COMP90002 Cluster and Cloud Computing Assignment 2

The Seven Deadly Sins in Social Media Case Study in Victoria Cities

Team 14

THE UNIVERSITY OF MELBOURNE

Dading Zainal Gusti (1001261)
David Setyanugraha (867585)
Ghawady Ehmaid (983899)
Indah Permatasari (929578)
Try Ajitiono (990633)

A. Introduction

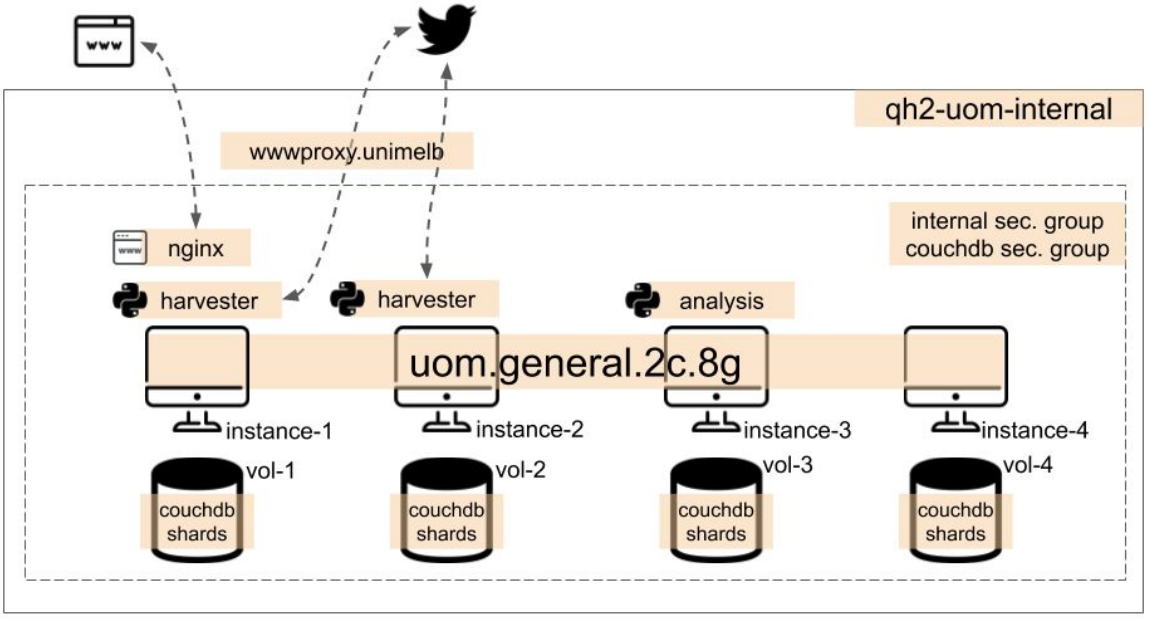
The main goal of this project is to detect the seven deadly sins in Social Media. These sins are pride, greed, lust, envy, gluttony, wrath and sloth (Aquinas, 2012). In this project, we focus on detecting sins on social media user within Victoria, Australia. Once the sins are detected, we want to use this information for a greater purpose. Using this information, our next goal is to find the best place to live in Victoria or the best liveable place. To validate these findings, we aggregate social media data with other data, in this case, AURIN (Australian Urban Research Infrastructure Network). AURIN is a research infrastructure which provides lots of interesting data within Australia. We provide an extensive analysis for each finding we get while doing this project.

In order to achieve our goals, we need a robust system that can handle big data from social media. Thus, we utilize cloud instances to store the data inside CouchDB Cluster. We use Ansible to deploy the system. We use Python scripts to create analysis scenarios for this project. These Python scripts communicate directly with CouchDB and produce the output that is later used for visualization. We provide several maps and charts visualization to help the user understand the insight of this project. We also employ several strategies to make this system work well.

This report is divided into nine sections: Introduction, System Design, Implementation, Unimelb Research Cloud (Nectar) Evaluation, AURIN Indicators, Twitter Analysis, Conclusion, Reference and Appendix. Introduction covers all the background and purpose of this project. System Design covers all the details in designing the system from instances and storage, CouchDB, Network and Security Group, until Tweet Harvester. Implementation section is all about the processes and issues we got when implementing this system. Unimelb Research Cloud (Nectar) Evaluation section covers everything about the advantages and limitations of using Nectar. AURIN Indicator section gives the explanation about how we integrate AURIN data to validate seven sins. Twitter Analysis covers all steps we did to land into the insights. Twitter Analysis also covers range of topics such as Data Pre-Processing, Unliveable Score, Challenges and Visualization Results. Conclusion section concludes everything about building infrastructure process and useful analysis insights. Reference includes all literatures used in this project and technology stacks documentations. The report is closed with Appendix which contains Role of team members, User Testing Guide, Source Code Link and Video Link.

B. Overall System Design

The major purpose of the target system is to provide insights about life in Australian cities based on harvested data from social media and provide visualization to these findings. This led to the necessity for an efficient use and utilization of cloud based infrastructure, for data collection, processing and retrieval. As part of the requirements, some aspects such as instance creation, storage, CouchDB cluster, network and security groups setup, and web server are deployed using Ansible automation script. The figure below illustrates the components of the overall system architecture, followed by a description of each component.



System Architecture

Instances and Storage

We built four VM instances, each consists of 2 cores vCPU and 8 GBs of RAM (30 GBs of storage comes by default with uom.general.2c.8g flavor). Four volumes of 60 GBs each are created and attached one to each of the instance.

CouchDB cluster

CouchDB is configured in clustered mode on top of the four instances. We decided to configure CouchDB to utilize the mounted volume (60 GBs each) to keep the shard files.

Network and security group

We use `qh2-uom-internal` network for this project, as well as two custom security groups, **internal** and **couchdb**. For **internal** security groups, we enabled all TCP, UDP, and ICMP for ingress and egress so each instance inside the network can communicate to each other freely. We don't want to limit certain capabilities with internal nodes connection. On the other side, for **couchdb** security groups, we only open some needed ports to the world, for example: all ICMP ports, TCP port 22 (SSH), 80 (HTTP), and 443 (HTTPS), and CouchDB related ports (4369, 5984, 5986, and 9100-9200).

Tweet harvester

There are two types of harvesters, the first one is a live streaming harvester to collect tweets posted from Victoria. The other one is pulling user ids captured from the streaming harvester with the help of CouchDB view and then it fetches timelines of each unique user. All the harvesters run in parallel.

Web Application

To serve our web application, we are using Nginx as the web server. We don't serve our web application in every instance, because in order to do so, we need a tool such as

HAProxy deployed in more than 1 node to do dynamic load balancing to prevent single point of failure. However, we decided not to use that as we have the limitation in resource infrastructure.

C. Implementation

Infrastructure Deployment

The system's infrastructure is deployed in the University of Melbourne Research Cloud, inside the team's project directory. We were allocated a quota of 4 VM instances, 8 vCPUs, 36GB of RAM, and 250GB of storage. The first task that is done in project is to build an automation script for creating, configuring and managing the instances in Nectar using Ansible. The following figure is a snapshot from the Ansible playbook and each component is explained afterwards.

```
# filename: ansible/nectar.yaml
- hosts: localhost
  vars_files:
    - host_vars/nectar.yaml
  gather_facts: true
  roles:
    - role: build-common
    - role: build-volume
    - role: build-security-groups
    - role: build-instance
    - role: build-node-groups

- hosts: remote
  vars_files:
    - host_vars/nectar.yaml
  remote_user: ubuntu
  gather_facts: true
  roles:
    - role: setup-remote-proxy
  vars:
    ansible_ssh_private_key_file: '~/cloudkey.pem'

- hosts: remote
  vars_files:
    - host_vars/nectar.yaml
  remote_user: ubuntu
  gather_facts: true
  roles:
    - role: setup-remote-couchdb
    - role: setup-remote-nginx
    - role: setup-remote-dep
    - role: run-harvesters
  vars:
    ansible_ssh_private_key_file: '~/cloudkey.pem'
```

Components	Description
`host_vars/nectar.yaml`	Contains variables used in ansible tasks
`role: build-common`	Setup dependencies for localhost to perform ansible tasks

`role: build-volume`	Creates volumes based on configuration specified in host_vars/nectar.yaml
`role: build-security-groups`	Creates security group and its corresponding rules to support couchdb and internal cluster communication
`role: build-instance`	Creates instances on Nectar
`role: build-node-groups`	Creates a group consists of ip address of created instances
`role: setup-remote-proxy`	Apply internet proxy setting to instances, followed by reset connection
`role: setup-remote-couchdb`	Setup couchdb cluster in remote nodes
`role: setup-remote-nginx`	Setup nginx in one node
`role: setup-remote-dep`	Install all dependencies to run harvesters
`role: run-harvesters`	Create database and run tweet two harvesters in two nodes

Below are some aspects we found critical to be performed to be able to spin up the VMs:

1. Instances are set to use floating IP on `qh2-uom-internal` network, therefore we keep the IP of each created instances then add them into a group. Next ansible tasks will benefit from this group to perform tasks on remote nodes

```
# snippet from ansible/roles/build-node-groups/tasks/main.yml
- name: adding all nodes to remote group
  add_host:
   groups: remote
   name: '{{ item }}'
  loop: '{{ nodes_ips }}'
  register: remote_nodes
```

2. The just-created instances are not ready to accept ssh, therefore a task to wait for ssh connection to be available is done for each instance.

```
# snippet from ansible/roles/build-node-groups/tasks/main.yml
- name: Wait 10 seconds for port 22 to become open and contain OpenSSH
  wait_for:
    port: 22
    host: '{{ item }}'
    search_regex: OpenSSH
    delay: 10
  loop: '{{ nodes_ips }}'
```

3. `qh2-uom-internal` network, as the name insists, is an isolated network. Therefore, we need to apply proxy setting to be able to connect to internet. While applying proxy setting is trivial (with ansible `block` module to append proxy settings in `/etc/environment` file), this setting is not applied immediately. Thus, we ask ansible to reset ssh connection without stopping the running play (to simulate user logout in remote instances).

```
# snippet from ansible/roles/setup-remote-proxy/tasks/main.yml
- user: name={{ansible_user}}
- name: reset ssh connection to allow user changes to affect 'current login user'
  meta: reset_connection
```

Cluster CouchDB

CouchDB plays an important role in this project. It manages the data gathered from both Twitter and AURIN, it also plays a pivotal role in the analysis with its built-in MapReduce capability. It is also a clustered database as in the data are replicated and distributed into nodes within a cluster. Database operations as well as setting up the cluster are assisted with HTTP API provided. Every node is a peer, as in no such thing as the master node, so every request or operation is propagated to all nodes and the response is decided based on the majority response from all nodes.

