COMP90024
Cluster and Cloud Computing
Assignment 2

Team 6

Bohan Su, 1174197

Tianhao Liu, 1105160

Xiangmin Zhou, 1313799

Wenwen Zhang, 1174470

Junbo Hu, 1038361

May 22, 2023

GitHub: https://github.com/Wen20011009/CCC-ass2-team6.git

YouTube: https://youtu.be/EBhZff-8_gI

Abstract:

This report presents the system and implementation architecture for the University of Melbourne's COMP90024 Cluster and Cloud Computing course for Assignment 2. The scenarios chosen for the main topic of the report relate to the level of education of each Greater Capital City in Australia and how it relates to user comments in Twitter.

The report includes the utilisation and discussion in depth of each individual system component and an overview of the division of labour and collaboration of the team during the process of assignment.

Table of Contents

1. Introduction

Education has always been a priority in Australia, and Australian education is widely recognized internationally as being of high quality and highly valued. In addition, the Australian government invests a lot of resources and money in education and many schools offer bilingual or other language courses. Therefore, given the growing importance of social media such as Twitter in recent years and in social discussions, this report will discuss and analyse the relationships between the level of education in each Greater Capital City in Australia and the information of tweets in Twitter.

In this report, the architecture deployment will be shown firstly. Then the workload of data collection and how to store it in CouchDB will be also discussed and given. Moreover, the information from SUDO, historical Twitter data and other resources will be further analysed from three directions. In addition, the next section introduces the web application. Furthermore, section 6 describes the easy-to-use steps of the overall system architecture, while Section 7 outlines what each team member was responsible for during the development process as well as their individual contributions. Section 8 summarises the overall report.

2. Architecture

2.1 Pipeline & Allocation of Resource

2.1.1 Pipeline

- ❖ Data source and uploading
    - ❖ Four basic data sources, including historical tweets data, ABS official data, SUDO official data, Data from GitHub Source.
    - ❖ Apply basic python search method, choosing the related tweet (Crime, Profanity etc)
    - ❖ Use mpi4py library to upload the data to CouchDB in different nodes.
- ❖ Data processing
    - ❖ Write the design document in CouchDB, do all the data analysis step through MapReduce, process the tweet data into key-value pairs.
- ❖ Data visualisation
    - ❖ Collect the output generated by MapReduce, utilise Tableau or relevant Python libraries for data visualisation, and transmit the results to our backends for integration into the website.

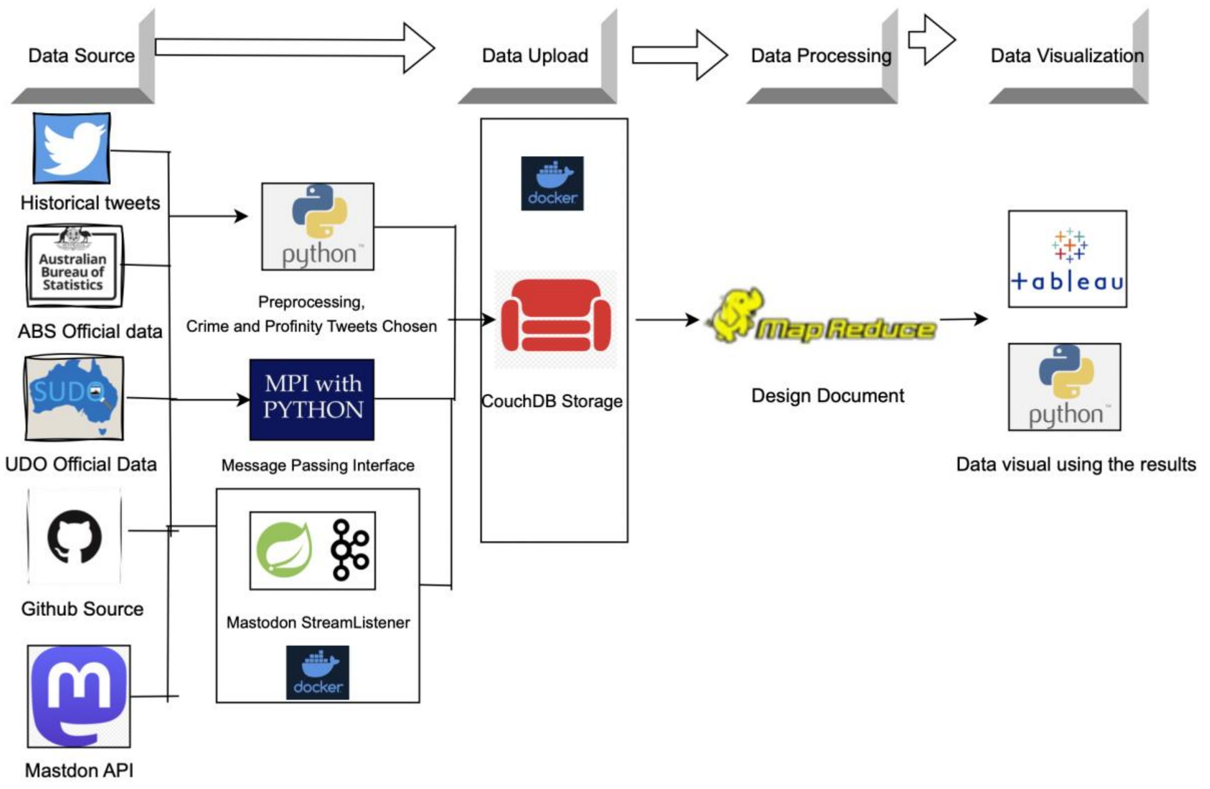


Fig 1: Basic Pipeline

2.1.2 Allocation of resources

In the initial stages of deployment, our plan is to deploy 4 instances on the Melbourne Research Cloud (MRC). We leverage Ansible for this task, as it automates the instance setup and configuration modifications, avoiding the need for manual intervention. For basic configuration settings, please refer to Figure 2.

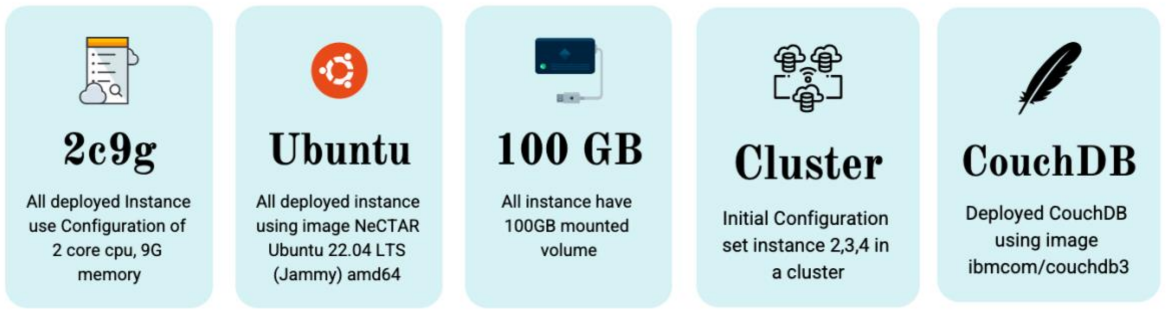

2c9g
All deployed Instance use Configuration of 2 core cpu, 9G memory
Ubuntu
All deployed instance using image NeCTAR Ubuntu 22.04 LTS (Jammy) amd64
100 GB
All instance have 100GB mounted volume
Cluster
Initial Configuration set instance 2,3,4 in a cluster
CouchDB
Deployed CouchDB using image ibmcom/couchdb3

Fig 2: Initial Configuration

Beyond the fundamental setup, we've enabled necessary security permissions for these instances. We've allowed access from all sources for both IPv4 (0.0.0.0/0) and IPv6 (::/0) on specific ports. These include the standard HTTP, and SSH ports, as well as some custom application ports such as TCP port 5984 for CouchDB.

To better understand the roles allocated to each instance, the subsequent diagram shows the responsibility assigned to each one.

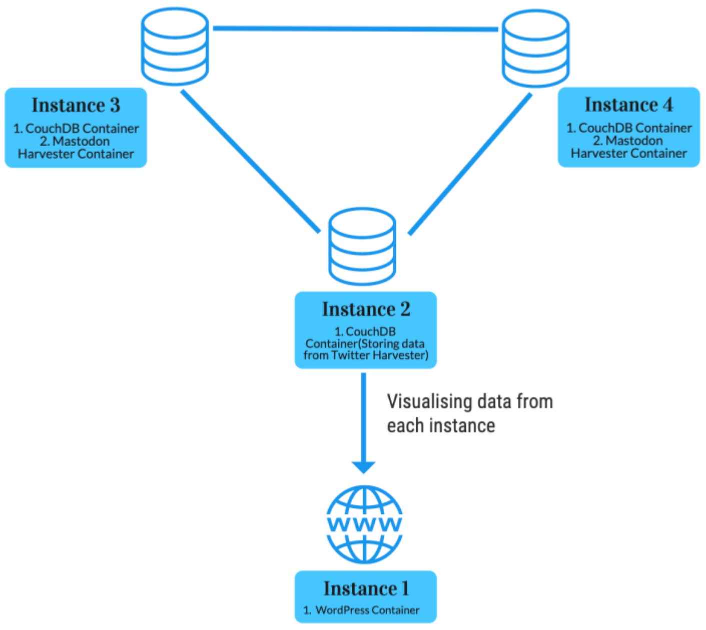

Instance 3
1. CouchDB Container
2. Mastodon Harvester Container
Instance 4
1. CouchDB Container
2. Mastodon Harvester Container
Instance 2
1. CouchDB Container(Storing data from Twitter Harvester)
Visualising data from each instance
WWW
Instance 1
1. WordPress Container

Fig 3: Instance Responsibility

In addition to the architecture illustrated above, further explanations are necessary for certain aspects. Specifically, the harvester container on instance 3 is designed to store Mastodon data in the CouchDB of the same instance. Likewise, the harvester container on instance 4 is expected to store Mastodon data in the CouchDB of instance 4.

On the other hand, CouchDB on instance 2 is dedicated to storing data from Twitter, which is uploaded from a local device. Lastly, the WordPress container deployed on instance 1 is responsible for visualising the data gathered from CouchDB on instances 2, 3, and 4.

2.2 Deployment (MRC, CouchDB, Twitter Harvester, Web App)

The approach that our team used for deploying instances inside Melbourne research cloud, CouchDB, Twitter Harvester and Web app through ansible. Ansible provides a simple and powerful way to automate tasks such as provisioning, deployment, orchestration of applications and infrastructure.

In the deployment process, our team used a tool called docker-compose, which is a skill in which we can operate several containers in a single file. Docker Compose uses yml file in its configuration files to define how Docker containers should behave.

2.2.1 Deployment of MRC, CouchDB

The deployment of our instances inside the Melbourne research cloud, which includes set up nodes in a cluster, and creation of our CouchDB in each node, also include downloading docker and some necessary python library inside each virtual machine (Ubuntu) by using ansible and docker.

There are seven key playbooks that were used in the automation of the deployment process and one important sh file which our group needs to authenticate against the identity.

- **Group6.sh**: This is an important sh file when using Ansible to ensure the correct authentication and environment variable settings, to communicate the operation with the OpenStack cloud environment. Which include defining the project name and symbols, the username and password of the operators, and defining the interface as public or private.
- **Config:** This playbook serves as the foundation for further configuration. It defines the essential settings for Ansible, such as the Python version, the target environment (Ubuntu in our project), and the private key required for connecting to the Ubuntu virtual machine. It also specifies the working directory and sets the username and password for CouchDB. Additionally, it sets up a cluster, defines the CouchDB name, port number, and the IP address of the local host.
- **Deploy:** This is a basic deployment playbook which helps to support the further deployment. It defines all kinds of rules when using the playbook, including our team can only use the internal website of unimelb to do all the deployment, also the instance key name, image and its size which is 2 core 9 G size for one node. Additionally, the security rule our team has created. (Which include the port number of which each ip address connect to)
- **Instances:** This basic instance playbook serves as a fundamental configuration framework for subsequent tasks. It encompasses the definition of security groups for each instance, the allocation of volumes to individual instances, and the specification of unique names for each instance. It lays the groundwork for further customization and fine-tuning of the instances within the environment.

After building the fundamental playbooks, all the playbooks below will access the information inside and do its own jobs.

- **Deploy Instances:** This playbook accesses the information from deploy and instances, and the role of the playbook is to create the columns and instances in Melbourne research cloud.

- **Config Instances:** This playbook accesses the information from playbook config,deploy, instances, and the role of the playbook is to install all the needed libraries in our virtual machine and also set up docker for each instance.
- **Deploy Db and Crawler:** This playbook accesses the information from playbook config, and the role of the playbook is to create the CouchDB inside each node.
- **Remove Deploy Instances:** This playbook is used when something went wrong in the process when setting up the node, as for our group wanting to add one more node or our group defining the wrong master node and so on. It accesses the information inside the deploy playbook and its role is to remove all the instances, volumes and security groups. (It is not recommended to use, or you will lose all your valued data)

2.2.2 Pro and Cons of Melbourne Research Cloud

- **Pros**
    - ❖ Designed for Scenarios: As for our group has different data source, some are static data another is dynamic streaming data, due to the large amount of data size, it's better to make them into a cluster, which MRC cloud provide a chance to create different nodes and make them a cluster, which our team could assign different nodes storing different data, and also use a node as master node to do its own job such as web application etc…
    - ❖ Network Security: As for Melbourne Research Cloud is from Melbourne university its own, which could provide a good network security, as for everytime when someone wants to access the MRC, they have to be confirmed as part of the unimelb students, preventing hackers from infiltrating and stealing our data.
    - ❖ Collaboration Opportunities: As our team is part of the university of Melbourne, MRC could provide more opportunities to collaborate with other researchers at the university.
- **Cons**
    - ❖ Inside Melbourne Research Cloud, there is a limitation of the number of nodes (8 maximum) and the maximum volume for each node (500 G) and also a limited number of security group rules (150 max), if there is a more complex mission as for our group need more than these number of nodes, it will be not enough for MRC.

- ❖ MRC sometimes provide potentially lower performance, such as the speed and limits erc… Some other clouds, such as AWS and Azure may have a better performance than MRC.
- ❖ MRC might provide less service than the other cloud service, the implementation in MRC would become harder if the requirement is getting more complex. In other could service, some real-time processing service or automated ETL jobs might reduce the workload of whole process.

2.2.3 Mastodon Harvester Deployment with Ansible and Docker

The deployment of our Twitter Harvester application was streamlined and automated using Ansible and Docker. The Twitter Harvester application was built and run using Docker. Each Docker container, functioning as two separate nodes on 2 different instances, each node would be responsible for a Mastodon Server, which provides a lightweight, self-sufficient environment for the application. This aligns perfectly with Ansible's approach of controlling and setting up systems without the need for a main, central server. This makes the process of setting up our application both effective and easy to handle.

There are four key playbooks that were used in the automation of the deployment process:

- **Deploy Python Script in a Docker Container:** This playbook is responsible for the initial deployment of the Twitter Harvester. It involves copying the Python script to the remote instance, creating a Docker file, building the Docker image, and finally running the Docker container.
- **Stop Docker Container:** This playbook is used when there's a need to halt the operations of the Twitter Harvester without discarding the container. This is useful when maintenance or upgrades are required.
- **Remove Docker Container:** This playbook is utilised when the Docker container running the Twitter Harvester is to be completely removed. This could be due to various reasons like moving to a new instance or discarding the old container.
- **Restart Docker Container:** This playbook restarts the Docker container running the Twitter Harvester. This can be necessary after a system failure or after updates have been made to the script or its dependencies.

