater-capital-city-statistical-areas (accessed May 25, 2023).

[7] "Introducing the Spatial Urban Data Observatory." eResearch Australasia Conference. https://conference.eresearch.edu.au/2022/08/introducing-the-spatial-urban-data-observatory/ (accessed May 25, 2023).

[8] "Mastodon.py — Mastodon.py 1.8.1 documentation." Mastodon.py — Mastodon.py 1.8.1 documentation. https://mastodonpy.readthedocs.io/en/stable/index.html (accessed May 25, 2023).

[9] "Mastodon.social — Geography & Country Targeting." Similarweb. https://www.similarweb.com/website/mastodon.social/#geography (accessed May 25, 2023).

[10] "Mastodon.world — Geography & Country Targeting." Similarweb. https://www.similarweb.com/website/mastodon.world/#geography (accessed May 25, 2023).

[11] "Mental health." Australian Bureau of Statistics. https://www.abs.gov.au/statistics/health/mental-health (accessed May 25, 2023).

[12] ”Models - Hugging Face.” Hugging Face – The AI community building the future. https://huggingface.co/models (accessed May 25, 2023).