We deployed CouchDB cluster using Ansible and chose the default configuration (3 replications, 8 shards), this means not every node in our cluster is having the same shards. We consider to use 3 replications only as suggested in CouchDB documentation. Adding too many value of replications only adds the complexity without any real benefit. Some challenges were faced during the installation, one is translating the guide described in CouchDB documentation (http://docs.couchdb.org/en/master/setup/cluster.html) into Ansible instructions. The biggest challenge comes when applying the cluster setup. We required a considerable amount of time to repeat the instructions given in CouchDB documentation and to make sure all the requirements are satisfied. It is later found that the default installation (apt-get install CouchDB) was the root cause of the problem; `couchdb@127.0.0.1` was being added by default into its own cluster setup. This led us into passing CouchDB installation parameters using a non-interactive script, e.g. node name, bind address, cookie, admin username, and admin password (illustrated in figure below). We came to this solution because, in the interactive installation, the user is required to input these fields.

```
#snippet from ansible/working_scripts/00-install-couchdb.sh
ARG_NODENAME=$1
NODENAME=${ARG_NODENAME:-127.0.0.1}

COUCHDB_PASSWORD=password
...
couchdb couchdb/nodename string couchdb@${NODENAME}
...
couchdb couchdb/adminpass password ${COUCHDB_PASSWORD}
...
couchdb couchdb/adminpass_again password ${COUCHDB_PASSWORD}
...
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y couchdb
```

Upon successful installation, we set up the cluster by following the step-by-step guide in the CouchDB documentation. Some conditions are put to accommodate changes in the cluster. For instance, when one or more new nodes are found, the coordinator node will perform “finish cluster” step. This provides the scalability of building CouchDB cluster and prevent

“Cluster is already finished” error from happening if there is no new node detected. We also put condition (whether CouchDB is installed or not) in Ansible task to make sure the task is idempotent (illustrated in figure below).

```
# snippet from ansible/roles/setup-remote-couchdb/tasks/main.yml
# Checks if couchdb is installed. Use with `"'couchdb' not in ansible_facts.packages"`.
- name: 'Check if couchdb is installed'
  package_facts:
    manager: 'auto'
...
# Save the initial state of cluster, only in coordinator node.
- name: save initial state of membership
  script: working_scripts/01-check-membership.sh {{ inventory_hostname }}
  when: inventory_hostname == groups['remote'][0]
  register: initial_membership
...
# Save the final state of cluster, only in coordinator node.
- name: save final state of membership
  script: working_scripts/01-check-membership.sh {{ inventory_hostname }}
  when: inventory_hostname == groups['remote'][0]
  register: final_membership
...
- name: finish the cluster
  uri:
    url: http://{{ groups['remote'][0] }}:5984/_cluster_setup
    method: POST
    user: admin
    password: password
    body: '{"action": "finish_cluster"}'
    force_basic_auth: yes
    status_code: 201,200
    body_format: json
    headers:
      Content-Type: 'application/json'
  when:
    - inventory_hostname == groups['remote'][0]
    - final_membership.stdout != initial_membership.stdout
    - "'couchdb' not in ansible_facts.packages"
```

Finally, we changed the configuration of each node to benefit the 60 GB volume to store the database (illustrated in figure below).

```
# snippet from ansible/roles/setup-remote-couchdb/tasks/main.yml
- name: make use 60 gb for db
  uri:
    url: http://{{ inventory_hostname }}:5984/_node/_local/_config/couchdb/database_dir
    user: admin
    password: password
    method: PUT
    body: '"/mnt/couchdb/database"'
    force_basic_auth: yes
    status_code: 200,201
  when: "'couchdb' not in ansible_facts.packages"
```

We did not see the need to implement a backend server to serve the data visualization because CouchDB’s views could be used to achieve a similar outcome as any backend

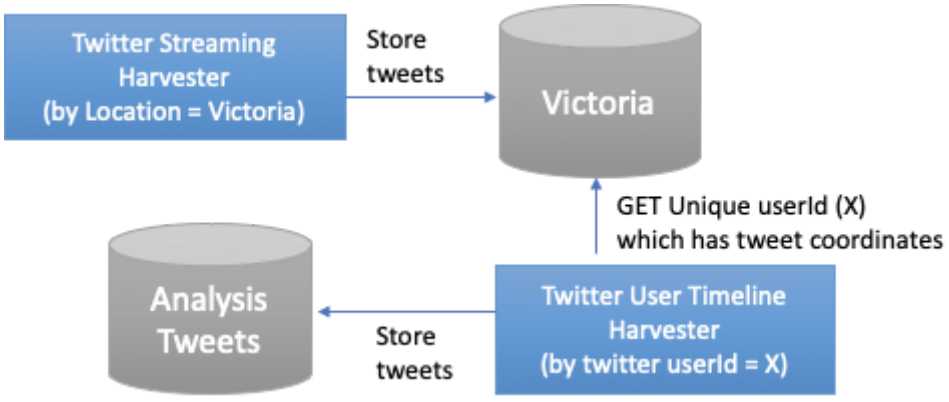
server that implements RESTful API. This is possible because CouchDB uses HTTP for the transaction, unlike other databases such as MySQL that uses LDAP (Lightweight Directory Access Protocol) which requires a driver to manipulate the database from an API server, e.g. JDBC in Java.

Fault Tolerance

According to the CAP theorem, any distributed system can only guarantee either Consistency or Availability as Partition tolerance is a given. Thus, trade-offs must be made at any point-in-time, CouchDB assure that it is highly available and partition tolerant, but it has eventual consistency. It also guarantees that its storage engine is fault-tolerant that prioritize the safety of the data. In this project we did not experience any failure cases and we believe that CouchDB handles the partition well. In terms of scalability, as we mentioned in the previous part, it is accommodated in our application. As explained in the following section, whenever a new node is added to the node list through ansible, the new node will be automatically added to the existing CouchDB cluster. Volumes are used to store the shard files so that just in case the instances went wrong, the shards are safe and can be attached to new instances. Also there is a chance to resize the volume size to accommodate more data.

Tweet Harvesting

As discussed in the system design, the implementation uses two instances of the live Tweet harvesters that utilize the Twitter Streaming API and one instance to harvest user tweet timelines. Each harvester runs in a different nectar instance to collect tweets posted from anywhere in Victoria (based on Victoria bounding box) and stores tweet text and related metadata to the CouchDB, then another process is runs separately and monitors new changes in the CouchDB and picks new unique users that have coordinate data as part of their tweets and fetch additional historical tweets issued by the user using the User Timelines API from Twitter. The Python library Tweepy was utilized to interface with both Twitter streaming and search APIs. With this approach, around 80K+ tweets were collected via streaming and at least 2.4+ million tweets were collected from the user timelines. Figure below illustrates the data flow.



Each of the streaming harvesters was configured to filter different portion of Victoria bounding box which minimizes the chances of having duplicate tweets. Using bounding box would lead to retrieval of additional tweets from surrounding states, but this is addressed at a later stage via CouchDB views to extract only tweets associated with the local government areas of Victoria. Additionally, the tweet ID was used as the ID for the CouchDB document so that any duplicate tweet would directly get rejected by CouchDB. The harvesters will catch all duplicate error found, then continue to search for other tweet. Three different API keys and tokens were used by each instance to avoid getting blocked from Twitter.

To overcome the quota imposed by the Twitter API on the number of tweets pulled within a timeframe, two different approaches were taken. Firstly, was on the user timeline retrieval process; each API query would return around 3200 tweets. After the query, the process is put to sleep for a certain period, then it attempts to query the next user's timeline. In this way, the query retrieval is throttled based on the limit rate restriction of 15 API calls in within a 15 minutes frame. Secondly, certain restrictions were added in the live streaming harvester; in the event of any disconnection or exception from Twitter, a retry mechanism was put in place to reattempt the connection after waiting for a certain period before each attempt. With sufficient control and error handling in place, the harvesters would be constantly running in the background and pushing twitter data to CouchDB for analysis.

One challenge was faced with the harvester, after a few days of execution the harvester kept disconnecting in the evenings with a "Connection broken: IncompleteRead" exception. The initial thought was that it is due to a random network issue that would not occur frequently. However, the same issue kept recurring every night. After further investigation, we realized that the disconnections were done from the Twitter side. Based on Twitter documentation, one of the reasons that Twitter closes a streaming connection is because "the client reads data too slowly" (Twitter API Documentation, 2019). Our hypothesis is that although the insertion to CouchDB is fast, the streaming rate was still faster which leads to a backlog after around 20 hours. To address this issue we implemented a queueing and threading approach so that incoming tweets are queued and then the data is fetched from the queue to be pushed to CouchDB in a separate thread minimizing the likelihood of falling behind the streaming rate.

Map Visualization

We begin by describing what kind of map visualizations we want to achieve based on the selected scenario. This scenario is defined in the upcoming section of Twitter Analysis. After that, three tools/libraries for visualizing Maps with Javascript were considered. Following is a summary of these libraries and why we opted for the latter one:

1. **Mapbox**: Although it has the ability to produce layers on top of OpenStreetMap, it is not capable of producing dynamic Maps because the maps are built from a tool called **Mapbox Studio** instead of from a resource on runtime.
2. **ArcGIS**: This tool is more suitable for a geological cases, because the base map is capable of showing the detailed height/depth of a terrain. However, it is not needed in our case. On top of that, the API documentation doesn't show clear cases of showing layers on top of the existing ones.

3. **OpenLayers**: This library uses OpenStreetMap and layers which is easy to build on top with JavaScript. It has a lot of examples and although the API documentation is not the best, the community is huge, whether in GitHub or forums such as StackOverflow, making it easier to find related bugs/solutions if something is not available in the documentation.

After considering the strengths and weaknesses of each option, we decided to use OpenLayers, using its API documentation and various forums (mainly StackOverflow and GitHub issues) as the source of truths.

There are two challenges that had to be handled when using OpenLayers during this project. Firstly, the OpenLayers API documentation does not provide examples for some of the APIs. This causes the users guessing how to exactly use this API, e.g. there is an example on how to create a "Line" in a map, but it is created with a **string** instead of **Points**. This particular information had to be searched in other sources because the API documentation does not provide detailed example that suffices our use case. Second, if there are a lot of layers (or points) in the map, the browser will slow down heavily. This is a problem during the development because it not only makes it harder to progress quickly, it also hinders the actual interactivity that is supposed to happen. One workaround to this is to **limit the number of documents** fetched from CouchDB view (using **limit** query parameter), but in practice we do not use it in order to show the whole datasets.

Implementation Challenges

Setting the Ansible script for automation is not a hard task but it is prone to typos which we experienced quite often. Thus, it is essential to be vigilant while checking the Ansible documentation. Furthermore, we initially wanted to ship the CouchDB in a Docker container and configure the cluster across instances. However, after going through some experimentations and considering the benefits of using Docker (which will be explained in the next section), we decided to not use Docker and install the CouchDB directly in the VM.

Scalability of clustered CouchDB has been considered in the Ansible script. This is done by:

1. Checking whether CouchDB is installed inside the node.
    a. If it exists, It skips the CouchDB installation process.
    b. If it doesn't exist, it will install the CouchDB using the noninteractive mode with the given installation parameters.
2. Before the clustering part begins, the coordinator node will check the membership status at that time. The output of it will be saved into a runtime variable.
3. After the clustering part finishes, the coordinator node will repeat the previous action, which is again, checking the membership.
4. The coordinator node will decide whether it should send a "finish cluster" action to itself based on the equality of the previous state and the state after. If they are not equal, then it means a new node is added, and "finish cluster" action is needed.

In order to collect tweets, the group initially suggested to have multiple tweet harvesters running on multiple machines using various keywords and geofence filtering criteria to fetch online streaming data from Twitter. However, we found that collecting tweets based on keywords would limit the data analysis. Furthermore, collecting current live tweets alone

would not provide as many valuable insights to identify sin behaviour of humans compared to going with a more focused approach that drills deeper in to the tweets posted by a specific user to monitor his behavior and categorize the possible sin associated. For example, posting one aggressive tweet may not mean that the person is guilty with the wrath sin, such categorization is too generalized and judgemental.

Given all available resources (one group project plus one pt project for each member), we feel staging (development) environment is required, hopefully with the same capacity as given in group project quota. This staging environment will make team confidence enough to perform scalability (up and down) tests as well as experimenting to add more functions to the existing infrastructure (e.g. load balancer for couchdb cluster) to the existing cluster without the worries of bringing the production environment down.

Docker

One of the benefits of using Docker is isolation. For this project, we're building Virtual Machines from scratch and we define the operating system, dependencies, and softwares installed ourselves. Since from our architecture each VM instance does not run applications with conflicting dependencies, we think that using Docker will only add more complexity to the equation without considerable benefit.