After the deployment of the python Script, the stream data from the mastodon server could be kept uploaded to our CouchDB database, and the preparation of the other three notebooks could enable us to easily interrupt the container if there are any further changes.

2.2.4 Front End deployment with Ansible and Docker

We streamlined the deployment of our WordPress application using Ansible and Docker, encapsulated in a single playbook which includes the function described as follows:

- Deploy the WordPress application: This playbook automated the process of setting up the necessary environment, including pulling the required Docker images and launching the Docker containers for WordPress and MySQL. It ensured the correct configuration of environment variables for both WordPress and MySQL containers, managed the persistence of MySQL data through Docker volumes, and properly linked the WordPress container to the MySQL container for smooth interaction. Additionally, the playbook handled the mapping of the WordPress container's port 80 to the host, making the WordPress site accessible.

This deployment process, utilising Infrastructure as Code, enables us to Store the code (Ansible playbook) in a version control system, like Git, which keeps a history of all changes made to the playbook. This benefits us if we want to track changes, roll back or Replicate Environments of our frontend configuration.

2.3 Dynamic Scaling

Dynamic Scaling refers to the automatic adjustment of system resources based on demand to accommodate varying workloads, and our initial deployment would consider both aspects in horizontal and vertical scaling.

Firstly, in our project, we have utilised Ansible for the automated deployment of CouchDB in the cloud environment, including instances. The playbook contains specifications for allocating mounted volume sizes to each instance. As a result, by modifying the playbook, we can conveniently adjust system resources at any time to adapt to changes in workload requirements to achieve vertical scaling.

Secondly, we have implemented a shell script for streaming data processing that checks the disk usage. If the disk usage reaches 90%, the system will automatically adjust the database resources. During periods of high load, it will close the container responsible for data crawling to reduce resource consumption and save costs. This dynamic adjustment ensures efficient resource utilisation and cost optimization in response to varying workload demands.

Using the dynamic scaling skills, it can enhance system elasticity and scalability, enabling the system to adapt to constantly changing workload demands. Reducing resource waste by allocating and releasing resources based on actual needs, thereby minimising costs and energy consumption.

Thirdly, CouchDB natively supports shading, where the data in a database is partitioned across multiple nodes in a cluster. A database can be shared into any number of shards, and those shards can be distributed among any number of nodes. Which could balance the workload among multiple nodes when the loads are high, performing a dynamic horizontal scaling in the cloud.

However, we are encountering some connection issues between each node in the cluster. Therefore, some of the functionalities are not performed in expectation.

2.4 Errors Handling

We have implemented a backup system for our CouchDB, where we store our stream data. The need for backing up these databases arises from their continuous receipt of stream data from the Mastodon server. Given the constant influx of data, the risk of system crashes is high. Every time a new piece of data is written into the system, it triggers changes or continuous operations in the view, utilising MapReduce.

The database that stores data from "aus.social" is backed up in real time. However, for the database storing data from "mastodon.social", we've opted for one-time backup updates. The rationale behind this is that "mastodon.social" is the largest forum we are engaged with. Its data volume is immense, primarily because this server contains the most frequently new posted content among all Mastodon servers. Continuous real-time backups would significantly increase the server load, and we believe that the over one million pieces of data we currently have is sufficient for our analysis. At last, all our backup jobs are finalised by the built-in replication function in CouchDB.

The deployment of the disk monitoring shell script could also be considered as an error handling method. This script is set up as a routine Cron job on these VMs, checking the disk usage every ten minutes. If the disk usage hits the 90% mark, the script automatically stops the data streaming container to prevent it from overloading.

Moreover, we have prepared several different shell scripts that run an Ansible playbook. The script can stop, removing, and restarting our deployed containers, which is aimed to tackle any upcoming requirements and avoid extra work of logging into the virtual machines.

2.5 Discussion and Unsolved Issues

After setting up all the clusters, creating instances, creating databases, and creating docker, there are still some unsolved issues which can make the cluster working better.

Some of the current issues as follows:

- Although we have set up a cluster, it appears that nodes can only have simple communication between each other but not collaborate in its best way. As a result, our four single nodes may not be efficient enough, and handling catastrophic events becomes challenging as replication is the only option across different nodes. With each node being listed in the cluster, they do not recognize each other, demonstrating a lack of inter-node awareness. Even numerous attempted solutions from Ed discussions or online resources, have yet to resolve this persistent issue. Therefore, some functionality we expected to have in deployment might not work as expected.
- With single nodes, the database may crush when a large amount of data is being used to read and write, which then multiple nodes can share the load of read and write operations, alleviating the pressure
- Ideally, our group plan to use main as the manager of the four nodes our group created, which main node is to manage the usage of each node, which then if one of the node got a really large amount of workload which the node cannot handle with its volume, then main node will assign some of the task from the high usage node into low usage node to keep the cluster in a balanced state.
- There are several unsolved issues in terms of security that could be addressed. Here's a list and justification of those security issues, that we have not solved due to time constraints and limitation of knowledge in web security among all group member:
    - ❖ **Security Group Configuration:** our security groups have numerous rules allowing open access from any IP address (0.0.0.0/0 in IPv4 and ::/0 in IPv6). This presents a significant security risk as anyone from any IP address could potentially attempt to access the resources protected by these security groups.
    - ❖ **Database Access:** We access the database using a plain HTTP connection when we store the data and create a view by the MapReduce function, which is not secure. The credentials (username and password) are clearly listed in our Python Script in the form http://username:password@instance_ip/database_name, which could be intercepted easily.

    - ❖ **SSH Access to Instances:** The private key for SSH access is stored on each group member's local system. If one system is compromised, access to the instance could be gained by an attacker.
    - ❖ **Inbound Rules for all ports:** The inbound rules allow all types of traffic (ALLOW IPv4 to 0.0.0.0/0 and ALLOW IPv6 to::/0) which is not secure as it exposes the instances to potential threats.

  As the limitation of the security aspect of our system, here are some future developments we could apply:

    - ❖ Only grant necessary IP addresses or ranges that require access, so that only specific IPs that belong to certain organisations could have permission.
    - ❖ CouchDB could only be accessed by HTTPS, because HTTPS encrypts the data sent and received, making it harder to intercept and read.
    - ❖ Using the key management service that is provided in Cloud services.
    - ❖ Limit the inbound rules to only necessary ports that could be accessed.

  After addressing these changes, we believed that the system would have a better security control ability.

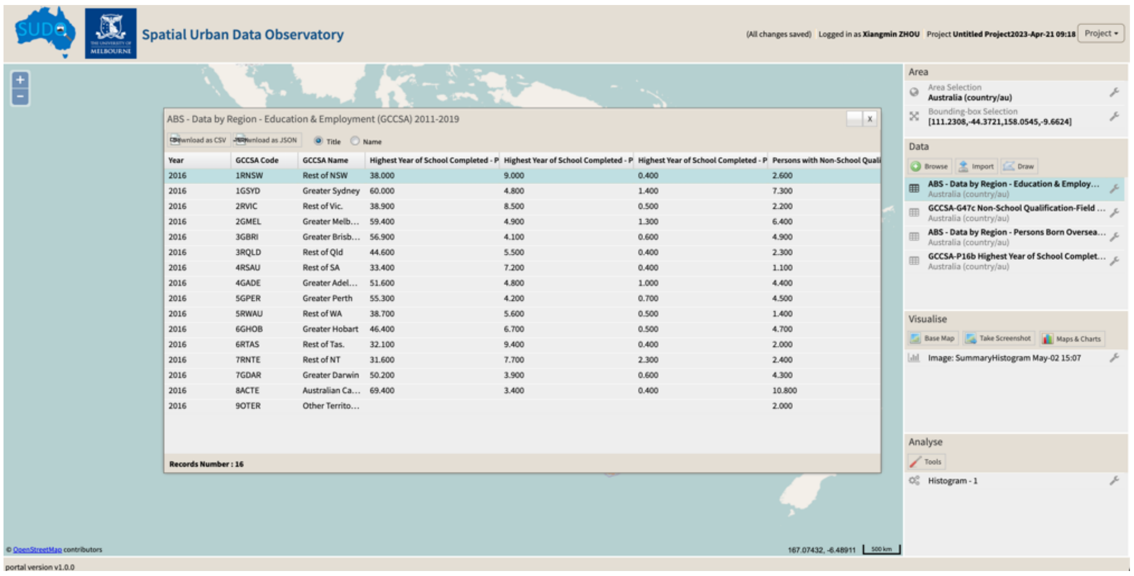
- With some further improvement in dynamic scaling, which our group can add, real-time monitoring of network traffic allows for adjustment of network speeds based on congestion levels when the network is not occupied by tasks from other nodes. If the network traffic is smooth, the speed can be adjusted accordingly to optimise performance.
- Due to the role of our main node is for website deployment, our group first built the website with a domain on other IP addresses. However, when planning to migrate the website, an unknown problem occurred, causing the migration to fail. So, the ip-addresses are still built on the original ip-address.

3. Data Collection and Storage in CouchDB

After the design and deployment of our architecture, the data collection and cleaning should be implemented here and then store the data after cleaning in the predefined CouchDB. Moreover, there are three types of resources of data, which are data from Spatial Urban Data Observatory (SUDO), historical Twitter Data (60G) and the stream data (Mastodon). In this section, the workload we collect the data and the issues we come across during collection will be explained and discussed below.

3.1 Data from Other Resources

3.1.1 SUDO - Education Level Data



Area Selection: Australia (country/au)

Bounding-box Selection: [111.2308,-44.3721,158.0545,-9.6624]

ABS - Data by Region - Education & Employ… Australia (country/au)

GCCSA-G47c Non-School Qualification-Field … Australia (country/au)

ABS - Data by Region - Persons Born Oversea… Australia (country/au)

GCCSA-P16b Highest Year of School Complet… Australia (country/au)

Image: SummaryHistogram May-02 15:07

Histogram - 1

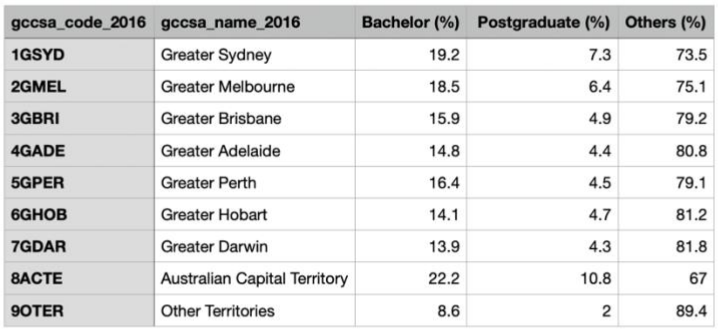
ABS - Data by Region - Education & Employment (GCCSA) 2011-2019

Year	GCCSA Code	GCCSA Name	Highest Year of School Completed - P	Highest Year of School Completed - P	Highest Year of School Completed - P	Persons with Non-School Quali
2016	1RNSW	Rest of NSW	38.000	9.000	0.400	2.600
2016	1GSYD	Greater Sydney	60.000	4.800	1.400	7.300
2016	2RVIC	Rest of Vic.	38.900	8.500	0.500	2.200
2016	2GMEL	Greater Melb…	59.400	4.900	1.300	6.400
2016	3GBRI	Greater Brisb…	56.900	4.100	0.600	4.900
2016	3RQLD	Rest of Qld	44.600	5.500	0.400	2.300
2016	4RSAU	Rest of SA	33.400	7.200	0.400	1.100
2016	4GADE	Greater Adel…	51.600	4.800	1.000	4.400
2016	5GPER	Greater Perth	55.300	4.200	0.700	4.500
2016	5RWAU	Rest of WA	38.700	5.600	0.500	1.400
2016	6GHOB	Greater Hobart	46.400	6.700	0.500	4.700
2016	6RTAS	Rest of Tas.	32.100	9.400	0.400	2.000
2016	7RNTE	Rest of NT	31.600	7.700	2.300	2.400
2016	7GDAR	Greater Darwin	50.200	3.900	0.600	4.300
2016	8ACTE	Australian Ca…	69.400	3.400	0.400	10.800
2016	9OTER	Other Territo…				2.000

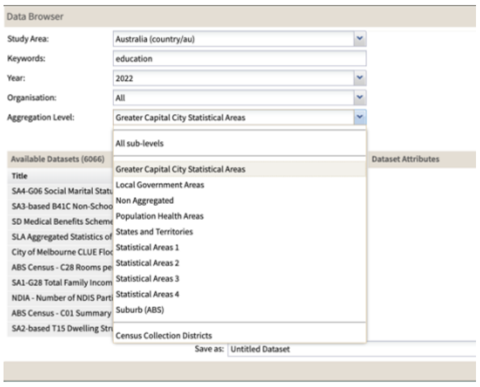
Records Number : 16

Fig 4: SUDO Dashboard

Based on the topic of education, we need to search the data that can reflect the level of education in each region of Australia. Also, the aggregation level should be set as "Greater Capital City Statistical Areas" because we can also get the information in the Twitter dataset based on the greater capital city. However, some attributes and rows for the original datasets are not useful for our further analysis. Hence, we need to remove the rows of data related with the areas except "1GSYD", "2GMEL", "3GBRI", "4GADE", "5GPER", "6GHOB", "7GDAR", "8ACTE" and "9OTER". Also, only attributes of "Bachelor (%)", "Postgraduate (%)" and "Others (%)" should be retained and counted. The table below shows the data after cleaning (csv type).



gccsa_code_2016	gccsa_name_2016	Bachelor (%)	Postgraduate (%)	Others (%)
1GSYD	Greater Sydney	19.2	7.3	73.5
2GMEL	Greater Melbourne	18.5	6.4	75.1
3GBRI	Greater Brisbane	15.9	4.9	79.2
4GADE	Greater Adelaide	14.8	4.4	80.8
5GPER	Greater Perth	16.4	4.5	79.1
6GHOB	Greater Hobart	14.1	4.7	81.2
7GDAR	Greater Darwin	13.9	4.3	81.8
8ACTE	Australian Capital Territory	22.2	10.8	67
9OTER	Other Territories	8.6	2	89.4



Study Area: Australia (country/au)