Other benefit is the speed when shipping things. For example, if our application is containerized, the customers/users can just run `docker run ourapplication:latest` to try our software in their local machine without needing to install the dependencies such as Python, Node, etc. However, since we are not shipping this software to Open Source community or customers -- and this software is deployed with Ansible instead of other tools such as Docker Compose or Kubernetes, we think that using Docker is not necessary.

D. UniMelb Research Cloud (Nectar) Evaluation

About Nectar

Nectar is an IaaS cloud service built on top of OpenStack framework consists of data centers across Australia and mainly used for academic purposes. It has a dashboard interface to interact with the infrastructure options as well as a neat overview about the given quota for this project. A lot of functionalities can be accessed from this dashboard, but ultimately we only explored compute, volume and network functionalities.

Positive Features

One of the advantages of using Nectar is the dashboard. It lets the user download the OpenStack RC file which makes the authentication easier and standardized. On top of that, it is free to use for university students.

Main Limitations

The load time between pages in Nectar Dashboard is rather slow (~3-5 seconds). Other than that, users are not notified of the changes related to the Nectar content, such as images. Our team experienced the "image not found" error due to newly updated Ubuntu 18.04 image on May 10th.

E. AURIN Indicators

This research used data from the Australian Urban Research Infrastructure Network – AURIN (2019), in order to attempt to find correlations and compare some of the aspects of each of the identified sin with the official data from AURIN and gain insights on the relation between demographics, social, economics, health and lifestyle with the deadly sins. Furthermore, to support the greater goal of ranking cities of Victoria based on livable indicators it was crucial to find tangible evidence-based research behind our classification.

It was decided to base our investigation on the local government area level in Victoria (LGA) not only for the clarity of visualization by dividing the state to 79 areas, but also because each local sub division share the same administrative and political structures and most of city boundaries in Victoria are sit to be the same as the LGA boundaries.

Initially, identifying and extracting relevant data from AURIN was a challenge; we attempted to search for relevant datasets via AURIN portal, but faced with errors while retrieving or attempting to download the datasets. We tried to find alternative ways by utilizing the AURIN APIs that have similar implementation to GeoServer and uses the default Open Geospatial Consortium (OGC) web feature service. The APIs were helpful to extract some dataset, but it was a time-consuming process to manually search for a possible relevant dataset through the portal, get the dataset API id and then extract the features of the dataset through web service call as there was no option to do dataset keyword search via AURIN APIs and not all the datasets listed in the portal are available through the API.

Eventually, with the support of AURIN data team we were able to identify the cause of the error in the portal, that was due to the mismatch in the statistical area selected in our area selection and the statistical area of the dataset, alternatively used the Bounding Box selection that supports all geometries. This simplified the process of extracting AURIN data by downloading the required datasets as JSON files then uploading each dataset to the CouchDB using python script then map, aggregate and consolidate the relevant statistics with twitter sin analysis process.

The following table summarizes the relation between each sin, the corresponding data retrieved from AURIN to validate each sin and the possible variation on which the AURIN data may not have high correlation with the sins.

Sin	AURIN reference	Considerations
Sloth	As sloth is associated with laziness and lack of movement, it would be correlated with the proportion of people who sit for 7 hours or more per day. This dataset was based on a survey done by VicHealth in 2011	This dataset also includes siting hours while driving, working at a desk or computer, reading, watching television and playing computer games. Including working to this may distort the Sloth categorization as

		many people are bound to 9-5 disk jobs.
Greed	As greed is a selfish desire for wealth, it would be linked with act of gambling and correlated the total amount of money lost on electronic gaming machines per adult individual. This dataset was sourced from the Victorian Commission for Gambling and Liquor Regulation (VCGLR) between 2014-2015	The dataset describes the amount of money lost in games, however greed could also be linked with people who don't take risks and seek to save their fortune or spend in guaranteed investments which would not be covered in these statistics. Greed could also be linked with status and the average household income but we rather focused on the act of seeking extra money more than what is earned via work or position.
Wrath	As wrath is the extreme anger that could lead to bad temper or irrational aggressive behavior, it would be associated with the number of non-sexual offences. This dataset was sourced from Victoria police reported assault by offence type between 2008 - 2017	The dataset reports the assaults reported to police. However, many of the domestic violence cases would not be reported.
Lust	As Lust is driven by strong sexual desire, it would be correlated with the number of sexual offences, stalking and harassment. This dataset was sourced from Victoria police reported assault by offence type between 2008 - 2017	Similar to wrath, many cases of sexual assaults goes unreported due to cultural constraints. Furthermore, some facts suggest that sexual assault is more about power and control than being motivated by sexual gratification (American Journal of Psychiatry)
Gluttony	The study by PHIDU that included Adults Health Risk Factor Estimates between 2014 - 2015 was used to get the estimated number of people aged 18 years over whose health was at risk due to overweight or obesity	This dataset covers only people categorized with overweight/obesity, which excludes people with Anorexia Nervosa disorder. This group would be underweight, but should be part of this group due to their Binge-eating and purging behavior.
Pride	As pride is linked with status and the individual love for power, we based the correlation with the total weekly family income. This dataset was sourced from	The dataset is purely about income, thus, not any contain negative attribute. This may return low

	Australia Government ABS Census 2011.	correlation to Pride if the pride sin is not only related to wealthy people.
Envy	Estimates of the prevalence of homelessness on Census night 2016, derived from the Census of Population and Housing using the Australian Bureau of Statistics (ABS) definition of homelessness. Prevalence is an estimate of how many people experienced homelessness at a particular point-in-time.	This could be a bias correlation between homelessness and envy

F. Twitter Analysis

The main task in this report is to detect and explore the seven deadly sins in social media. The characteristic of the seven deadly sins as mentioned is over-obsession of something. Gluttony is associated with food obsession, greed for money, lust for sex, wrath for anger, pride for oneself, sloth for laziness, and envy for jealousy. Defining the scenario to detect each of these sins is the main challenge since there is no sufficient past work or literature that explores such behaviors in social media. Thus, for this part only, several approaches were trialed before we came to the final decision.

Once the sins are detected, as mentioned in the introduction, we propose to extend the scenario not only to detect sins in social media but also to use this information for a greater purpose. The ultimate goal is to help people discover the best place to live in Victoria or the best liveable city. Finding the best one is equivalent to avoiding the worse one, or in term of scores: sorting the rank of unlivable cities in reverse order. The reason why unlivable cities scenario is chosen is that all the features that form the ranking are based on the seven deadly sins in Twitter and negative events, such as crime and offenses gathered from AURIN. These negative factors are then aggregated to form the unliveable score for each area.

Having the unlivable score might be enough to fulfill the scenario. However, in reality, people will not be satisfied by only looking at the scores and the statistics. Therefore, we also embrace more information about the characteristics of each city in Victoria, such as the connection between cities and typical people or environment of each city. Another used metrics in this project are the minimum distance or time from one city to another, most common trending topics in each city, and the overall sentiment of tweets in a particular area. This information will enhance people's understanding of how one city could be different from the others and what are the trade-offs.

In the process of providing all above insights, we notice that there is some other valuable information that can be extracted. This information might not be directly related to support information about the liveable cities, but it might be useful to open one's eye about typical characteristics of the people who live in Victoria, such as how likely they travel to destinations outside Victoria or outside Australia, what is the most common overseas travel

destination that can be reached in a day and most likely bring happiness to the travellers, and how likely people in Victoria vary according to their tweet language or their place of origin.

Data Preparation

As mentioned in the harvesting part, the first harvesting is based on Victoria bounding box and the second harvesting is based on timelines of all unique users fetched from the first harvested records. From this user timelines, a View is made to take tweets that coordinates are not null.

We tried timeline harvesting in two ways, the first one is using the timeline API in compatibility mode and the second one is using the extended mode. The benefits of using the former one is that it is fast, all the past history of tweets from each unique user can be collected in short duration. However, for each user twitter limits the result to only take up to 200 records. While the later one allows to take more tweet history which can reach above 3200 tweets per user.

There are trade-offs between number of unique users and number of tweets in these two harvesting approaches. At the same time when these databases are used for the analytics, the former one returns 1,024 unique users from 66,196 tweet records with coordinates, while the latter only returns 624 unique users from 288,709 tweet records with coordinates. Even though having more unique users in the dataset will contribute to more information diversity, in this project we choose the latter approach as the data source for the analysis because it is done on tweet level that later will be aggregated to area level.

Data	Number of Records
Tweet Harvested based on Victoria bounding box	53,370
Unique Users from Victoria database	1,383
Timeline harvesting in compatibility mode	
Tweet Timelines based on Unique users	205,066
Tweets with coordinates in User Timelines	66,196
Total Unique Users	1,042
Timeline harvesting in extended mode	
Tweet Timelines based on Unique users	1,504,862
Tweets with coordinates in User Timelines	288,709
Total Unique Users	624

Data Pre-Processing

The analysis process starts with data pre-processing including tweet text normalization and feature extraction. Feature extraction is done not only on tweet text but also from other tweet attributes.

Text Normalisation:

1. Expand word contraction for “I’m” to capture the pronoun “I” to be used as one of pride detection
2. Remove non-ASCII characters as they do not contributed to the analysis
3. Remove URLs and User Mentions
4. Remove unnecessary punctuations and any special characters
5. Do lemmatization and lower casing to each word

This text normalization is needed to make the tweet text ready for keyword matching and text classification. Since every word is clean from unnecessary characters, the chance of not getting keyword match in tweet text is minimized. The normalization process is run after storing the hashtag in tweet text for the corresponding area. This hashtag is to accomplish the extended purpose to provide information of typical area’s characteristics. Other feature extraction from tweet text to support this purpose is the sentiment polarity. It will be later used to describe how likely people sentiment in each area.

Based on the background approach to formulate sins detection, it is noted that keyword matching is not enough to describe some sins. Thus, we derive other features to be used in sins filtering and support for deeper analysis. The following features are extracted from the coordinate metadata and the time when the tweet is posted. The usage of each feature with reference to the analysis is explained in the results section.

Coordinate related features:

1. Tweet end coordinate, which is fetched from the next tweet of the same user on the same day. For example, if a user tweets 3 times on one day, the first two tweets will have an ‘end coordinate’ feature added based on the coordinate of the next tweet while for the last tweet the ‘end coordinate’ will be null.
2. The country where the tweet was posted and the country of the next tweet by the same user on the same day.
3. The city where the tweet was posted and city of the next tweet by the same user on the same day. The exact city name is set only if the coordinate is inside Victoria polygon, otherwise this is marked as ‘Outside Victoria’
4. Spatial distance based on the coordinate when the tweet was posted and the ‘end coordinate’ retrieved from the next tweet if posted on the same day.
5. Closest street when the tweet was posted. Measured only for coordinates inside Victoria because the streets are limited to Melbourne. For other cities outside Melbourne, the corresponding distance to the closest street will be high.

Time related features:

1. Time difference based on the time when the tweet was posted and the time of the next tweet.
2. Time range to categorize time when the tweet posted into: Early morning (5 - 8 am), Morning (9 - 10 am), Late Morning (11 am), Early Afternoon (12 - 3 pm), Late Afternoon (4 - 5 pm), Early Evening (6 - 7 pm), Evening (8 - 9 pm), Night (10 pm - 4 am).

MapReduce is indispensable in the whole analytical process for both preparing input for the analysis and preparing input for the visualisation. The analysis only read from the timelines harvested data then produce a database that contain the sin flags and those features from the tweets that has the coordinates not null, thus, the LGA city can be detected. From this database, then we make some views as the input for the visualization with mainly use *_sum*, *_count*, and *_stats* reduce function.