Keywords: education

Year: 2022

Organisation: All

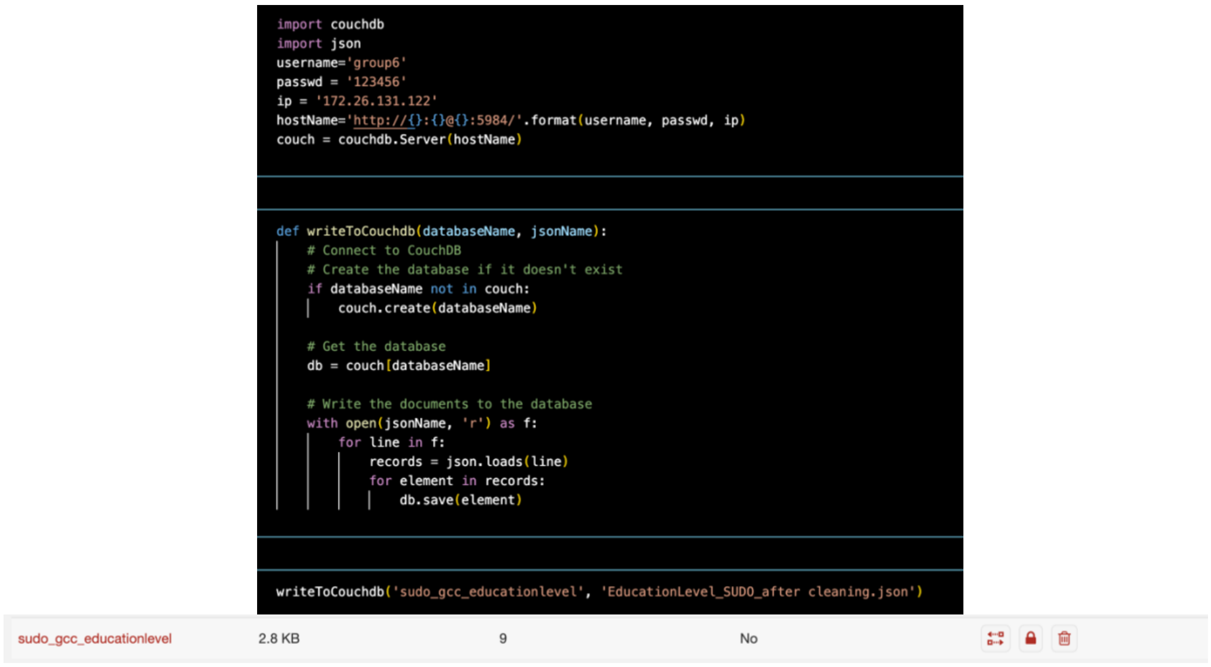
Aggregation Level: Greater Capital City Statistical Areas

Title
SA4-G06 Social Marital Statu
SA3-based B41C Non-Schoo
SD Medical Benefits Scheme
SLA Aggregated Statistics of
City of Melbourne CLUE Floc
ABS Census - C28 Rooms pe
SA1-G28 Total Family Incom
NDIA - Number of NDIS Parti
ABS Census - C01 Summary
SA2-based T15 Dwelling Str

Save as: Untitled Dataset

Fig 5: SUDO Data after Cleaning

Then, under the coding below, the dataset is uploaded and stored successfully.



```
import couchdb
import json
username='group6'
passwd = '123456'
ip = '172.26.131.122'
hostName='http://{}:{}@{}:5984/'.format(username, passwd, ip)
couch = couchdb.Server(hostName)
```

```
def writeToCouchdb(databaseName, jsonName):
    # Connect to CouchDB
    # Create the database if it doesn't exist
    if databaseName not in couch:
        couch.create(databaseName)

    # Get the database
    db = couch[databaseName]

    # Write the documents to the database
    with open(jsonName, 'r') as f:
        for line in f:
            records = json.loads(line)
            for element in records:
                db.save(element)
```

```
writeToCouchdb('sudo_gcc_educationlevel', 'EducationLevel_SUDO_after cleaning.json')
```

sudo_gcc_educationlevel	2.8 KB	9	No

Fig 6: MapReduce and Store in CouchDB

3.1.2 ABS - Crime Data

We obtained the data for our project directly from the Australian Bureau of Statistics (ABS) website [3]. The specific data set we used is found in the "Recorded Crime - Offenders" section, which provides detailed statistics on crime and justice in Australia. The part ‘Principal Offence Type’, which provides valuable insights into crime rates and offender characteristics in Australia, the graph was visualised by a line chart, and we download the CSV which is used to visualise the graph.

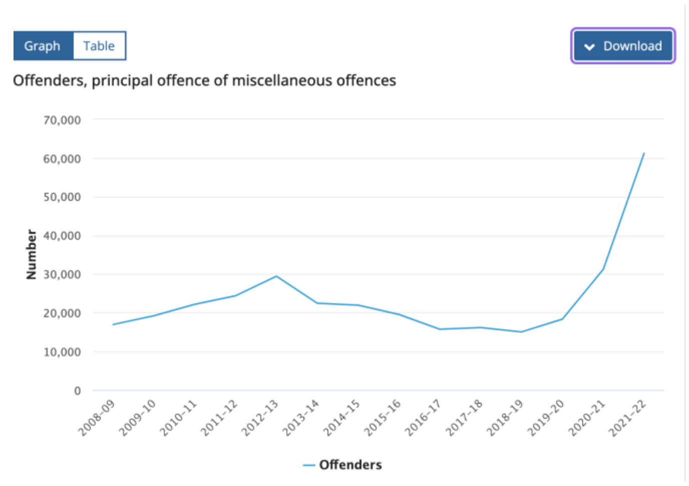


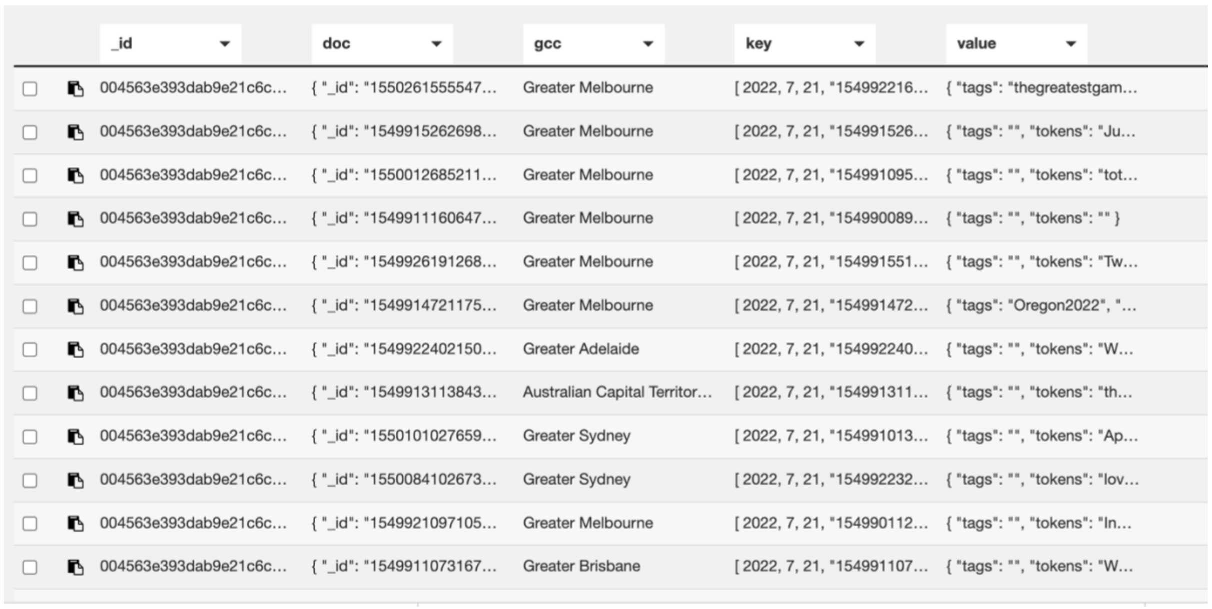
Fig 7: Crime Trends [3]

3.2 Historical Twitter Data (60G)

The historical twitter data consist of json formatting which is already being processed in some stages. Including some of its important features, it's tokens, geolocations, sentiment which our team will use those features in our further study.

Due to the size of the historical Twitter data given by the coordinator is around 60G, which means we cannot directly open it through a laptop. Hence, the Linux language used here to read the top 10000 tweets in terminal, and we can see the specific structure for the json file. It shows some tweets do not have information of the location ("geo"), which prompts us to get rid of these tweets without geographic information. Therefore, the size of this data set has been reduced to 2.7G approximately. Then we use MPI to upload the 2.7G data to Couchdb.

Further Mapreduce and data analysis based on this dataset will be conducted in the next section



_id	doc	gcc	key	value
004563e393dab9e21c6c...	{ "_id": "1550261555547...	Greater Melbourne	[2022, 7, 21, "154992216...	{ "tags": "thegreatestgam...
004563e393dab9e21c6c...	{ "_id": "1549915262698...	Greater Melbourne	[2022, 7, 21, "154991526...	{ "tags": "", "tokens": "Ju...
004563e393dab9e21c6c...	{ "_id": "1550012685211...	Greater Melbourne	[2022, 7, 21, "154991095...	{ "tags": "", "tokens": "tot...
004563e393dab9e21c6c...	{ "_id": "1549911160647...	Greater Melbourne	[2022, 7, 21, "154990089...	{ "tags": "", "tokens": "" }
004563e393dab9e21c6c...	{ "_id": "1549926191268...	Greater Melbourne	[2022, 7, 21, "154991551...	{ "tags": "", "tokens": "Tw...
004563e393dab9e21c6c...	{ "_id": "1549914721175...	Greater Melbourne	[2022, 7, 21, "154991472...	{ "tags": "Oregon2022", "...
004563e393dab9e21c6c...	{ "_id": "1549922402150...	Greater Adelaide	[2022, 7, 21, "154992240...	{ "tags": "", "tokens": "W...
004563e393dab9e21c6c...	{ "_id": "1549913113843...	Australian Capital Territor...	[2022, 7, 21, "154991311...	{ "tags": "", "tokens": "th...
004563e393dab9e21c6c...	{ "_id": "1550101027659...	Greater Sydney	[2022, 7, 21, "154991013...	{ "tags": "", "tokens": "Ap...
004563e393dab9e21c6c...	{ "_id": "1550084102673...	Greater Sydney	[2022, 7, 21, "154992232...	{ "tags": "", "tokens": "lov...
004563e393dab9e21c6c...	{ "_id": "1549921097105...	Greater Melbourne	[2022, 7, 21, "154990112...	{ "tags": "", "tokens": "In...
004563e393dab9e21c6c...	{ "_id": "1549911073167...	Greater Brisbane	[2022, 7, 21, "154991107...	{ "tags": "", "tokens": "W...

Fig 8: Twitter Data in CouchDB

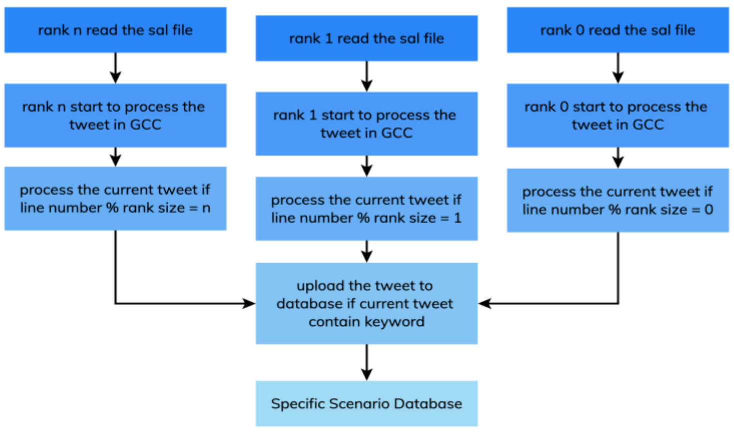
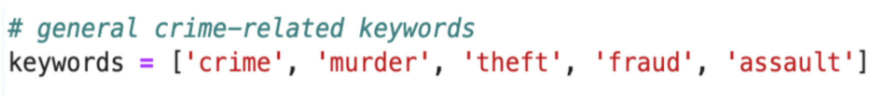

rank n read the sal file
rank 1 read the sal file
rank 0 read the sal file
rank n start to process the tweet in GCC
rank 1 start to process the tweet in GCC
rank 0 start to process the tweet in GCC
process the current tweet if line number % rank size = n
process the current tweet if line number % rank size = 1
process the current tweet if line number % rank size = 0
upload the tweet to database if current tweet contain keyword
Specific Scenario Database

Fig 9: MPI Upload Flow Chart

3.2.1 Pre-process Crime Data

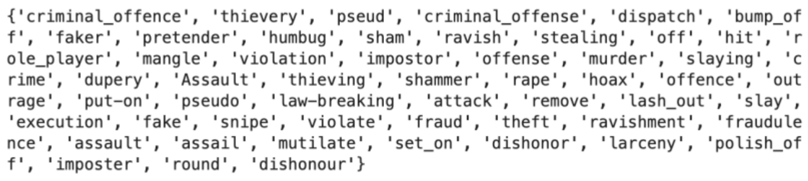
Applying text pre-processing on tweets stored in CouchDB can be quite complex and also increases the load on the CouchDB server. To mitigate these issues, we've chosen to create a separate database that solely contains tweets related to crime keywords. This setup eases the burden on our primary database and streamlines the data processing flow. Before embarking on the actual data storage phase, we initially establish a broad list of crime-related keywords. Subsequently, we utilise the WordNet functionality within the Natural Language Toolkit (NLTK) to expand this list, creating a more comprehensive collection of terms associated with criminal activities. The following is a simplified demo of our keyword selection process:



```
# general crime-related keywords
keywords = ['crime', 'murder', 'theft', 'fraud', 'assault']
```

Fig 10: General crime keyword

And the crime keyword could be expanded to:

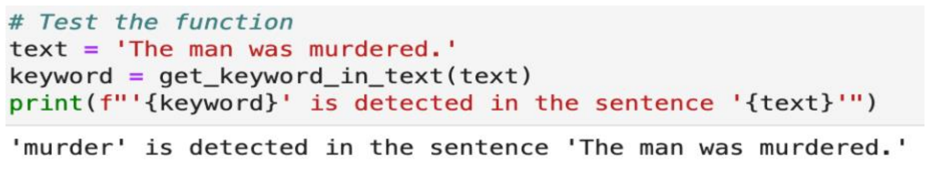


```
{'criminal_offence', 'thievery', 'pseud', 'criminal_offense', 'dispatch', 'bump_of
f', 'faker', 'pretender', 'humbug', 'sham', 'ravish', 'stealing', 'off', 'hit', 'r
ole_player', 'mangle', 'violation', 'impostor', 'offense', 'murder', 'slaying', 'c
rime', 'dupery', 'Assault', 'thieving', 'shammer', 'rape', 'hoax', 'offence', 'out
rage', 'put-on', 'pseudo', 'law-breaking', 'attack', 'remove', 'lash_out', 'slay',
'execution', 'fake', 'snipe', 'violate', 'fraud', 'theft', 'ravishment', 'fraudule
nce', 'assault', 'assail', 'mutilate', 'set_on', 'dishonor', 'larceny', 'polish_of
f', 'imposter', 'round', 'dishonour'}
```

Fig 11: Expanded crime keyword by Wordnet

However, this approach has its limitations. Certain words generated during the expansion might not be pertinent or logical in the context of crime. To tackle this, we manually review the extended list of words, carefully selecting the final keywords associated with crime. Upon pre-defined this crime-related keyword list, we utilise mpi4py to process the data from the GCC. You can refer to section 3.2 for a more detailed overview of the upload process.

Relying solely on direct matches with pre-defined crime keywords may not ensure complete accuracy and may lack flexibility in identifying whether a tweet is truly crime-related. Therefore, we implement the WordNetLemmatizer from the Natural Language Toolkit (NLTK). This tool enables lemmatization on the keywords and text input, allowing the script to identify keywords in varying grammatical forms. This enhancement ensures that a keyword like "murder" can be identified even if it appears as "murdered" or "murdering" in the text. The following figure illustrates a demonstration of this functionality:



```
# Test the function
text = 'The man was murdered.'
keyword = get_keyword_in_text(text)
print(f"'{keyword}' is detected in the sentence '{text}'")
```

```
'murder' is detected in the sentence 'The man was murdered.'
```

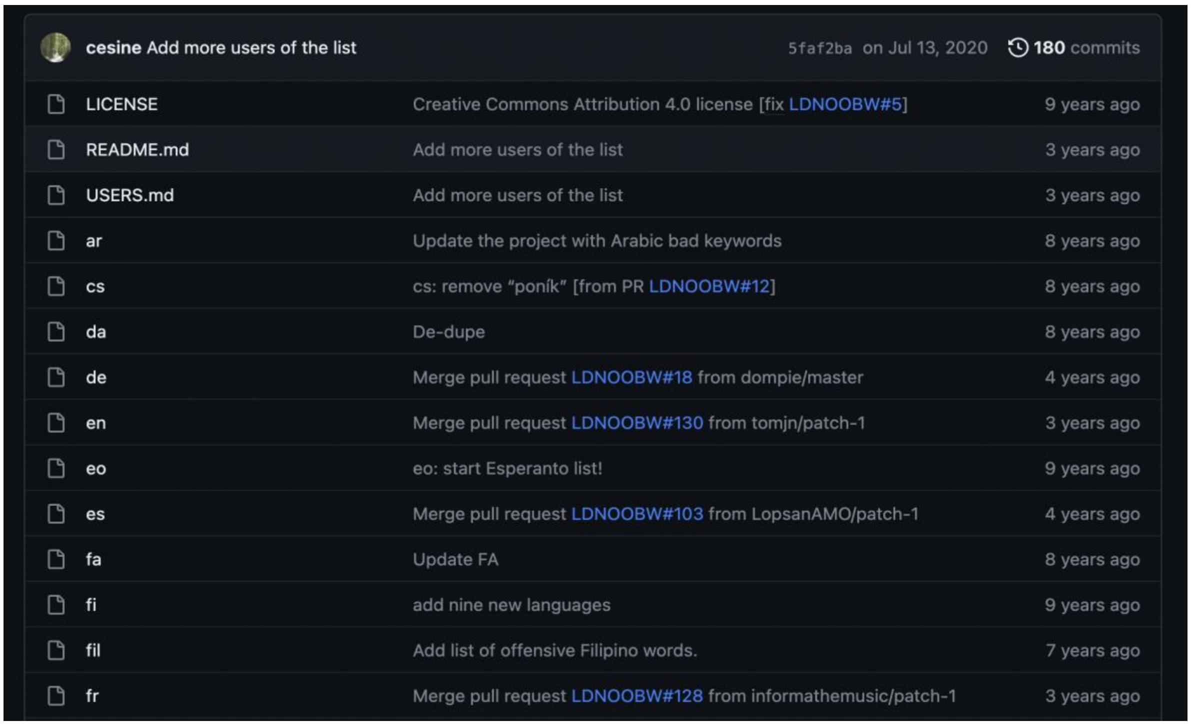
Fig 12: detect keyword in any form

By utilising all the tools described above for keyword detection, we can assure that our system has a satisfactory capability to extract crime-related tweets. Additionally, the system's scalability is ensured by the incorporation of additional processors.

3.2.2 Pre-process Profanity word Data

Due to the size of our tweets data, it is not a good idea to upload the full 60G tweets data into CouchDB, thus our group does some pre-process steps in choosing the Profanity related tweets data.

Inside this project, our group found an open source from GitHub, which includes all profanity words in different languages. Which the formatting of the profanity word shows below.

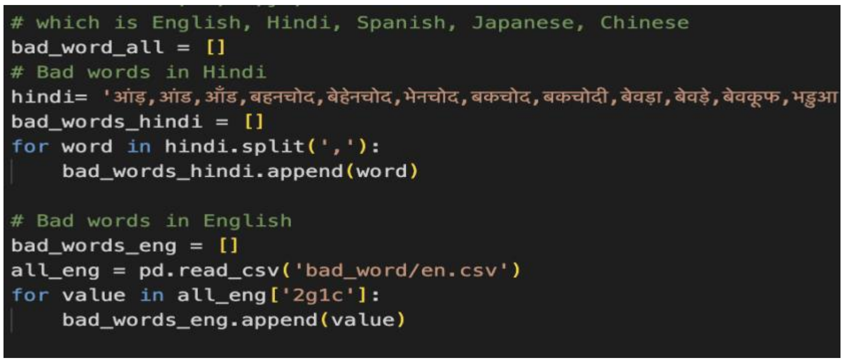


cesine Add more users of the list 5faf2ba on Jul 13, 2020 180 commits

LICENSE	Creative Commons Attribution 4.0 license [fix LDNOOBW#5]	9 years ago
README.md	Add more users of the list	3 years ago
USERS.md	Add more users of the list	3 years ago
ar	Update the project with Arabic bad keywords	8 years ago
cs	cs: remove "poník" [from PR LDNOOBW#12]	8 years ago
da	De-dupe	8 years ago
de	Merge pull request LDNOOBW#18 from dompie/master	4 years ago
en	Merge pull request LDNOOBW#130 from tomjn/patch-1	3 years ago
eo	eo: start Esperanto list!	9 years ago
es	Merge pull request LDNOOBW#103 from LopsanAMO/patch-1	4 years ago
fa	Update FA	8 years ago
fi	add nine new languages	9 years ago
fil	Add list of offensive Filipino words.	7 years ago
fr	Merge pull request LDNOOBW#128 from informathemusic/patch-1	3 years ago

Fig 13: Profanity Word Cloud from Github

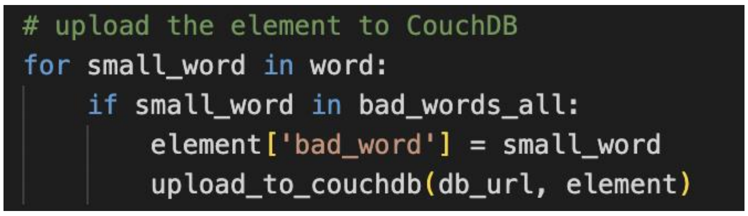
In the process trying to search all the tweets which include profanity words in its token, there is one issue occurred, which is there are so many types of language in github resource, also due to the reason we cannot taken profanity words using the access token, our group measure the top5 used language in tweets and choose the profanity words in those five languages. The method for getting the profanity words is shown below.



```
# which is English, Hindi, Spanish, Japanese, Chinese
bad_word_all = []
# Bad words in Hindi
hindi= 'आंड़,आंड,आँड,बहनचोद,बेहेनचोद,भेनचोद,बकचोद,बकचोदी,बेवड़ा,बेवड़े,बेवकूफ,भड़ुआ
bad_words_hindi = []
for word in hindi.split(','):
    bad_words_hindi.append(word)

# Bad words in English
bad_words_eng = []
all_eng = pd.read_csv('bad_word/en.csv')
for value in all_eng['2g1c']:
    bad_words_eng.append(value)
```

Fig 14: Profanity words from top 5 languages

After selecting all the profanity tweets, in order to help the further analysing of the top profanity words in tweets, our group added an extra feature which is the profanity words used in its token as 'bad_words'.



```
# upload the element to CouchDB
for small_word in word:
    if small_word in bad_words_all:
        element['bad_word'] = small_word
        upload_to_couchdb(db_url, element)
```

Fig 15: Matching Profanity tweets and uploading into CouchDB

By utilising all the tools described above for keyword detection, we can assure that our system has a satisfactory capability to extract profanity-related tweets.

3.3 Stream Data (Mastodon)

Our approach to processing streaming data incorporates several technologies, including nltk, profanity-check, Beautiful Soup, and API requests. These tools enable us to carry out the following stages:

- Account Registration: Initially, we register an account to generate an API key. This key allows us to request real-time streaming data. Having a valid access token is essential to retrieve posts as they are being made by users.
- Data Structuring: We load the received data into a JSON object for easy manipulation. Given the HTML structure of the text data, Beautiful Soup is utilised to extract valuable information. This powerful tool aids in pulling out details such as hashtags, shared links, and user mentions from the post.
- Data Pre-processing: At this stage, we employ the nltk library. While Beautiful Soup helps eliminate non-word information, the text may still contain extraneous data. NLTK assists in filtering out such unnecessary elements, particularly stop words.

- Feature Engineering: To extract relevant information for our specific use case, we create various tags for each post. These include tags indicating whether a post contains crime-related keywords or vulgar language - whether it's in the username, post text, or user's display name. For the detection of vulgar words, we use a pre-trained model from the profanity-check library, offering a fast and efficient solution. This approach ensures code efficiency and reduction of redundancy.
- Data Upload to CouchDB: As discussed in the CouchDB deployment section, we have three CouchDB nodes set up. For storing streaming data, two nodes are designated, with each server corresponding to one node. This distribution ensures ample storage and balanced load across the nodes. We selectively upload relevant data to the database - this includes the tokens extracted from the text, the categorised tags of each post, and some other numerical features. This careful curation ensures clear, structured data, conducive to future analyses. Data is inserted into the pre-existing database using the POST command from the Python requests library.

Upon completion of these steps, our database continually receives streaming data, ready for subsequent analysis.

3.4 Discussion and Comparisons between Official Data (SUDO) and Twitter Data

Overall, there are several differences between the data from the official database (SUDO) and the information from historical Twitter data. First, we can download the data (csv or json) directly from the official websites or database, such as ABS, SUDO etc. Also, the downloaded data can be given clear information after simple data cleaning. However, for the json file of historical Twitter data, it is still needed to do further steps of data processing and sometimes it also needs machine learning algorithms, such as unsupervised learning. Hence, the next section will give more detailed analysis based on the data we get, and we also may come across the problem of inconsistency between the results from different resources.

4. Scenario related Data Analysis

4.1 Mastodon

Since we are working on the historical Twitter data, it would be valuable to first check whether those topics still reflect reality today. As mentioned before, we are going to explore the server ‘aus.social’ and ‘mastodon.social’, so we first explore the language contribution in the ‘aus.social’.

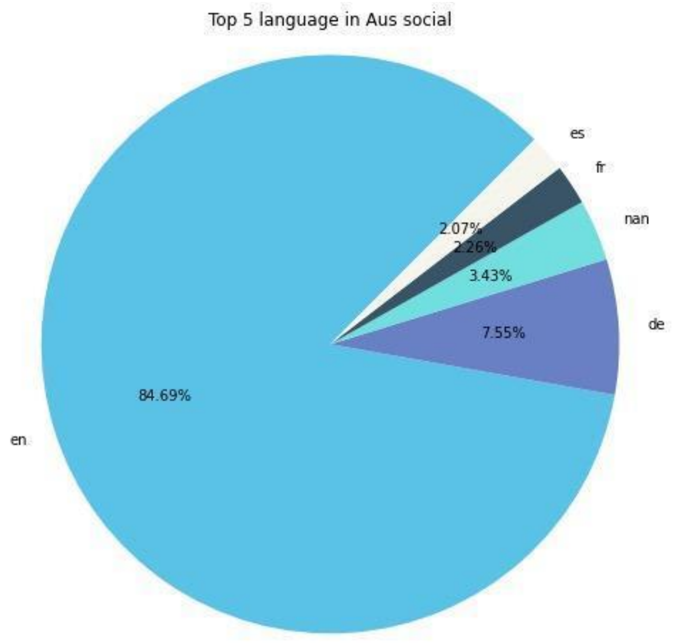


Fig 16: Language Percentage in Australian social server

Following our analysis, it's apparent that English (en) is the prevalent language. A comparison with historical Twitter language distribution may yield valuable insights regarding language usage patterns on these platforms.

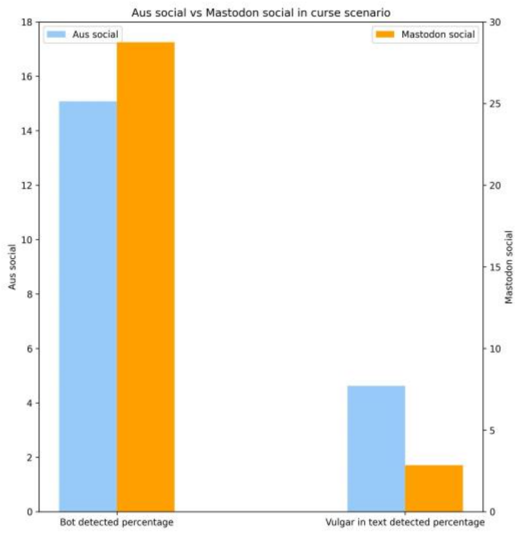


Fig 17: Bot detected Percentage and Vulgar detected Percentage.

Subsequently, we analyse the proportion of tweets containing profanity across the two servers. A bar chart providing a visual representation of this comparison is presented. The chart reveals that 'aus.social' has over double the percentage of profanity tweets compared to 'mastodon.social'.

An interesting attribute that emerged during the stream data processing from Mastodon was the 'bot' feature. We observed a marginally higher percentage of bot-generated tweets on 'mastodon.social', suggesting that a portion of profanity-containing tweets could potentially be automated.

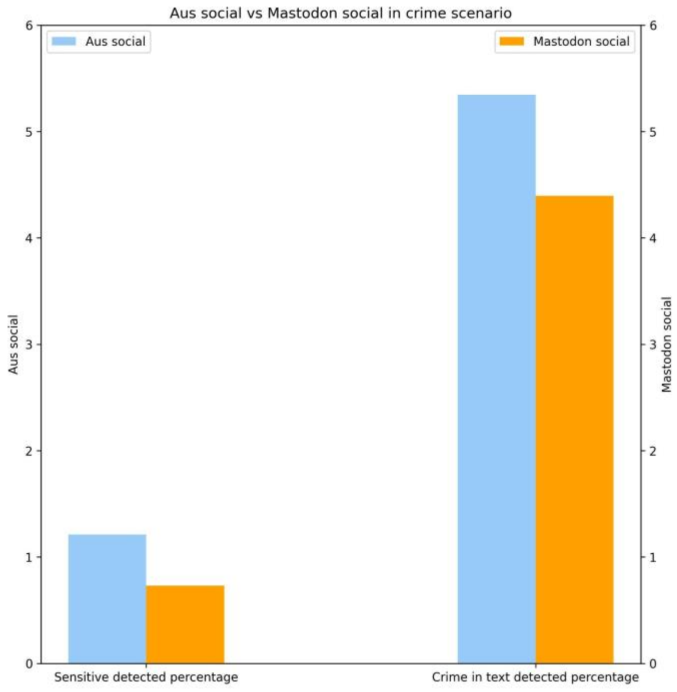


Fig 18: Sensitive Detected Percentage and Crime Detected Percentage

Lastly, we examine the fraction of tweets on Mastodon associated with crime. The following visualisation illustrates that roughly 5.3% of 'aus.social' tweets are crime-related, and slightly higher than that of 'mastodon.social'. Additionally, we incorporate the 'Sensitive' feature in our graph, given its possible correlation with criminal activity. As a result, 'aus.social' again has a slightly higher percentage of sensitive tweets.

In conclusion, the topic discusses crime and tweets containing profanity regularly appear in the tweet today. However, it's worth noting that our analysis has an inherent limitation; the continuous data streaming from Mastodon implies that the results could shift with the introduction of new tweets. The data visualised above was derived when we had accumulated approximately 380,000 tweets from 'aus.social' and 1,300,000 tweets from 'mastodon.social'. As such, the accuracy of our analysis might be affected by the sample size and might not truly reflect the current state of affairs on these servers.

4.2 Scenario One: Education Level vs Profanity Words

In this civilised society, people always have different levels of education, while those with higher levels of education may have more knowledge and culture due to their teaching environments. As a result, they are more likely to possess better communication skills and conflict resolution abilities, as well as stronger social awareness and respect for others. These factors may make them more inclined to engage in civilised and friendly communication with others, while avoiding the use of insulting language.

In this scenario, our group aims to investigate the relationship between education level and profanity words used in their daily tweets.