Sins Detection

Following is a description of the sin detection approach used to identify each sin based on either tweet text or metadata

Sin		Detection Rules
1 Sloth	:	Tweets of the same user that is not moving on a day (by measuring the distance between consequent tweet coordinates) or users who do not post in early mornings. It is presumed that these users are too lazy to be awake early in the morning and they keep tweeting without much movement.
2 Greed	:	In contrast to Sloth, Tweets of the same user on the same day with distance greater than 3000 km apart are classified as Greed because it is assumed that these users possess a large amount of money to travel frequently and they show it by posting new tweets in the new destination.
3 Wrath	:	Tweet of angry/aggressive people are detected by their tweet texts. Two steps of classification are applied, the first one is a machine learning model using a Naive Bayes classifier that was trained on a corpus of "Twitter hate speech" downloaded from Kaggle and contains 16,130 tweets that were manually categorized as either aggressive or not[1]. The second approach is a Lexicon based sentiment analysis. A tweet is flagged as Wrath if it is classified both as angry tweet based on the first classifier and with negative polarity sentiment based on the second one.
4 Lust	:	Tweet that contains keywords reflecting sexual desire by users with more than a thousand statuses count. The keywords are: 'sex', 'sexual', 'passion', 'sexy', 'intimate', 'lust', 'sensual', 'passionate', 'loves', 'kiss', 'girlfriend', 'fucking', 'fuckers', 'fuck', 'bitches', 'loveshis', 'shave', 'my gf', and 'mygf'. It is assumed that Lust people are mostly male who look for fame and attention, thus most likely have thousands of twitter posts.
5 Gluttony	:	Tweet that contain keywords reflecting overindulgence in food. We evaluate this sin deeper than the others, the keywords and the explanation how the rules perform is in Background Process part.
6 Pride	:	Tweet that contains pronoun "I", a bigram of Adjective and Noun, include at least one hashtag, and the user have favorites count less than friends count. It is presumed that pride people think of themselves, most likely to show a 'beautiful stuff', and do not really care about their friends' tweets.

[1] https://www.kaggle.com/vkrahul/twitter-hate-speech

7 Envy	:	Tweet that contain keywords reflecting jealousy or eagerness and the user has friends count twofold followers count. The keywords are: 'wish', 'need', 'want', 'desire', 'jealous', 'eager', and 'look'. It is assumed that Envy people keep track of other people posts thus they have more friends than followers.

These rules are determined after several steps of analysis that is explained in Background Process section. The discussion of Sins in Result and Visualization will refer to this definition. Note that in terms of global definition, some sins might have a close definition to Pride as people who like showing of for example would instantly post after an activity, includes travel or food. There will be an evaluation of this concern in Correlation between Sins and AURIN data part in Result and Visualization section.

Detected Sins

The following texts are a sample of tweet texts for each sin. This sample is chosen to describe that our approach can capture what is intended to be flagged as the sins. However, the accuracy cannot be measured directly, it can be approximated by comparing the results with the factual data from AURIN. The discussion of this will be in the Result and Visualization section below.

Sins	Sample Tweet Texts
Sloth	'Nup I'm going nowhere! Too rainy and I don't like thunder! You check the cows yourself!! #farmdogs #brains 😂 @ Bira'
Greed	'Waiting to board my flight from Dubai to Sydney... Getting there... eventually! (@ Emirates Business Class Lounge'
Wrath	'Twighlight over #wilsonsprom cows not happy about daylight savings losing an hours sleep and worried their coats'
Lust	'MY LOVE __sarahlouise___xo 😍💕💋 This was the best weekend! MPDA, naughty dresses, and lots of booze! 😂'
Gluttony	'This is not a drill guys!! I think I've found a place comparable to the incredible pizza and pasta I had in Italy 😍'
Pride	'The nifty fifty lens, 50mm. I hope I can grab some more autumnal colours with this. #darylhunt #mansfieldmtbuller'
Envy	"I'm a bit self conscious about how I look, which makes me wonder what the heck I'm doing, putting myself front'

Unliveable Score

The Unliveable score is measured based on total number of each sin and its corresponding AURIN data per LGA. All the features are normalised by dividing with the maximum. The sins are later weighted by their correlation with the AURIN data. Unliveable score is the total number of these normalized features.

Background Process

While the infrastructure deployment and harvesting was in progress at early stages of the project, a preliminary analysis was conducted with pre-harvested Twitter data for the city of

Melbourne from January 2015 to December 2016. We tried several different approaches before deciding the best one that is used for the final analysis. The following approaches briefly explains the past trials.

1. Rough approach

The first approach was to specify a simple keyword based matching rules, for Sloth, Greed, Gluttony, and Lust, then expand the keywords using Thesaurus before match to the tweet text. The reason why using Thesaurus rather than Wordnet is because Thesaurus provides more casual words than Wordnet in term of synonyms. Whilst, for Wrath was classified based on negative sentiment polarity using Lexicon-based approach and for pride based on Part-of-Speech (POS) tagging. It is assumed that Pride people will review about a product or service thus it has a form such as 'the beautiful bag' or tagged as 'Determiner (DT) - Adjective (JJ) - Noun (N)'.

Keywords to Thesaurus	
Sloth	'lazy'
Greed	'money', 'rich', 'wealth'
Gluttony	'food', 'meal'
Lust	'sex', 'sexual', 'desire', 'passion'
Envy	'wish', 'need', 'want'

The result of this approach was not satisfying as there are many tweets incorrectly classified into a certain sin. It is suspected that this is due to rough filtering based on Thesaurus. Accordingly, the next trial was to evaluate a sin to get better understanding about which keyword should be used.

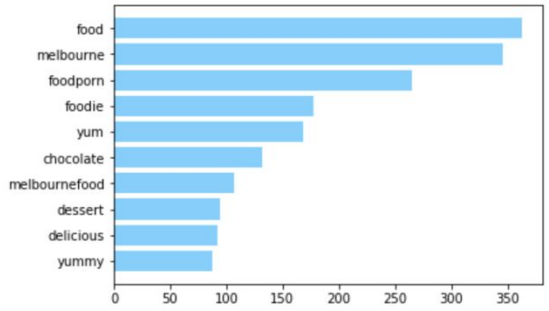
2. Evaluate a sin - Gluttony

The easiest sin to be deeply evaluated is Gluttony, since there are many resources in the internet that mentioned top popular hashtag related to food. Thus, at the beginning, Gluttony was detected based on these popular hashtags.

#foodie, #foodporn, #foodgasm, #nom, #nomnom, #nomnomnom, #food, #eatingfortheinsta, #chefmode, #hungry, #cleaneating, #yummy, #instafood, #delicious, #realfood, #food, #yum, #foodgram

Having compared the result from the first approach, the above keywords produce more tweets that represent food obsession. In order to prove the assumption, the next evaluation was to elaborate more on the tweet texts. According to the most common hashtags, we found other hashtags that are not the keywords tend to appear together with the keyword hashtags.

Top 10 Most Common Hashtags

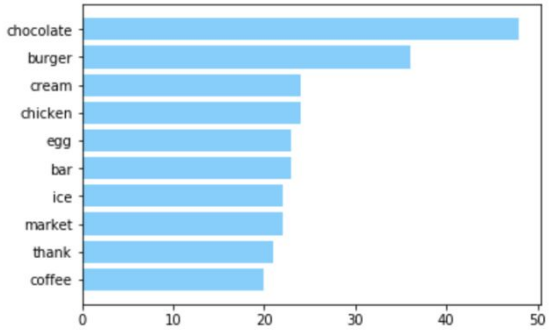


Therefore, we added these common hashtags as the new keywords to detect Gluttony sin.

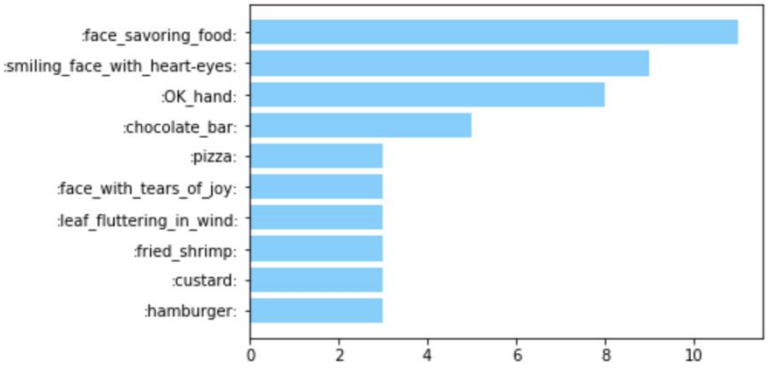
#pizza, #icecream, #chocolate, #seafood, #dessert, #melbournefood, #burger, #burgers, #melbourneeats, #foodstagram, #cleaneatin, #delish

The interesting fact was, as we detected all the nouns in tweets that are tagged as Gluttony and remove the most common nouns, the remaining nouns were various different food types. Moreover, if all the emoticons are extracted, we found that the most common emoticons in Gluttony tweet are related to people craving for food. Thus, this result convince us that this approach is more effective than the previous one.

Most common nouns

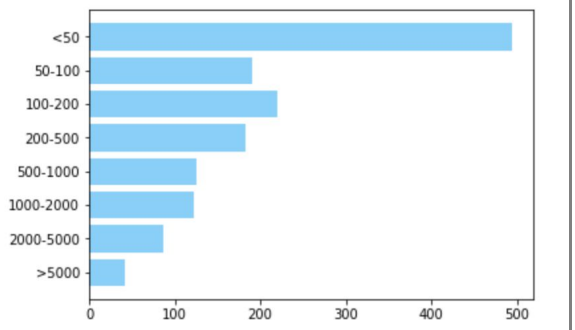


Most common emoticons

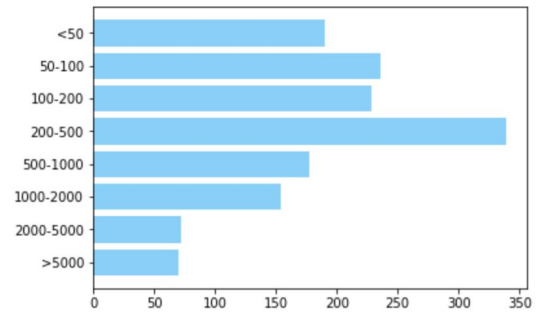


Furthermore, we tried to evaluate the Gluttony sin based on user profiles in terms of their favorites count, followers count, friends count, statuses count, the time range when the tweet is posted, and the time zone when the user made their Twitter account. As can be seen in the following charts, Gluttony tweets are mostly posted at dinner time by users with low favorites count, moderate followers count, and high statuses count. The logical relation between these statistics is Gluttony people seem to be highly active Twitter users who often tweet but rarely like others' posts.

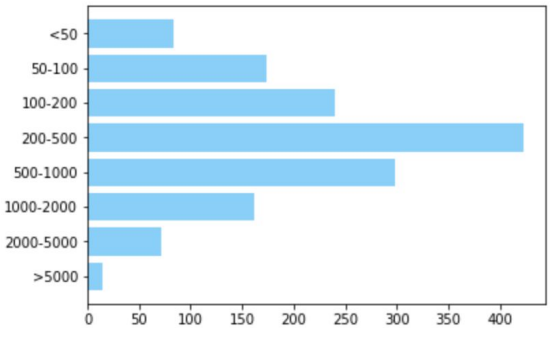
Distribution based on Favorites Count



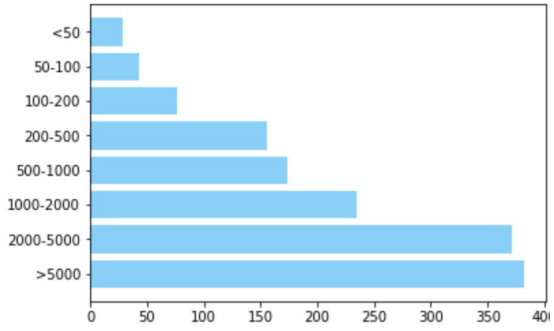
Distribution based on Followers Count



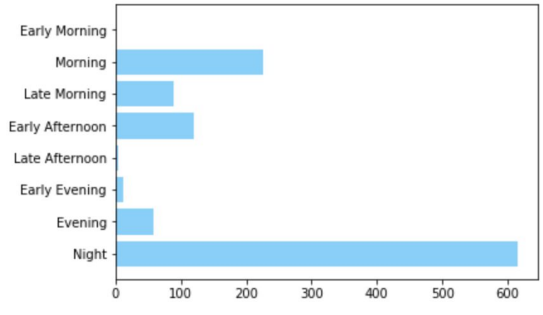
Distribution based on Friends Count



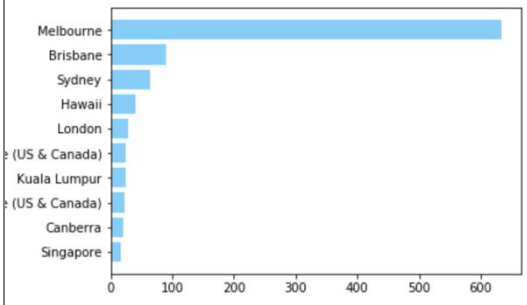
Distribution based on Statuses Count



Distribution based on Time Posting



Distribution based on User Time Zone



As proven by the previous facts that the hashtag keywords are sufficient enough to capture tweet mentioning food possession, the logical relation with user profiles does not need to be added in detection rules. However, this logical relation has given a hint to enhance other sins detection. Every sin can be associated with user profile. The assumptions of Lust, Pride, and Envy that mentioned in the final approach for Sins detection above is determined accordingly.

The fact that a number of Twitter users in Melbourne made their Twitter account in other countries shows the diversity in Australia. As can be seen in the right-bottom chart, outside Australia cities, the time zones are from developed countries. This fact stimulates an idea to measure the behavior of Twitter users based on their movements between tweets on the same day. Users who travel in long distance in a day and post a new tweet in their next destination are more likely to be wealthy people. It is worth notifying that sins detection do not necessarily to be based on tweet texts, but it can also be based on this kind of attitudes. As wealthy associated with money, thus, Greed sin can be reflected by this characteristic. On the contrary, people who post multiple tweets in the same areas on the same day can be categorized as Sloth.

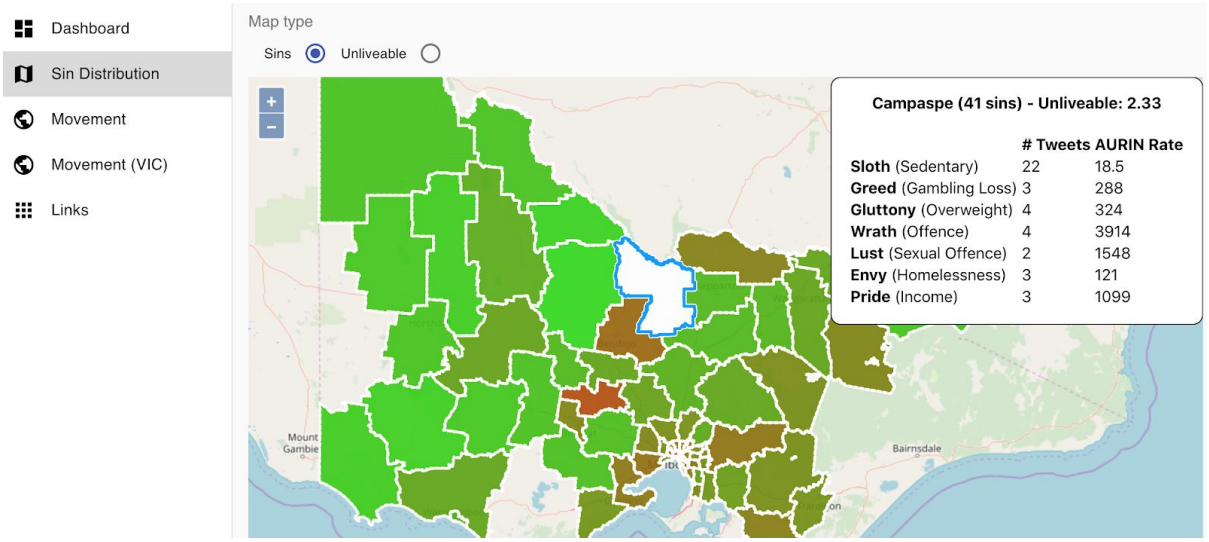
Challenges

Even though the idea of sins detection is clear, along the process we need to adjust some parameters according to the results. In Sloth sin for instance, we found out later on that there is an anomaly if using zero distance as the only criteria; because the majority of tweets are posted in early mornings. Since Sloth sin is associated with laziness, we should exclude any tweet posted in early morning for this sin. While in Wrath, after applying Naïve Bayes classifier there is a considerable portion of tweets classified as wrath, thus sentiment analysis is added as the second filtering to only consider tweets with negative polarity score. Tweet text normalization process also was determined after several trials, there are some standard steps that are dropped from model at a later stage, such as spelling correction and language standardization for non-English tweet text. This is due to the performance of spelling correction and translation that did not return convincing results, it rather changed the meaning of the tweet. Overall, handling tweet text is not an easy task, and even more difficult to classify it as social media sins. Human annotated dataset is needed to have high accuracy results.

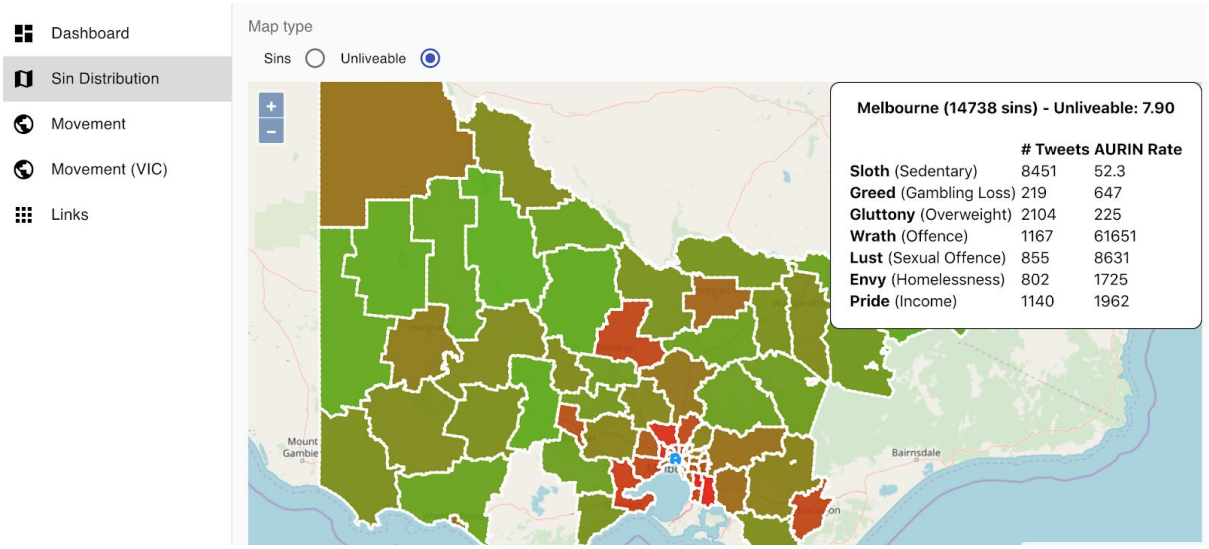
Result and Visualization

I. Area Map Visualization based on total sins and unliveable score

This is the primary visualization of this project as it summarizes all the main findings; seven deadly sins in social media, their AURIN counterparts, and the unliveable score. There are two choices to determine the gradient color of each area, it can be based on total number of sins or based on unliveable score. Furthermore, once the cursor point to an area, there is a hover shows all the details of Seven Deadly Sins, the AURIN statistics, and the Unliveable score for that area. It can be seen that the distribution of sins is closer to Melbourne city. Unliveable score share similar distribution but more vary to throughout Victoria areas. Recall the main goal of the scenario, through this map people can easily point to an area and get all the related information about that area. The interesting fact is, Melbourne which hold the label as the most liveable city in the world for seven years in a row up to 2018 is the least liveable compared to other cities in Victoria. It has small area, dense population, high number of offences, and high intensity of seven deadly sins (limited to the definition in this project).



Area map based on total number of sins, darker colour means higher frequency



Area map based on the unliveable score, darker colour means higher score

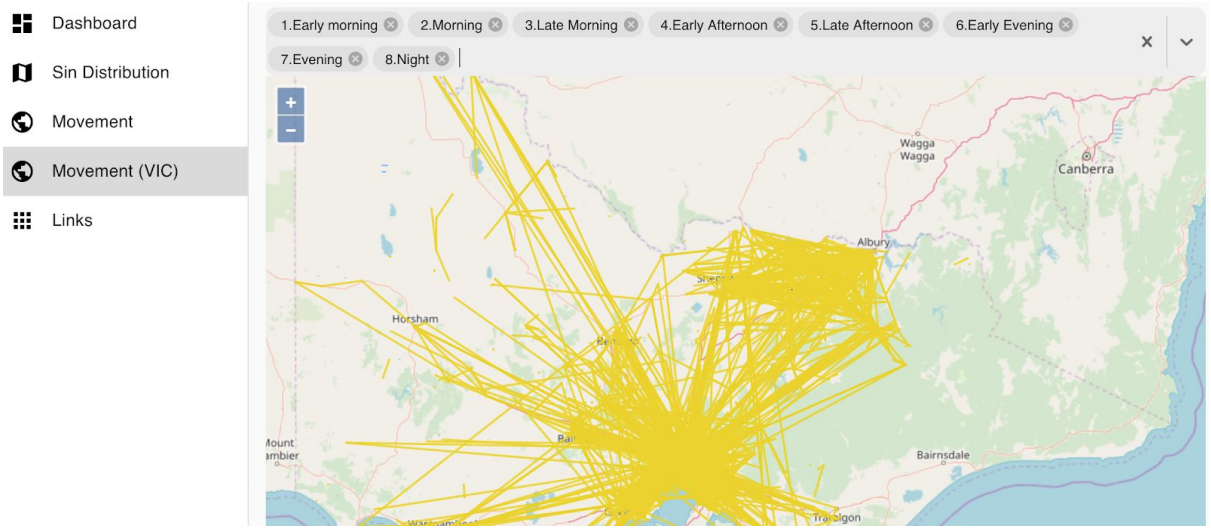
II. Point-to-Point Map Visualization for connected tweets from Victoria to anywhere

As mentioned before, we extract the 'end coordinate' based on multiple tweet of the same user on the same day. This visualization shows what we define as 'Greed' sin which comes from people who have money to travel and tweet directly after they arrived in the new destination. Apart from Greed, this map also stimulates idea to extract the most common travel destinations from people in Victoria along with extra information to know which ones are the closest according to distance between coordinates and which ones are the fastest to reach according to time difference between the tweets.