4.2.1 LDA Model interpretation

For the first stage in this scenario analysis, our group built a model to see the distribution of profanity words based on different topics. This model is called Latent Dirichlet Allocation (LDA). LDA is an unsupervised model, which randomly classifies the data into different numbers of clusters and also returns the most important words in each cluster.

4.2.1.1 LDA Working Basic

Which this model includes following steps:

- Initialization: Assign a random topic to each word in every document.
- Iteration: Repeat the following steps until convergence: Calculate the probabilities of the word belonging to each topic (topic proportions) and the word distribution within each topic. Reassign the topic of the word based on these probabilities.
- Output: After repeating the iteration process, obtain the topic distribution for each document and the word distribution for each topic.

4.2.1.2 Parameter Estimation and Inference Algorithms

To get a deeper understanding on how this model does its parameter estimation and inference algorithms, our group did a variety of research about the method used in the LDA model, and our group listed two commonly used algorithms in the LDA model.

- Gibbs Sampling [1]:

Gibbs Sampling is an iterative algorithm used to estimate the topic distribution and word distribution in the LDA model. In each iteration, it computes the probability of each word belonging to each topic and reassigns each word to a new topic based on these probabilities.

The exact formula used in Gibbs Sampling shows below.

$$\mathrm{P}(\mathrm{z}_i = k|z_{-i}, w) = \frac{C^{w_i}_{-i,k}+\beta}{\sum_{v=1}^{V}(C^{v}_{-i,k}+\beta)} \cdot \frac{C^{k}_{-i,d}+\alpha}{\sum_{j=1}^{K}(C^{j}_{-i,d}+\alpha)}$$

Here, z_i *represents the topic assigned to the* $i-th$ *word,* z_{-i} *represents all the topic assignments except for the* $i-th$ *word,* w *represents the set of words,* $C^{w_i}_{-i,k}$ *represents the count of word* w_i *assigned to topic* k *in all topic assignments except for the* $i-th$ *word,* $C^{k}_{-i,d}$ *represents the count of topic* k *in document* d *excluding the* $i-th$ *word,* V *represents the total number of words,* K *represents the total number of topics, and* α *and* β *are hyperparameters.*

Fig 19: Gibbs Sampling Formula by Latex

- Variational Inference [2]:

Variational Inference is an approximate inference method used to estimate the topic distribution and word distribution in the LDA model. This algorithm approximates the true posterior distribution by maximising the variational lower bound (ELBO).

$$\mathrm{ELBO} = \mathrm{E}_q[logp(w,z)] - E_q[logq(z)]$$

Here, q(z) represents the variational distribution, p(w, z) represents the joint distribution, $\mathrm{E}_q[logp(w,z)]$ *denotes the expected log−likelihood of the observed data and topics, and* $E_q[logq(z)]$ *represents the entropy of the variationaldistribution.*

Please note that the ELBO is a measure of how well the variational distribution approximates the true posterior distribution.

Fig 20: Variational Inference Formula by Latex

4.2.1.3 Model output and Explanation

The model's output is displayed below. On the left side, the LDA model observes ten distinct clusters. Each cluster is assigned based on the word's importance within that group. Group 1 represents the most significant word, while group 10 represents the least significant word. The right side of the results reveals the most important word within each cluster. The formula employed to assign words to their respective groups based on their weighted importance value is presented below.

By seeing the output from our LDA model, there is a clear pattern on the top profanity word used, which top five profanity words are fucking, ass, sexy, paedophile and shit with its overall frequency on the top.

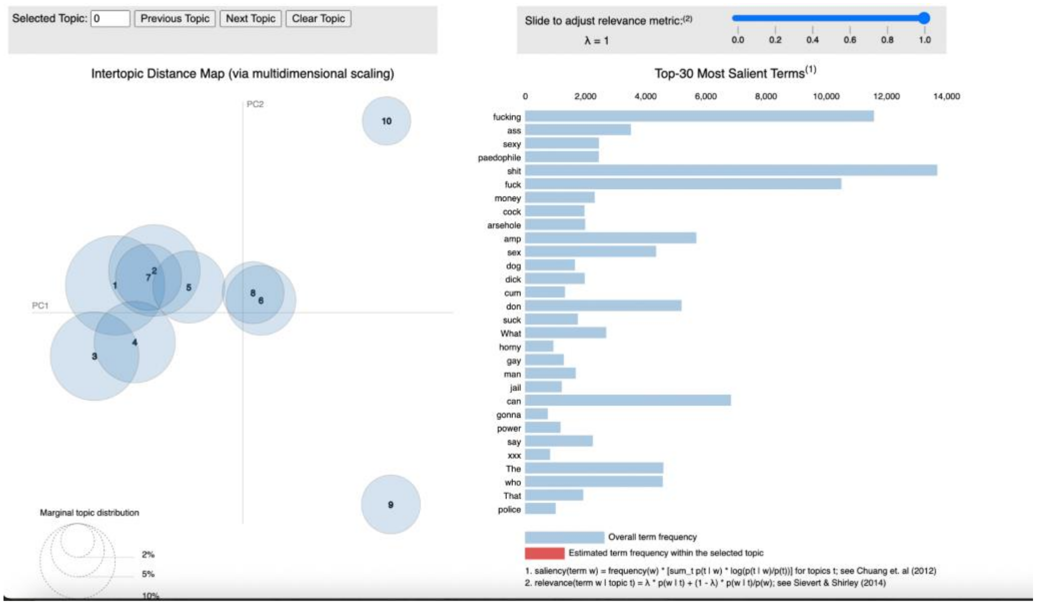
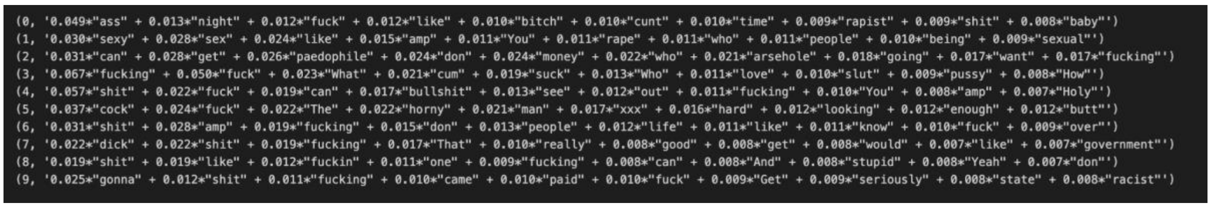


Fig 21: Output of the Latent Dirichlet Allocation model

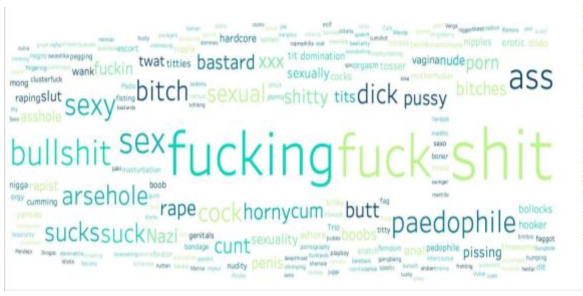


```
(0, '0.049*"ass" + 0.013*"night" + 0.012*"fuck" + 0.012*"like" + 0.010*"bitch" + 0.010*"cunt" + 0.010*"time" + 0.009*"rapist" + 0.009*"shit" + 0.008*"baby"')
(1, '0.030*"sexy" + 0.028*"sex" + 0.024*"like" + 0.015*"amp" + 0.011*"You" + 0.011*"rape" + 0.011*"who" + 0.011*"people" + 0.010*"being" + 0.009*"sexual"')
(2, '0.031*"can" + 0.028*"get" + 0.026*"paedophile" + 0.024*"don" + 0.024*"money" + 0.022*"who" + 0.021*"arsehole" + 0.018*"going" + 0.017*"want" + 0.017*"fucking"')
(3, '0.067*"fucking" + 0.050*"fuck" + 0.023*"What" + 0.021*"cum" + 0.019*"suck" + 0.013*"Who" + 0.011*"love" + 0.010*"slut" + 0.009*"pussy" + 0.008*"How"')
(4, '0.057*"shit" + 0.022*"fuck" + 0.019*"can" + 0.017*"bullshit" + 0.013*"see" + 0.012*"out" + 0.011*"fucking" + 0.010*"You" + 0.008*"amp" + 0.007*"Holy"')
(5, '0.037*"cock" + 0.024*"fuck" + 0.022*"The" + 0.022*"horny" + 0.021*"man" + 0.017*"xxx" + 0.016*"hard" + 0.012*"looking" + 0.012*"enough" + 0.012*"butt"')
(6, '0.031*"shit" + 0.028*"amp" + 0.019*"fucking" + 0.015*"don" + 0.013*"people" + 0.012*"life" + 0.011*"like" + 0.011*"know" + 0.010*"fuck" + 0.009*"over"')
(7, '0.022*"dick" + 0.022*"shit" + 0.019*"fucking" + 0.017*"That" + 0.010*"really" + 0.008*"good" + 0.008*"get" + 0.008*"would" + 0.007*"like" + 0.007*"government"')
(8, '0.019*"shit" + 0.019*"like" + 0.012*"fuckin" + 0.011*"one" + 0.009*"fucking" + 0.008*"can" + 0.008*"And" + 0.008*"stupid" + 0.008*"Yeah" + 0.007*"don"')
(9, '0.025*"gonna" + 0.012*"shit" + 0.011*"fucking" + 0.010*"came" + 0.010*"paid" + 0.010*"fuck" + 0.009*"Get" + 0.009*"seriously" + 0.008*"state" + 0.008*"racist"')
```

Fig 22: The Formula of the importance words in each cluster

4.2.2 General Understanding Profanity Through Word Cloud

Another method our group uses for general understanding of Profanity is using word clouds. Below are the two-word cloud output which came from tableau and python library taken into consideration the words frequency counted, which both of the method shows satisfactory results, and the most profanity word among all the tweets are fucking, sex, shit and so on…



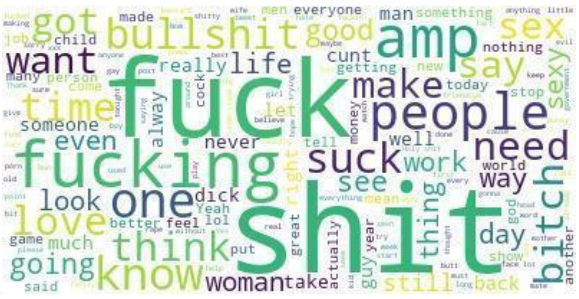
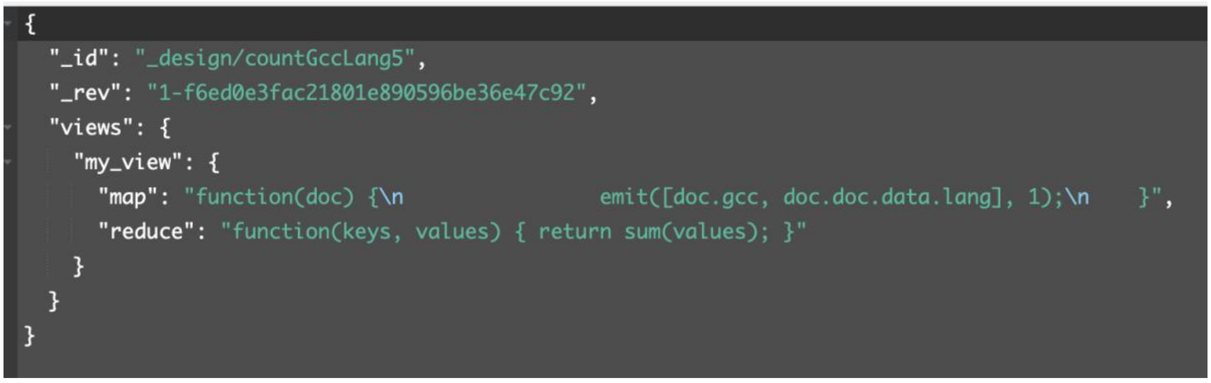


Fig 23: Output of word cloud using tableau *Fig 24: Output of word cloud using python Library*

4.2.3 Relations between Education level and number of Profanity related tweets in each GCC

In this analysis, our team first utilised the MapReduce function to calculate the number of profanity-related tweets in each geographical area (GCC) based on the classification criterion of GCC. We then compared this data with the education level data we obtained. By plotting the key-value pairs, our team generated two bar charts. The left chart represents the number of profanity-related tweets in each GCC, while the right chart represents the corresponding education levels categorised as postgraduate, undergraduate, and below undergraduate.

From the first chart, our team observed that Melbourne and Sydney had the highest number of profanity-related tweets among all the analysed tweets, with 21,002 and 18,910 tweets, respectively. On the other hand, Greater Darwin had the fewest profanity-related tweets, totalling 330. Our team then analysed the data from the right chart. It revealed that the percentage of individuals with an education level below undergraduate was relatively low in Melbourne and Sydney, at 75.1% and 73.5%, respectively. This indicates that Melbourne and Sydney have a higher proportion of individuals with a certain degree of higher education compared to other GCC areas. However, they also have a higher number of individuals who engage in profanity. This suggests that there is no significant relationship between education level and the tendency to use profanity in comments. This could be explained by the fact that personal values, family background, social environment, and individual traits also influence a person's inclination to use profanity. Some individuals may be more prone to using vulgar language in specific situations or when experiencing intense emotions, and education level may not necessarily completely alter this tendency.



```
{
  "_id": "_design/countGccLang5",
  "_rev": "1-f6ed0e3fac21801e890596be36e47c92",
  "views": {
    "my_view": {
      "map": "function(doc) {\n                emit([doc.gcc, doc.doc.data.lang], 1);\n    }",
      "reduce": "function(keys, values) { return sum(values); }"
    }
  }
}
```

Fig 25: Example of Map Function

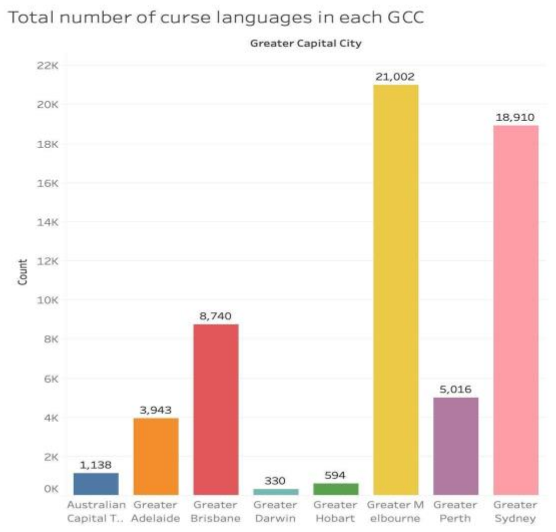


Fig 26: Total number of Profinity lang per gcc

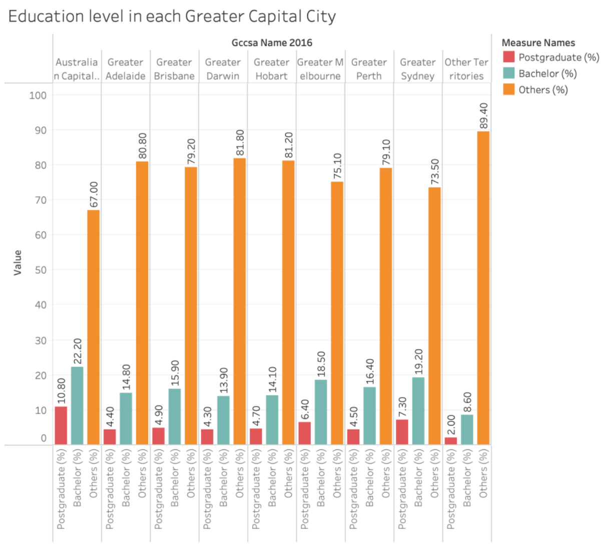
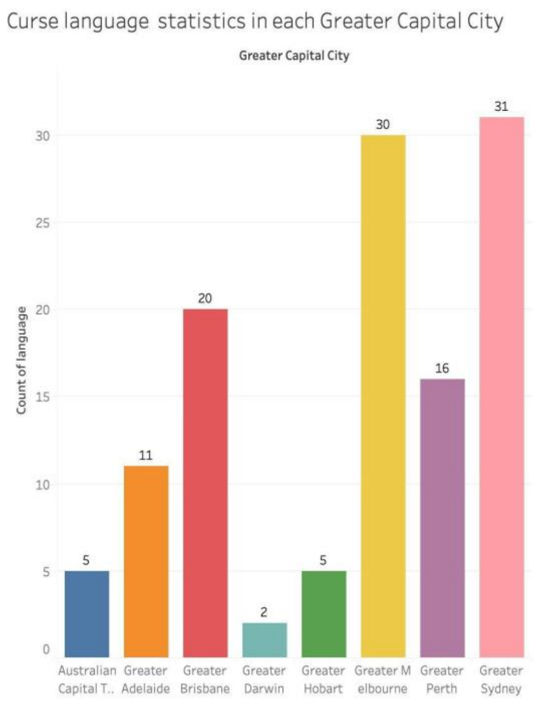


Fig 27: Education level in per gcc

4.2.4 Analysis of the correlation between education level and the diversity of profanity language

Another interesting subsection inside related to this scenario is education level and diversity of profanity languages, which the hypothesis of this subsection is if a person has a higher level of education, they are more likely to have more knowledge, and they will think more when they are angry which could reflect into the diversity of profanity words they speak. Including the language and profanity words number in the tweet comment.