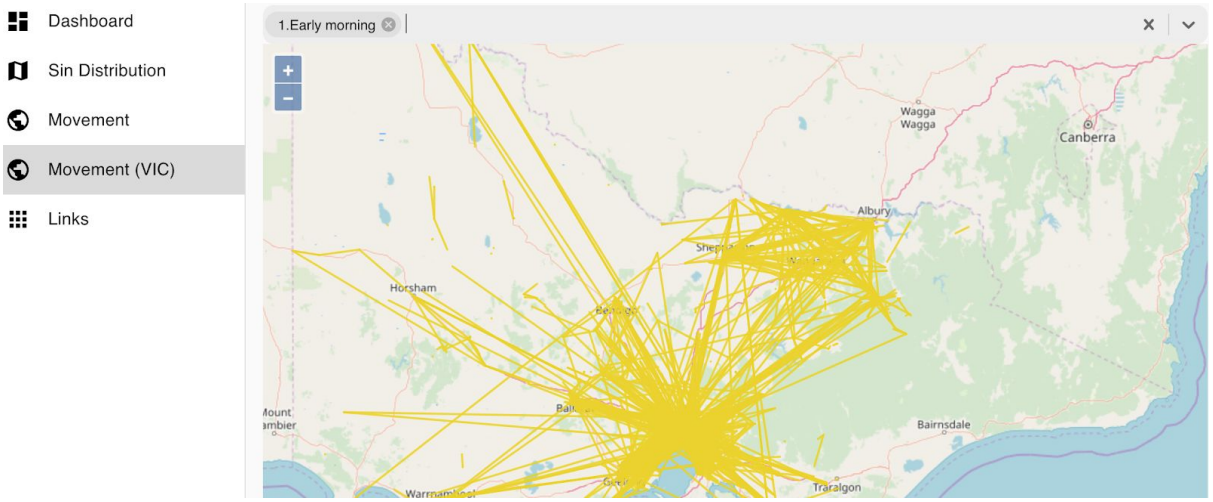
Connection between multiple tweets from the same user on the same day starting from Victoria

III. Point-to-Point Map Visualization for connected tweets inside Victoria

This map has the same function as the previous map, but specifically to show the distribution inside Victoria. It also has filter function based on time when the tweet in the starting point is posted, therefore we can see how likely people move across time range. It is obvious the majority of Greed sin is in Melbourne since most lines comes from this capital city. Similar as the previous one, according to this map there are some areas in Victoria that people used to travel to, thus, gives idea to extract the most common travel destination inside Victoria.



Connection between multiple tweets from the same user on the same day inside Victoria

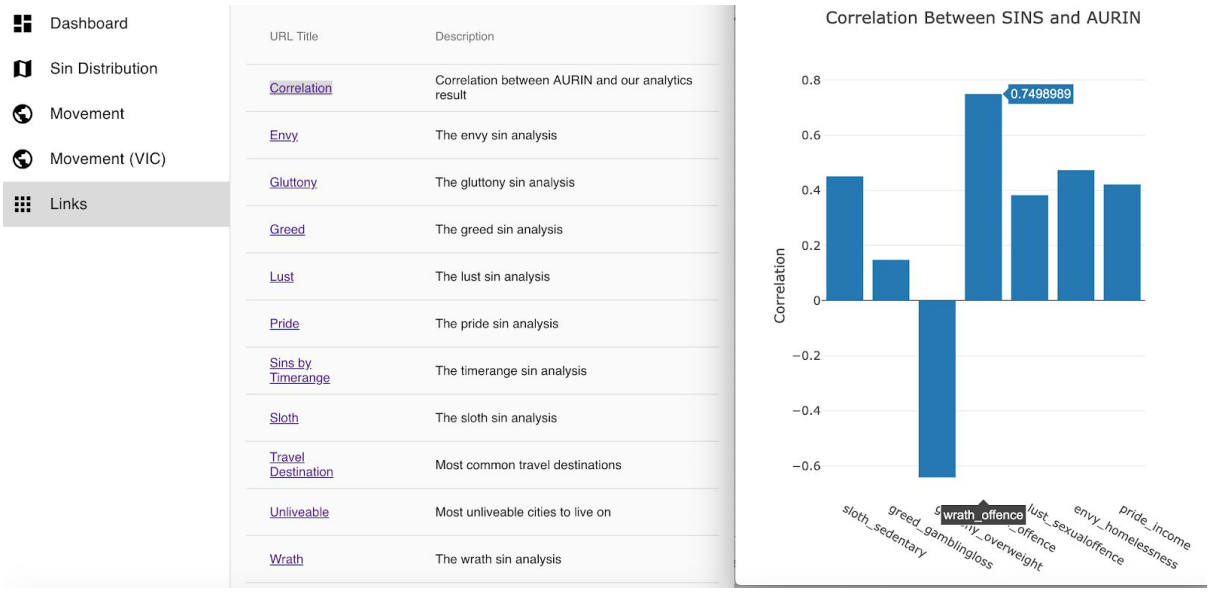


Connection map inside Victoria in early morning

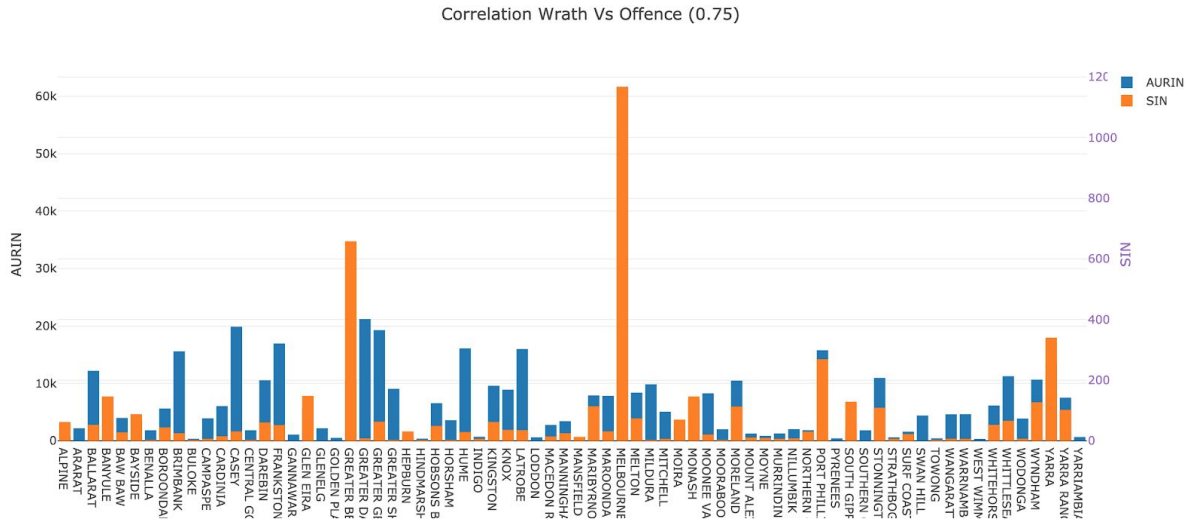
IV. Correlation between Sins and AURIN data

All the analysis results are provided as HTML links in this menu that defaults to open in a new tab. There are links for overall correlation between sins and data from AURIN and the distribution of the statistics in all areas for each sin. As can be seen in the following charts, the highest correlation is for Wrath and Offences which accounts for 0.75, if the data is filtered only take tweets that has 'end coordinate', it will be 0.82. Since Wrath is the only sin that using annotated corpus to detect angry tweets and using two types of classification, the result is as expected. There is a tendency that in areas where there are many tweets tagged as Wrath are most likely high in crimes.

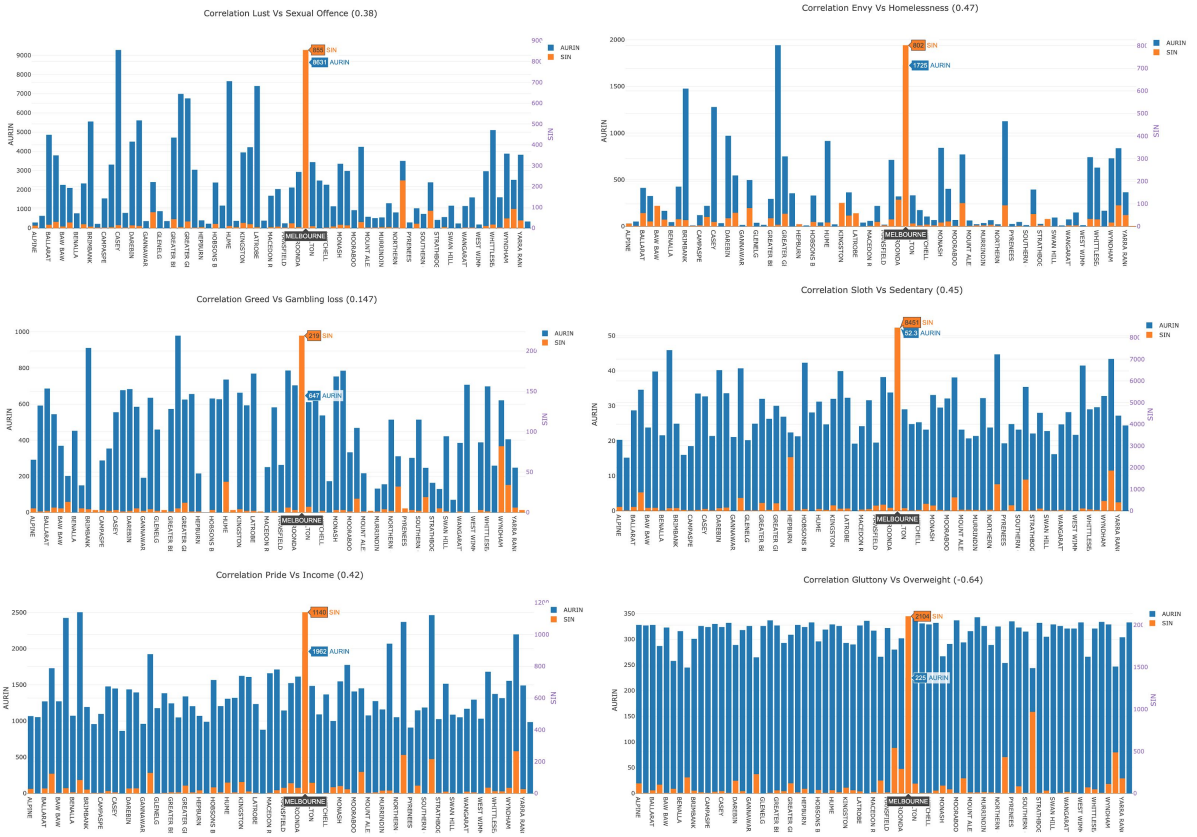
The next highest correlation and also the most surprising one is shown by Gluttony and Overweight which account for -0.64. Since the Gluttony rules has been evaluated deeply in the beginning, it can be said that people who tweet about food are not necessarily overweight. In fact, there is a possibility of reverse behavior, people who are overweight do not really like to post about food. As we tried to find the quantitative explanation for this, it is found that if we do Agglomerative Clustering based on number of sins for all unique users and tried to plot between two sins, Gluttony and Pride seem overlapping in one cluster. They also have 0.94 correlation rate. It shows that Gluttony is intertwined with Pride, thus not positively correlated with overweight. While for other sins, the correlations are below 0.5, the least one is Greed and Gambling Loss. It is later noticed that Gambling Loss is not associated with Greed, similar to Gluttony, it is highly correlated with Pride (0.91) and has higher correlation with income (0.3).



Overall correlation of sin and its corresponding AURIN indicator



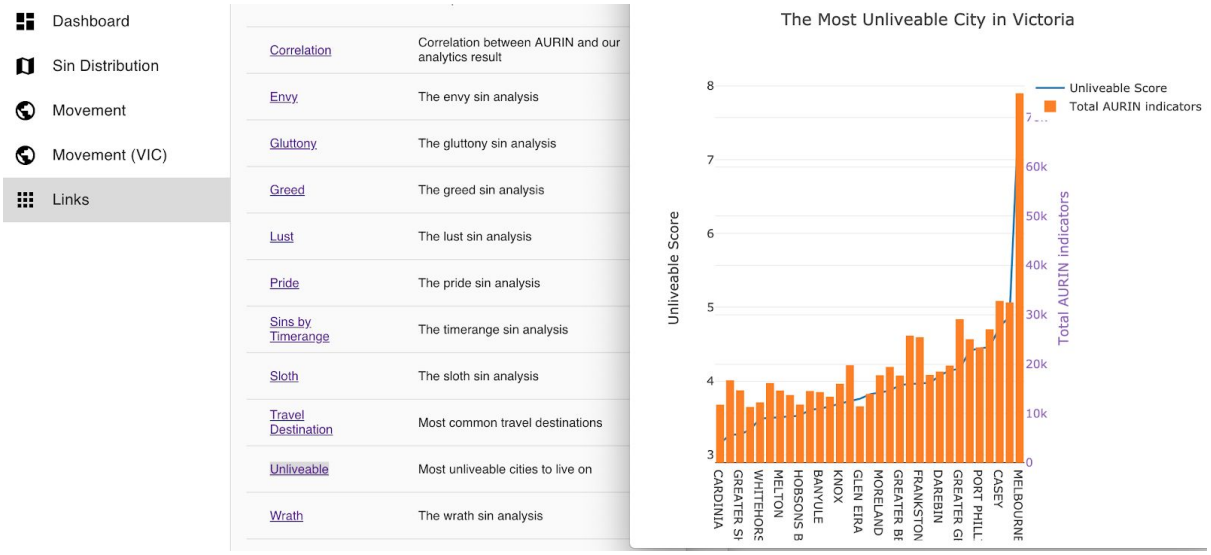
Distribution of Wrath and Offences across Victoria cities and their correlation



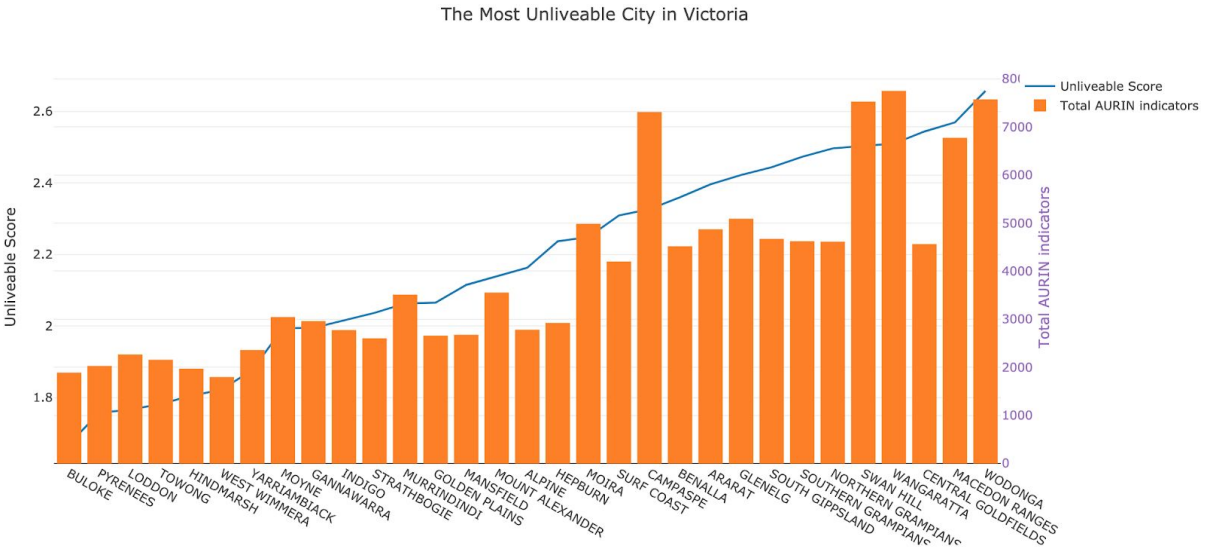
Distribution of Sins and AURIN indicators across Victoria cities and their correlation

V. The Most Unliveable City in Victoria

This visualization is for our extended scenario to provide insight about the most unliveable city in Victoria to suggest the most liveable city based on reverse order. It can be found as link in the links menu. As noticed in the map that Melbourne is the least liveable city and it is shown in the following charts that Melbourne is in the top rank. The five least unliveable cities are Buloke, Pyrenees, Loddon, Towong, and Hindmarsh.



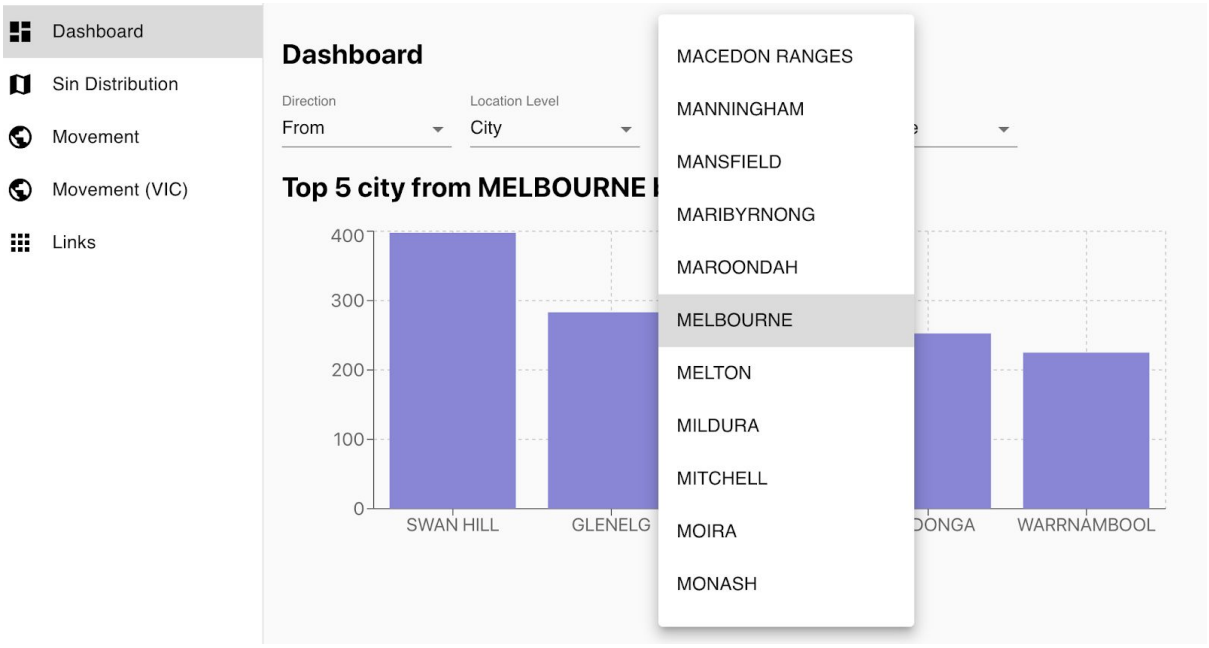
The top unliveable cities in Victoria based on unliveable score



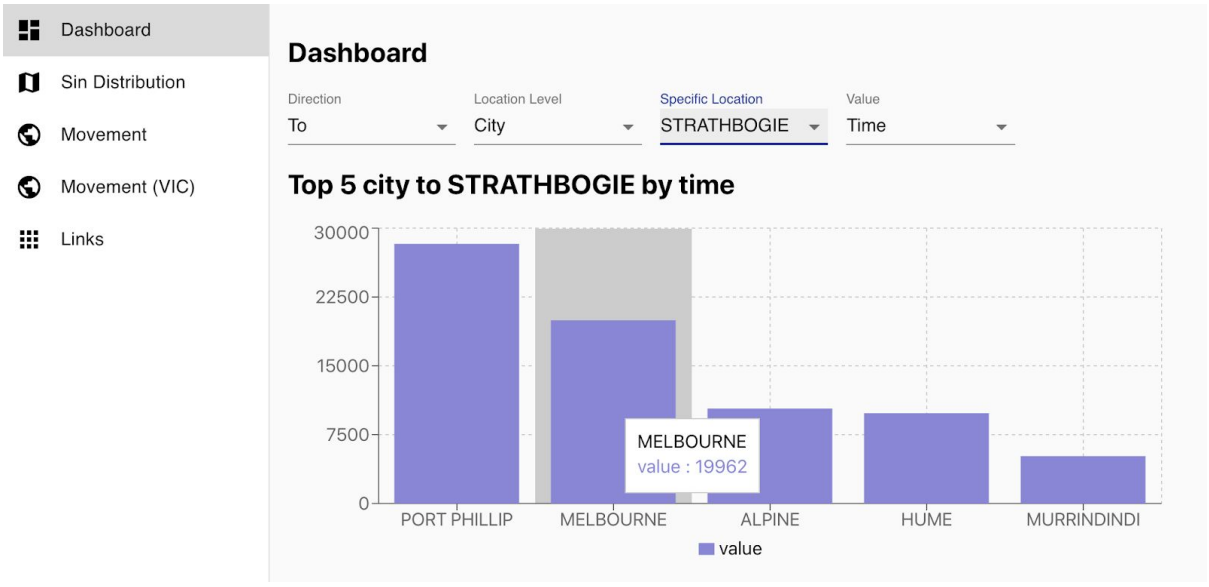
The least unliveable cities in Victoria based on unliveable score

VI. City Characteristics based on Distance and Time Difference

As explained earlier in the introduction of analysis part, the insight is to not stop only just to suggest cities but also to provide comparison between their characteristics apart from sins and AURIN attributes. Therefore, there is a dashboard provided for user experience to compare the minimum distance (in km), minimum time difference (in seconds), and total sins from or to a city. All these statistics are extracted from Twitter data. For example, if people work in Melbourne and choose to live in Strathbogie, they can click direction “To” Strathbogie and choose value “Time” to know from which city is the fastest to reach this city or otherwise “From” Strathbogie which cities is the fastest to reach. Furthermore, a hidden travel destination can also be extracted by choosing “From” Melbourne to the farthest “Distance”, as shown below it is Swan Hill and, indeed, there are some tourist spots in this city.



Dashboard showing distance from Melbourne to other cities



Dashboard showing time difference from other cities to Strathbogie

VII. City Characteristics Based on Common Hashtags

Before the tweet text is normalized, all the hashtags are stored based on area. This is done to explore the most common Twitter topic for all areas in Victoria. It is assumed this information can describe typical people's lives in a particular area. The following wordclouds are generated from the top 40 common hashtags in four cities. Moyne is the 65th, Murrindindi is the 61th, Port Phillip and Melbourne are the top 5 of unliveable cities in Victoria. It is noticeable that Moyne and Murrindindi are peaceful rural area while Port Phillip and Melbourne are busy big cities. This information is provided to help people decide the

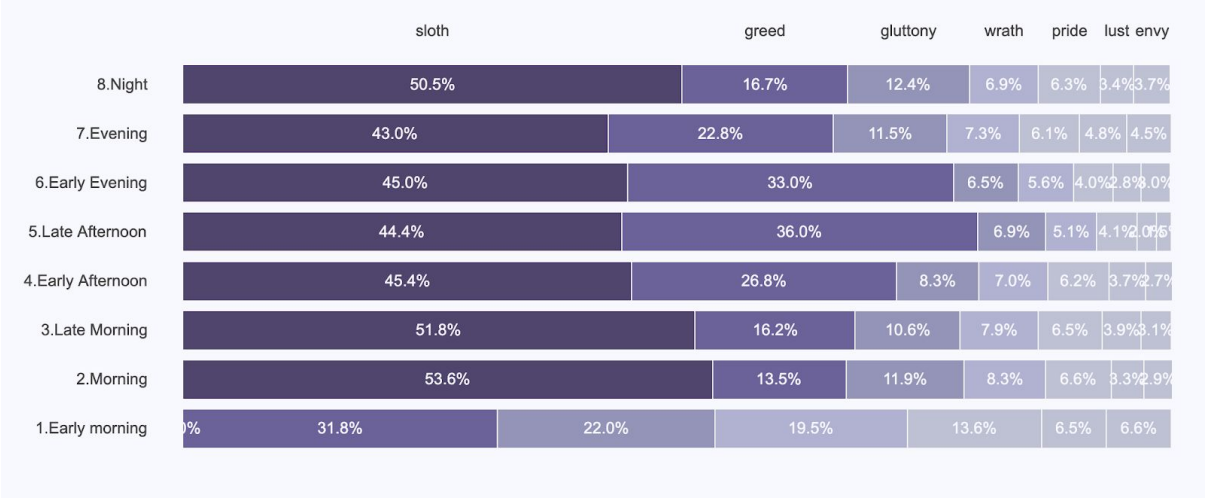
most liveable city not only based on score but also based on their preferences to the area characteristics.