In order to get the information of the diversity of profanity languages, our group write a designed documents which our team recorded the number of different profanity words used in each region, as well as the number of different languages used for profanity, to determine whether higher education levels are associated with a greater diversity of profanity expressions. Using the key-value paired data, our group plots two bar charts which indicate the number of Profanity words each gcc and the number of languages which include in tweet respectively. Which all shows the similar results in which both Greater Melbourne and Greater Sydney got the greatest number of profanity words and the languages. With enough evidence to reject the hypothesis, the population with lower levels of education in these two regions is the lowest, but they do not exhibit a higher number of profanity words or languages used for profanity. When considering the education levels in other regions as well, it appears that these three factors do not have a significant correlation.



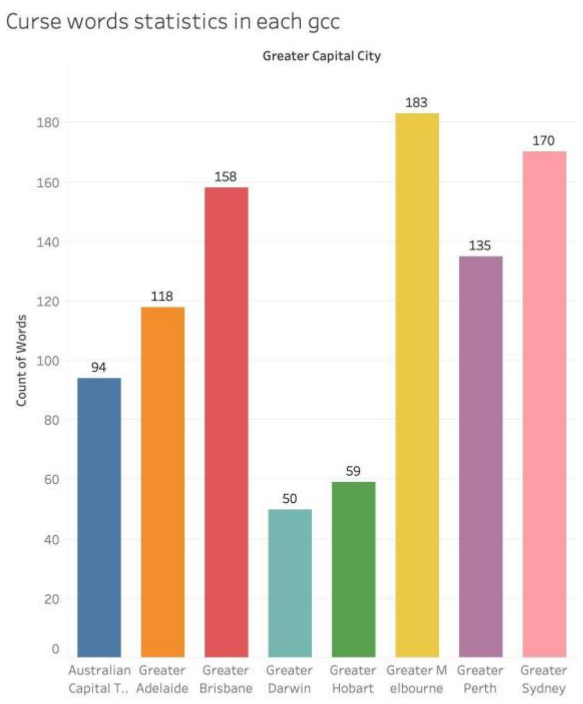
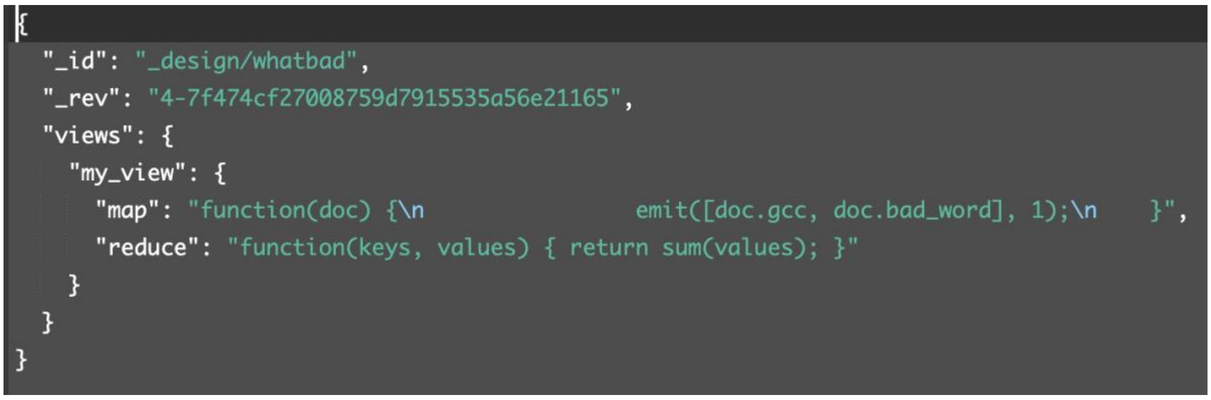


Fig 28: Number of Curse words each gcc

Fig 29: Number of lang statistics each gcc



```
{
  "_id": "_design/whatbad",
  "_rev": "4-7f474cf27008759d7915535a56e21165",
  "views": {
    "my_view": {
      "map": "function(doc) {\n                emit([doc.gcc, doc.bad_word], 1);\n    }",
      "reduce": "function(keys, values) { return sum(values); }"
    }
  }
}
```

Fig 30: A sample of Map Function

4.3 Scenario Two: Proportion of highly educated people per GCC vs Crime topic in tweets per GCC

As we seek to explore the dynamic relationship between education levels and public discourse around crime, it's instrumental to first understand the trends of crime rates within the studied regions. To that end, this section starts with an examination of crime data, specifically miscellaneous offences and illicit drug offences rate, reported by the Australian Bureau of Statistics from 2008-09 to the 2021-22 financial years.

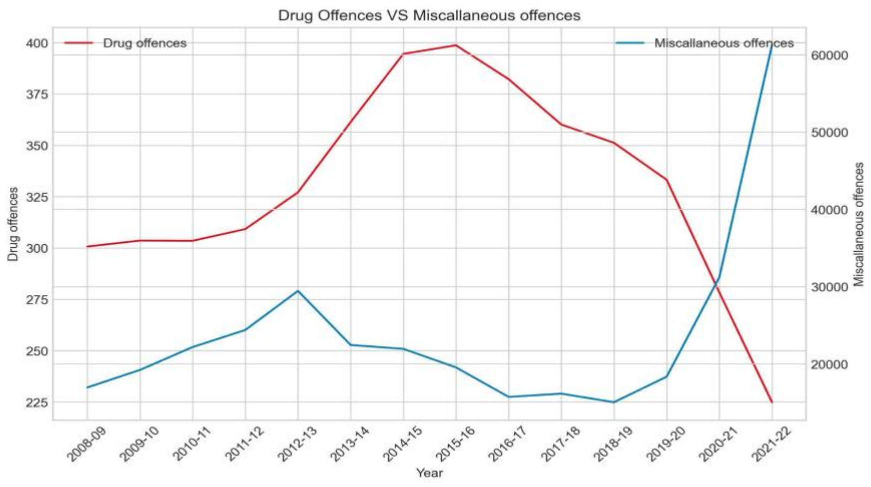
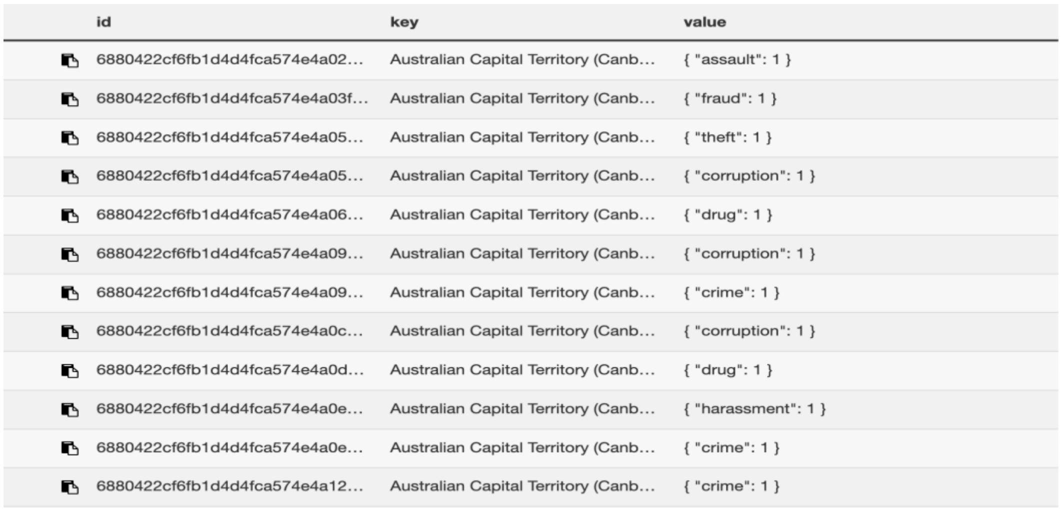


Fig 31: Drug Offences & Miscellaneous Offences [3]

Our starting point lies in the evaluation of the total number of offenders implicated in miscellaneous offences. There is some obvious fluctuation in our line chart visualisation, which indicates a variable criminal landscape over the years. However, the notable change in the 2021-22 period, which saw the number of offenders jump to 61,229 from 31,209 the previous year, deserves particular attention.

Alongside this, we analysed the rate of illicit drug offences, which showed a different trend. This rate, calculated per 100,000 individuals aged 10 years and over, demonstrated a gradual incline until 2015-16, and subsequently a steady decline through 2021-22.

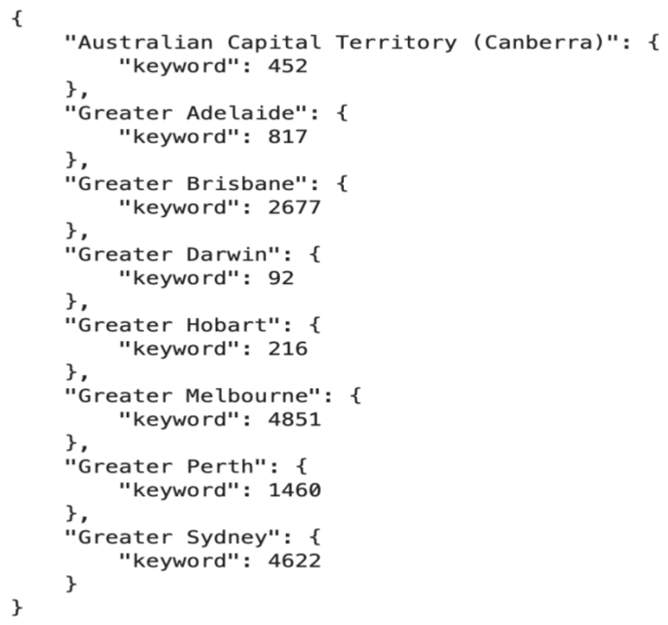
Driven by the two greater changes on the end of the line, we pivot towards the core of our investigation: assessing the relationship between the proportion of highly educated individuals within Great Capital Cities (GCCs) and the prevalence of crime-related discussions on social media platforms like Twitter.



id	key	value
6880422cf6fb1d4d4fca574e4a02…	Australian Capital Territory (Canb…	{ "assault": 1 }
6880422cf6fb1d4d4fca574e4a03f…	Australian Capital Territory (Canb…	{ "fraud": 1 }
6880422cf6fb1d4d4fca574e4a05…	Australian Capital Territory (Canb…	{ "theft": 1 }
6880422cf6fb1d4d4fca574e4a05…	Australian Capital Territory (Canb…	{ "corruption": 1 }
6880422cf6fb1d4d4fca574e4a06…	Australian Capital Territory (Canb…	{ "drug": 1 }
6880422cf6fb1d4d4fca574e4a09…	Australian Capital Territory (Canb…	{ "corruption": 1 }
6880422cf6fb1d4d4fca574e4a09…	Australian Capital Territory (Canb…	{ "crime": 1 }
6880422cf6fb1d4d4fca574e4a0c…	Australian Capital Territory (Canb…	{ "corruption": 1 }
6880422cf6fb1d4d4fca574e4a0d…	Australian Capital Territory (Canb…	{ "drug": 1 }
6880422cf6fb1d4d4fca574e4a0e…	Australian Capital Territory (Canb…	{ "harassment": 1 }
6880422cf6fb1d4d4fca574e4a0e…	Australian Capital Territory (Canb…	{ "crime": 1 }
6880422cf6fb1d4d4fca574e4a12…	Australian Capital Territory (Canb…	{ "crime": 1 }

Fig 32: Example of view of map function

In undertaking this process, our goal is to study deeply public sentiment and discourse concerning crime within historical Twitter data. To gain insights into the frequency of each crime-related keyword, we can build a specific view in CouchDB. Given that CouchDB natively supports MapReduce, we're able to efficiently construct these views, thereby providing a valuable perspective on the occurrence of various crime-related terms.



```
{
    "Australian Capital Territory (Canberra)": {
        "keyword": 452
    },
    "Greater Adelaide": {
        "keyword": 817
    },
    "Greater Brisbane": {
        "keyword": 2677
    },
    "Greater Darwin": {
        "keyword": 92
    },
    "Greater Hobart": {
        "keyword": 216
    },
    "Greater Melbourne": {
        "keyword": 4851
    },
    "Greater Perth": {
        "keyword": 1460
    },
    "Greater Sydney": {
        "keyword": 4622
    }
}
```

Fig 33: Example demo of result from reduce step

We have developed three views using CouchDB's MapReduce model. Each view uses a map function to process the input data, and a reduce function to aggregate the results.

The first view's map function checks each document for a GCC name and a count of crime-related keywords, emitting a key-value pair with the GCC name and a dictionary of the keywords like Fig 3. It reduces function sums up the counts for each keyword per GCC.

The second view applies a similar logic but emits an object containing the keyword with a count of 1 as the value in its map function. It reduces function operates like the first, aggregating the keyword counts for each GCC. The third view's map function emits a key-value pair with the GCC name and a value of 1 for each document with a GCC name, representing a single tweet. It reduces function then sums up these values, yielding the total count of tweets per GCC.

Once these views are created, the results can be retrieved using a CURL request, the result would be present like the Fig.4, and visualisations can be generated with the help of tools like Tableau or Python.

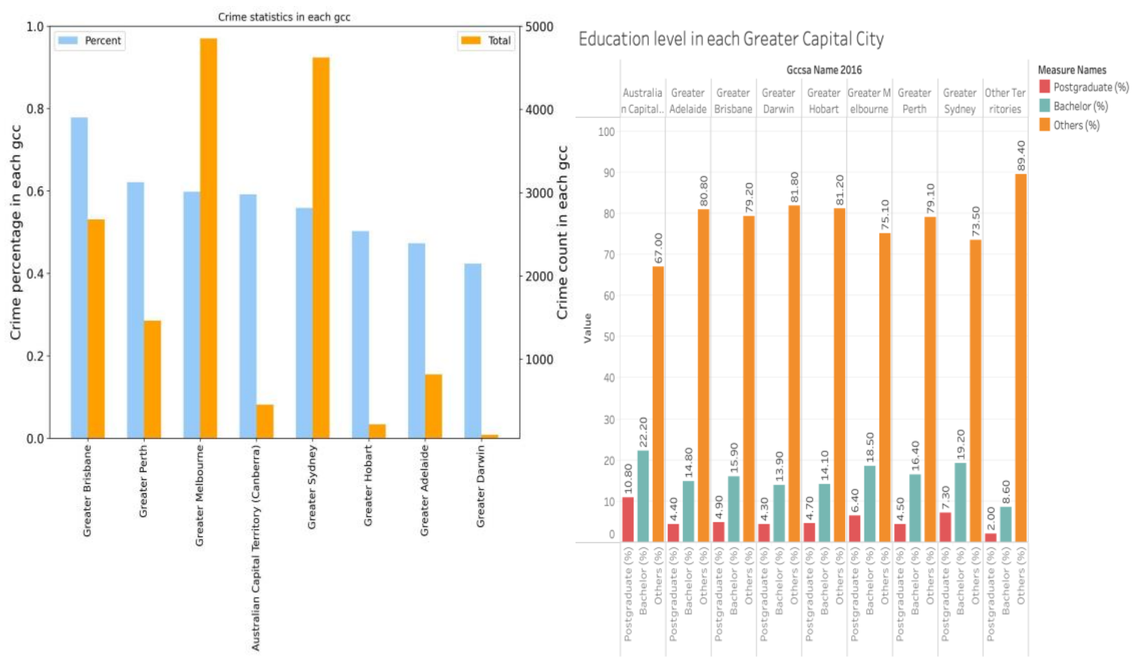


Fig 34: Percentage of crime keyword in the Twitter in each Great Capital City

Now that we've established the context of crime rates within the studied regions. We'll first analyse data on crime-related tweets, based on the occurrence of crime keywords. From our dataset, we have calculated the percentage of tweets containing crime keywords for each GCC. Our bar chart illustrates a varied landscape with Greater Brisbane leading with 0.78%, closely followed by Greater Perth at 0.62%. At the lower end of the spectrum, we have Greater Darwin with a 0.42% tweet frequency.

Alongside the percentage representation, we also have absolute numbers of tweets containing crime keywords from each GCC. A glimpse at this data set reveals Greater Melbourne and Greater Sydney to be the primary sources of crime-related tweets with 4,851 and 4,622 occurrences, respectively.

On the other side of our investigation, we also put data representing the educational level across the GCCs again, categorised into bachelor's degree holders, Postgraduate degree holders, and Others. Which is used to compare the relationship between the tweets containing crime keywords on the left.

Interestingly, Greater Melbourne and Sydney, with high education levels, record the most crime-related tweets. Greater Brisbane, with lower degree holders than Greater Melbourne and Sydney, however, has the highest proportion of crime-related tweets at approximately 0.78%. It suggests a complex relationship between education, which encourages us to investigate any further analysis on the tweets.

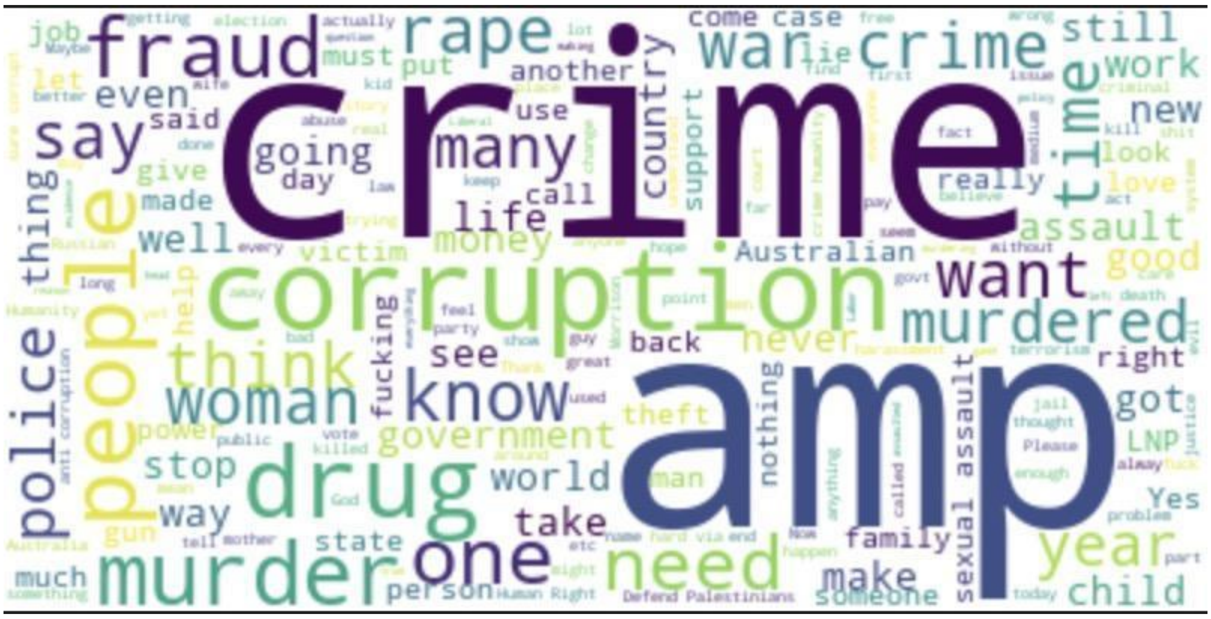


Fig 35: Word Cloud of all Crime Keyword

Utilising the capabilities of the Python WordCloud library, we can present a visualisation of the prevalence of crime-related keywords in tweets. In our generated WordCloud, keywords such as 'crime', 'corruption', 'fraud', 'drug', and 'murder' and some crime related keywords appear to have a significant presence. The size of the font corresponding to each keyword in WordCloud serves as an indicator of its relative frequency or impact within the collected tweets.

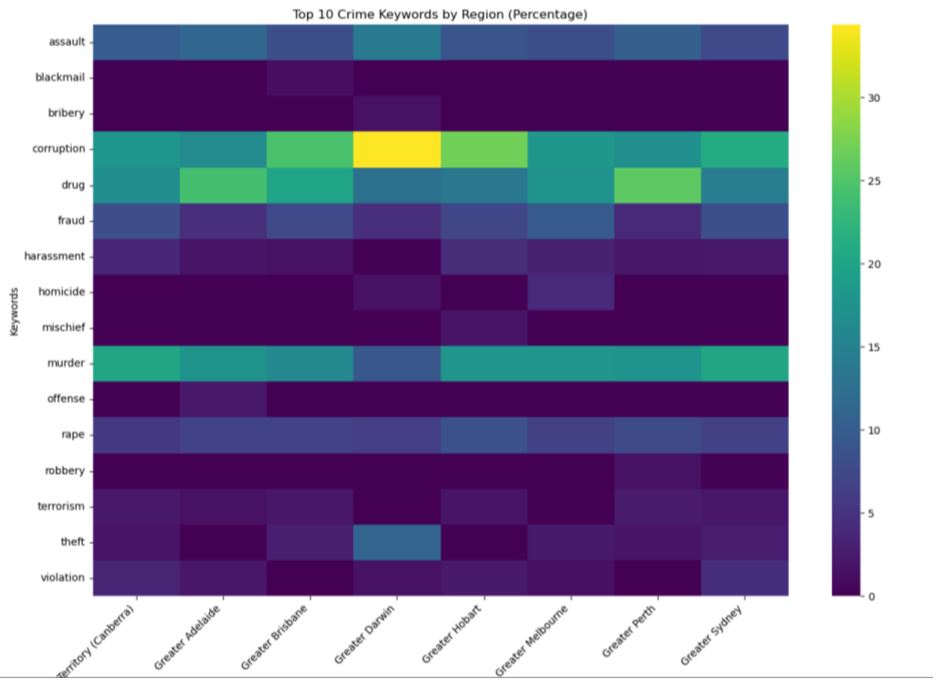


Fig 36: Heatmap of Crime Keyword

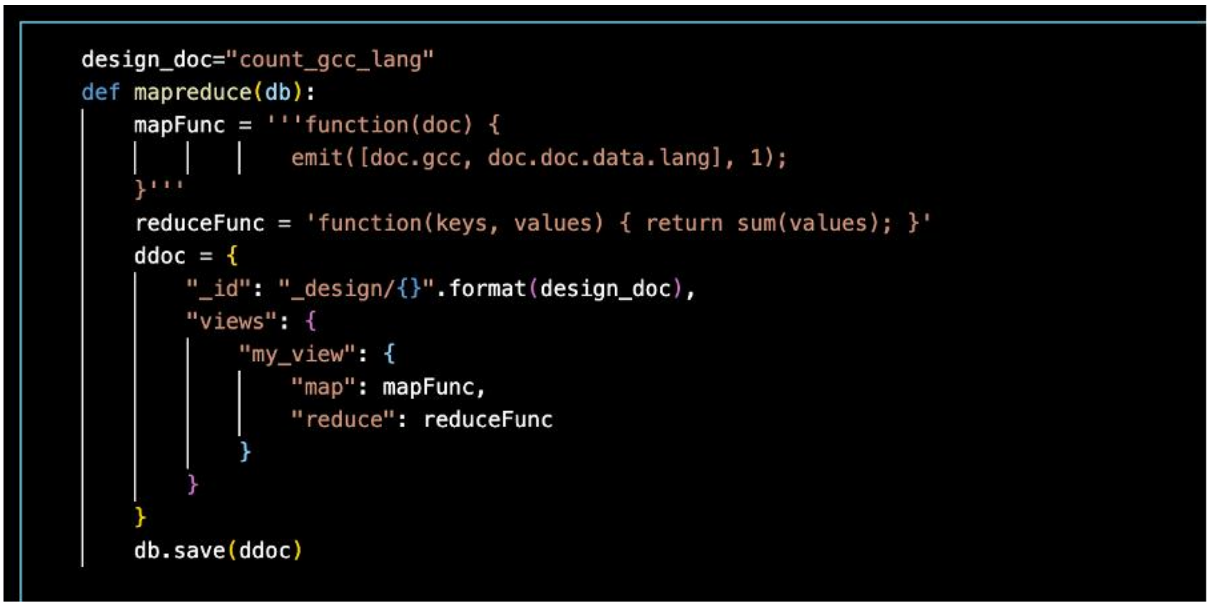
Beyond examining the global word frequency, we extend our analysis to consider the unique impact of each crime type within the different GCCs. Utilising the 'seaborn' library in Python, we have constructed a heatmap displaying the distribution of the top 10 crime keywords across the GCCs. This heat map uses a colour spectrum to represent the percentage of tweets containing particular crime keywords within a specific region, with a lighter colour (ex. Yellow) indicating a larger percentage.

Through this heatmap, it is evident that the keyword 'corruption' occupies a significant share of crime-related tweets in Greater Darwin. Concurrently, we observe that 'drug' and 'murder' are prevalent topics in discussions surrounding crime across the regions. This visualisation allows us to gain an understanding of regional variances in crime discourse, highlighting specific areas of concern for each city.

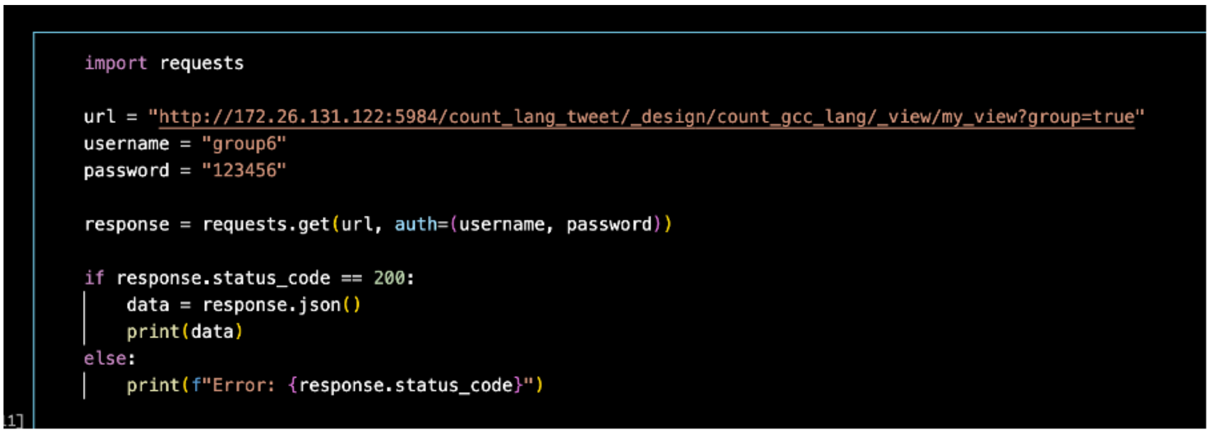
4.4 Scenario Three: Proportion of highly educated people per GCC vs Linguistic diversity per GCC

4.4.1 Data Processing

After uploading the tweets data with location information, we need to use MapReduce to count how many unique languages in tweets per great capital city. It means the statistical data can be used to estimate the linguistic diversity of each great capital city. As seen from the coding below, first we search the information of "gcc" and "lang" that is required, and then use the counter of "1" to count the number of different languages per gcc.



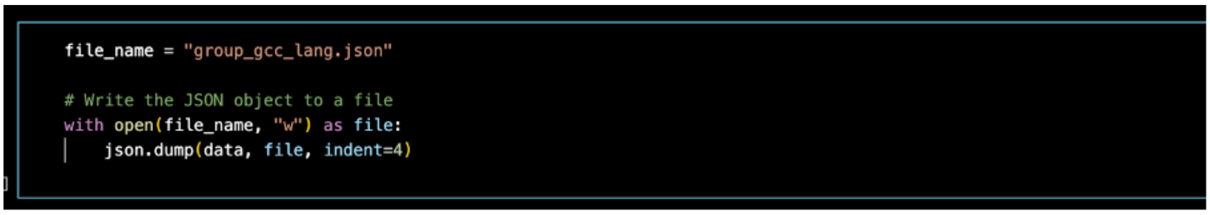
```
design_doc="count_gcc_lang"
def mapreduce(db):
    mapFunc = '''function(doc) {
            emit([doc.gcc, doc.doc.data.lang], 1);
    }'''
    reduceFunc = 'function(keys, values) { return sum(values); }'
    ddoc = {
        "_id": "_design/{}".format(design_doc),
        "views": {
            "my_view": {
                "map": mapFunc,
                "reduce": reduceFunc
            }
        }
    }
    db.save(ddoc)
```



```
import requests

url = "http://172.26.131.122:5984/count_lang_tweet/_design/count_gcc_lang/_view/my_view?group=true"
username = "group6"
password = "123456"

response = requests.get(url, auth=(username, password))

if response.status_code == 200:
    data = response.json()
    print(data)
else:
    print(f"Error: {response.status_code}")
```

Fig 37: MapReduce for group_lang_gcc

Also, we need to write the data into the json file “group_gcc_lang.json” used for further graphing.



```
file_name = "group_gcc_lang.json"

# Write the JSON object to a file
with open(file_name, "w") as file:
    json.dump(data, file, indent=4)
```

Fig 38: Coding for writing into file

4.4.2 Analysis

As seen from the figure from the SUDO data, the greater capital cities with the top 3 proportion of non-school qualifications are Australian Capital Territory, Greater Sydney and Greater Melbourne. Also, the proportion of non-school qualifications refers to the rate of persons with educational attainments except those of a pre-primary, primary or secondary education level.

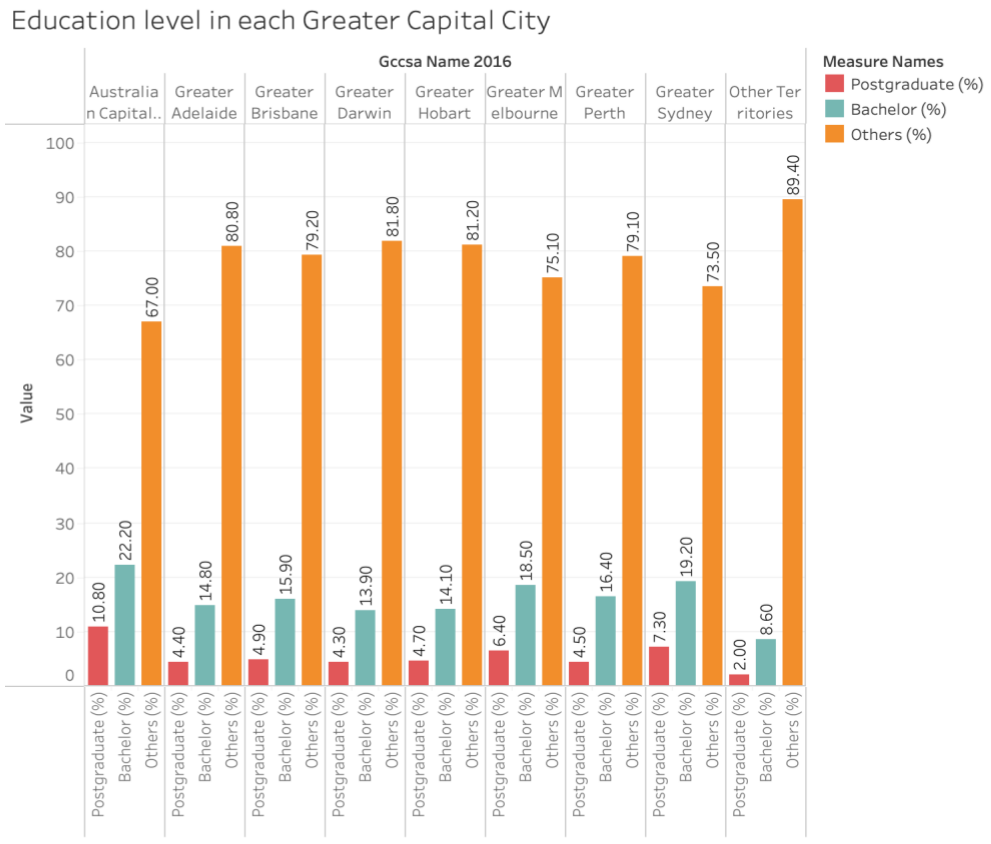


Fig 39: Bar Chart for Education Level

Hence, it means the higher the ratio for "others", the lower the ratio for "non-school qualifications" includes "bachelor postgraduate".

Furthermore, as seen from the bar chart from the historical Twitter data, the areas with the top 3 number of unique languages in tweets are Greater Sydney, Greater Melbourne and Greater Brisbane. Therefore, for the data of Greater Sydney and Greater Melbourne, it shows that the cities with higher proportion of "non-school qualifications" have greater linguistic and cultural diversity, reflecting the contribution of the cities to pluralistic societies.

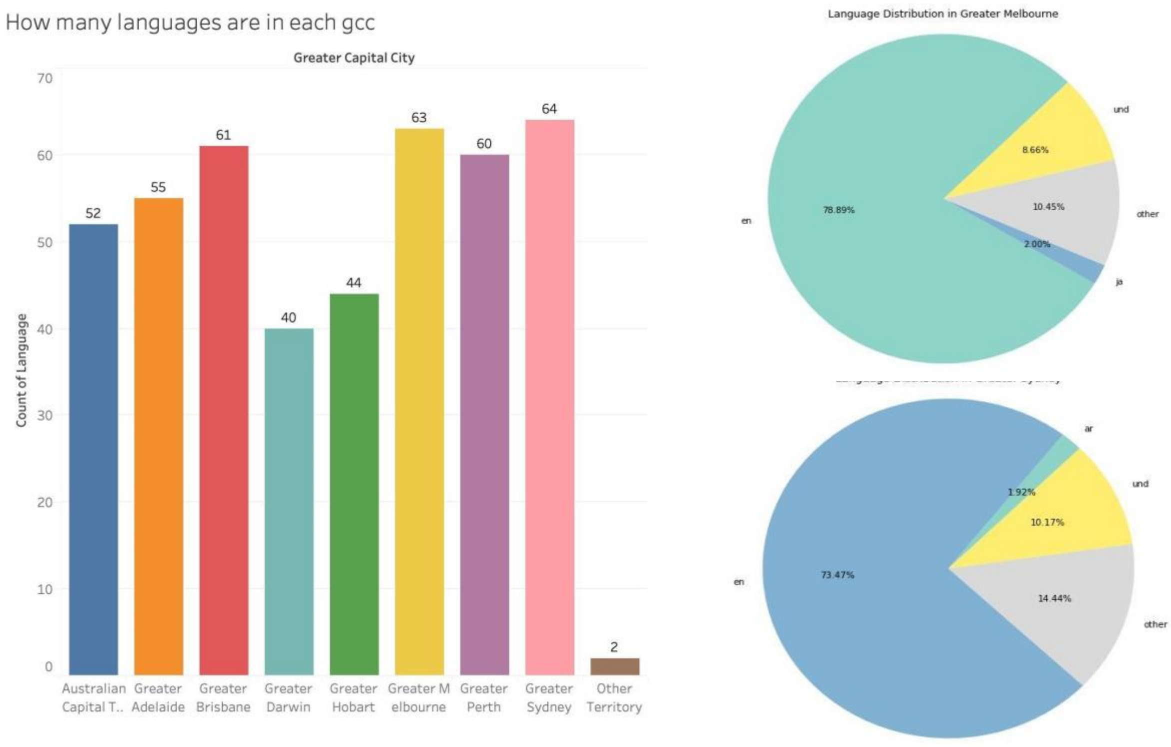


Fig 40: Bar Chart & Pie Chart for # of Languages in Tweets per GCC

However, the highest proportion of persons with high education level for the Australian Capital Territory from SUDO data is not consistent with the number of unique languages for this area from Twitter data. It reflects the viewpoint that a higher proportion of high education levels may not lead to greater cultural inclusiveness. Also, it can be due to the error of comparison and prediction by such a method. In other words, the number of unique languages in tweets does not objectively reflect the linguistic diversity of one area.

5. Web Application

Using GeoJSON, HTML, CSS, JavaScript, and ArcGIS, as well as several other web pages written in HTML, CSS, and JavaScript.

• HTML provides the basic structure of sites, which is enhanced and modified by other technologies like CSS and JavaScript.

• CSS is used to control presentation, formatting, and layout.

• JavaScript is used to control the behaviour of different elements.

• GeoJSON is a format for encoding a variety of geographic data structures.

• ArcGIS is a family of client, server, and online geographic information system (GIS) software developed and maintained by Esri.

6. User Guide

The source code of this assignment is available online and can be checked below.

Source Code Repository:

https://github.com/Wen20011009/CCC-ass2-team6.git

Demonstration of Web Application:

https://junbohcom.wpcomstaging.com/

Video Demonstration:

https://youtu.be/EBhZff-8_gI

6.1 System deployment

Here was the documented user guide of how the whole system is built, and README.md in the GitHub Repository also could help you to set up the environment, so this part would be a detailed or complementary version to the README.md. At first, the user should have any necessary library installed on the device that is stated in the 'requirment.txt' in GitHub. Secondly, the file name 'ansible & docker/deploy_instance' contains all the files you need to set up an instance, following the readme inside the folder, the user would have an instance deploy in the cloud, and then the other notebook would finish the setting of CouchDB, which means that after the deployment of instance, the user could access their CouchDB through 'http://instance_ip:5984'. Then, we also prepare a notebook to deploy WordPress in one of the instances, after following the guide of WordPress deployment in GitHub in 'ansible & docker/deploy_wordpress', the user could access their website through any of the devices by 'http://instance_ip'. Then, we also prepare a notebook to deploy the Python script in a container to a specific instance, which could be running a script that continually collects the data from Mastodon to CouchDB, at last, the notebook in the 'error_handling' file would be running the disk checking shell script to the specific instance, which would stop the harvester container when the disk usage reaches 90%. After that, the whole environment of the system is built, and the user could perform their own analysis using the system or follow the guide in GitHub to produce a similar analysis of our work.

6.2 Usage of website

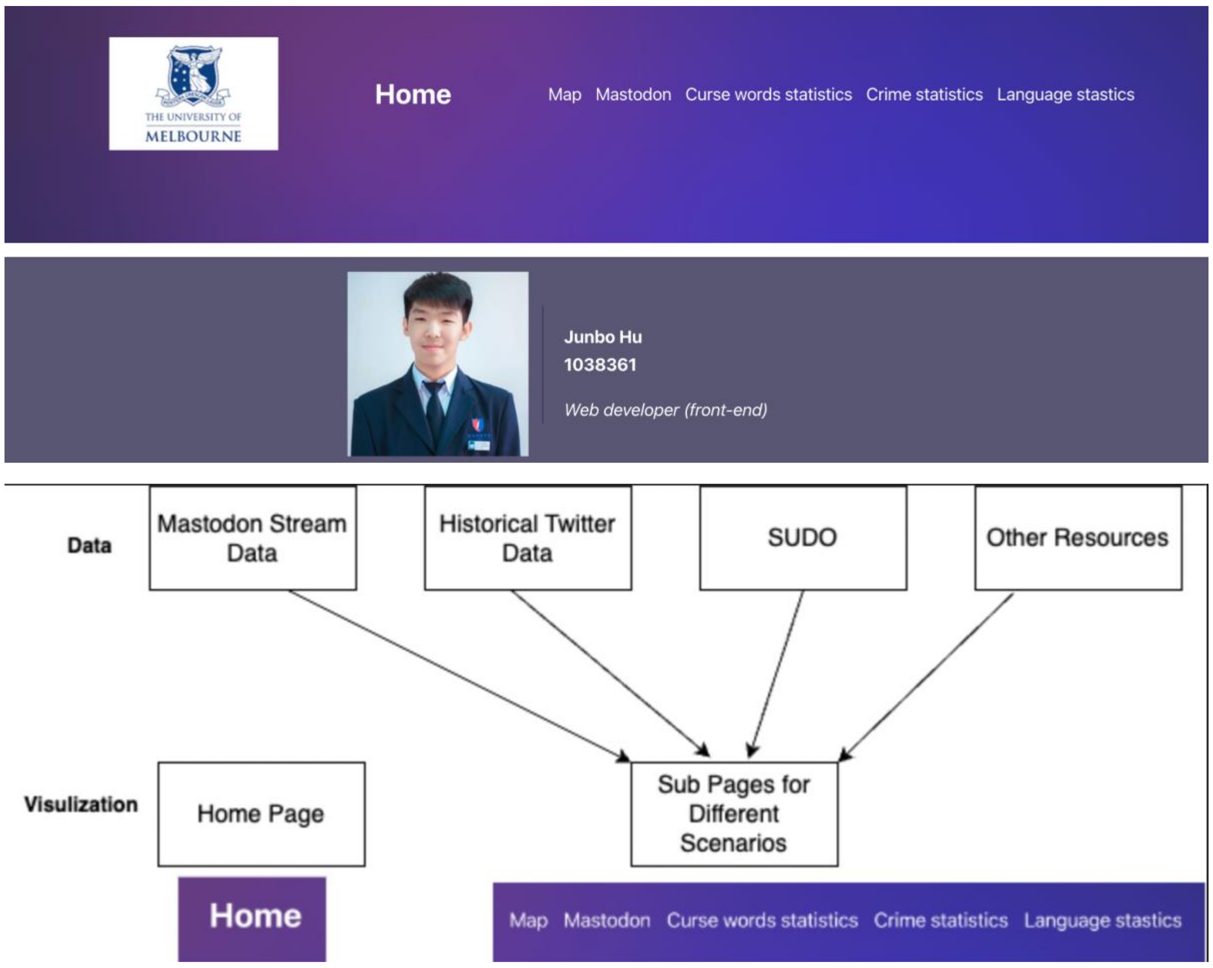


Fig 41: Web Structure

The data of the three scenarios to be presented include education level, crime, curse words and language data collected from Mastodon Stream, Historical Twitter, Sudo and Other Resources and supplied by CouchDB and mapreduce computations. The main page shows information about group members and the links to 5 sub-pages consisting of three scenarios, Map, and mastodon part.

The border geographic data of Australia was processed by geojson and arcgis which is an active map to be the form for visualisation on the Map page.

7. Arrangement of each member

During this assignment, each group member took an active role in the group tasks and discussions, and each member contributed suggestions and solutions. Each of the team members was tasked in a separate set of responsibilities which are further stated in the table 1 below. During the allocation and communication process, our group often holds face to face meetings, and Trello is used to assign tasks.

Team Member	Task or Role
Tianhao Liu	Harvester, Data Analyst (MapReduce)
Bohan Su	Ansible Deployment, Data Analyst (MapReduce)
Xiangmin Zhou	SUDO, Data Analyst (MapReduce)
Wenwen Zhang	UI, SUDO, Data Processing
Junbo Hu	Web developer (front-end)

Table 1: Overview of each team member's responsibilities

8. Conclusion

let's summarise the topic selection for our project. In this project, our group chose education as the overarching theme, which is closely aligned with current trends. Based on popular discussions, our group identified several relevant directions within this theme, such as crime rate, language diversity/cultural inclusivity, and level of civilization. Through our research and analysis, we found that although some aspects may not have a direct connection to education, there were intriguing findings that reflected many interesting facts about Australia. In the future, our group can conduct more in-depth research on these topics.

This project is an open-minded project from Cluster and Clouding computing which is a brand-new area our group has been faced with. After finishing this project, there are some parts our group did well and some parts which we need to improve on. One thing our group did well is we had clear group responsibilities allocation, with different people doing different jobs, including Ansible, Data pre-processing, CouchDB, MapReduce, frontend and backend. Each person does their own job, and we hold weekly meetings to discuss how things are going. Our group also did a great job in preparing for this project by spending a lot of time learning about Cluster and Cloud Computing, as well as what the main focus is on this project, and how important it is to know about clustering and all of its skills, including Ansible and cloud computing. Additionally, there are some areas where our group needs to improve. There is a cluster in our organization, and our group could spend more time discussing how different nodes can collaborate effectively. In order to increase the efficiency of the system, the main node should be able to manage other worker nodes, and those worker nodes should be able to help each other.

9. Reference

[1] Vidhya, Analytics. "Topic Modeling Using LDA and Gibbs Sampling Explained." Medium.com, 25 May 2023, https://medium.com/analytics-vidhya/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045.

[2] Wang, X., & Blei, D. M. "Collaborative topic modeling for recommending scientific articles." Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, PMLR, 2011, pp. 448-456. DOI: 10.1145/2020408.2020490.

[3] Australian Bureau of Statistics. 'Offenders, Principal Offence of Miscellaneous Offences.' Recorded Crime - Offenders, 2021-22 Financial Year. Table. Unpublished raw data. Australian Bureau of Statistics, 2022.