Moyne	Murrindindi
	
Port Phillip	**Melboune**
	

Top 40 most common hashtags in tweets posted in the areas

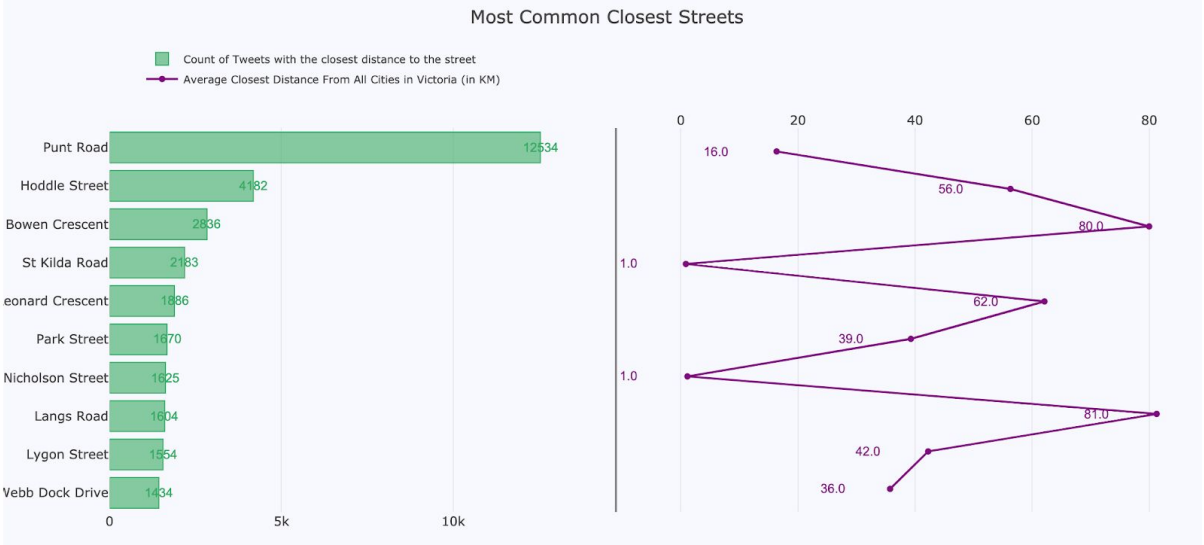
VIII. Other Valuable Information

The previous results are enough to fulfil all the goals in this project but insights are not limited to the those. The following chart shows the distribution of sins based on time range. Through this chart, it is known that Sloth and Greed are the majority of sins across time. The interesting fact is the only time range that has the most proportional percentage of all sins, despite the Sloth, is early morning. Thus, it is more likely that this is the most crucial time for people to be trapped in social media sins.

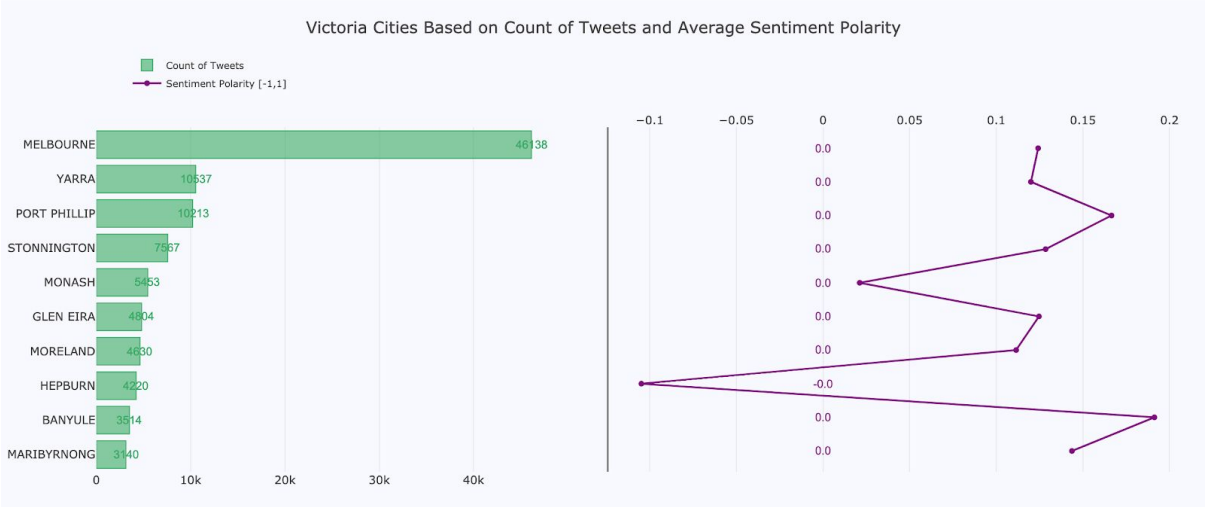


Distribution of sins by time range

Another interesting information that we extracted are the most common closest streets, cities, and even countries based on tweet counts or based on the minimum distance to reach and based its sentiment polarity as visualized in the following charts. The street here is limited to Melbourne street, thus, it is measured as closest street with corresponding distance. It is worth noted that from all tweets across Australia, the majority are close to Punt Road with average distance 16km. Thus if one wants to live in the city that has flexible access to other cities in Victoria in reasonable distance, Punt Road could be a choice. St. Kilda road and Nicholson Street could be a choice for one who mainly move around Melbourne. Even though Melbourne has the highest unliveable score, in terms of sentiment polarity, the tweets posted from this city in average are in positive manner. Hepburn, instead, is a city where the overall sentiment from tweets posted in this area is negative. These information enhance one's knowledge to compare which city matches their constraints. Furthermore, even though it is not related to the main goal, knowing most common country destination from tweets starting in Victoria could reflect typical choices of people in Victoria for overseas travelling.



Most common closest city streets from other cities in Victoria and their average closest distance



Most common cities by frequency of tweets and their average sentiment polarity

Most common countries visited from Victoria and their minimum distance

Conclusion

In conclusion, this project has successfully created a system to detect seven deadly sins in social media for Victoria cities and provide insights about the most liveable city in Victoria. This system is built under Nectar cloud using CouchDB cluster and the map visualization is built using OpenLayers. Despite all the challenges that are encountered during the implementation, the system managed to run properly. All the deployment process from setting up the instances in the cloud up to creating CouchDB cluster, deploying and running harvesters and move other related scripts and apps for analysis is done automatically using Ansible. Scalability of CouchDB cluster is also considered in the Ansible script. We use Twitter streaming and timeline API as the main data source for the analysis and use some indicators from AURIN to validate the results. We utilize CouchDB RESTful API gateway to connect the harvested data, analysis process, and map visualization. MapReduce is highly used along the process.

The main challenge in analysis is to define the rules to detect the seven deadly sins in social media. The final approach is to combine both keyword matching, text classification, and user profiling. According to results, we found that the Wrath sin is highly correlated with number of offences thus the rules is convinced. We also found that some sins are intertwined with Pride sin, thus, Pride is more likely to be the root of the sin. This is in line with the opinion from CS Lewis (2001) that said pride is the central evil, far more wicked than anger, greed, etc. As we extend the purpose of this project not only to detect sins but also to provide insights about the most convenient city to live in Victoria, we provide some useful information such as the city rank based on unliveable score, most common topic in each city based on hashtags, the nearest streets, cities, even countries to be reached from or to particular city in Victoria. The highlight of this extended scenario is that Melbourne is, unfortunately, the least liveable

References

Ansible Documentation - Ansible Documentation. (2019). Retrieved from https://docs.ansible.com/ansible/latest/.

Apache/couchdb-pkg. (2019). Retrieved from https://github.com/apache/couchdb-pkg/blob/master/debian/README.Debian

Aurin (2019). Retrieved from https://aurin.org.au/about/the-aurin-journey/

Rape: power, anger, and sexuality. (1977). American Journal of Psychiatry, 134(11), 1239–1243. https://doi.org/10.1176/ajp.134.11.1239

Lewis, C. S. (2001). *Mere christianity*. Zondervan.

OpenLayers v5.3.0 API - Index. (2019). Retrieved from https://openlayers.org/en/latest/apidoc/

Overview - Apache CouchDB® 2.3 Documentation. (2019). Retrieved from https://docs.couchdb.org/en/stable/

Saint Thomas (Aquinas). (2013). *Summa theologica*. E-artnow Editions.

Twitter API Documentation. (2019). https://developer.twitter.com/en/docs

Appendix

Role of Team Member

We decided to separate roles into four categories: Deployment, Harvesting, Analysis, and UI. Each person will master one part of this roles. However, it is not necessary only him/her struggling with this role. If there was an issue arose, we try to help each other within the team.

The team works well in helping solving each other issue. We decided to create a group chats within a Messaging App to discuss everything related to this. When we created this system, we face lots of issues. We even had a very long discussion via a Messaging App. If there is a very difficult task, next day we will meet and discuss the solution together as a team. If the team can not solve it, we escalated this issue to the discussion board. For example, we escalated one issue related to ansible in discussion board.

Here is the specific roles for each Member in delivering the system:

Name	Roles
Dading Zainal Gusti	Deployment (Ansible, CouchDB)
David Setyanugraha	UI, Harvester (Tweepy and CouchDB)
Ghawady Ehmaid	Analysis, Harvester, Aurin
Indah Permatasari	Analysis, UI
Try Ajitiono	UI, Deployment (Ansible, CouchDB)

User Testing Guide

Before proceeding further, you will need to have these softwares in your system.

1. **Ansible v2.7.10** or any versions that are compatible with it to run the Ansible tasks
2. **Node.js® LTS v10.x** for the Javascript modules/runtimes
3. **Yarn v1.15.2** or any versions that are compatible with it for the UI package managements

The steps below are needed to get our application up and running.

1. Configuring Ansible variables. We will need to configure needed variables for the deployment because some variables aren't meant to be pushed to the Git repository for security reasons.
    a. Clone this repository with git clone https://github.com/davidsetyanugraha/c3-assignment2, then open the folder.
    b. Look for the **ansible** folder, then open it.

    c. Copy **nectar-credentials.sh.example** as a new file named **nectar-credentials.sh** inside the same folder. Put your Nectar credentials inside the newly copied file.
    d. In the project root, open **ansible/host_vars/nectar.yaml**. Change the value of **`instance_key_name`** to the name of your key-pairs in Nectar, e.g. **`nectar-key`**. This name is provided in the key-pairs menu inside the Melbourne Cloud Dashboard.
    e. In the project root, open **ansible/nectar.yaml**. Change the value of **`ansible_ssh_private_key_file`** to the absolute path of private key that you use in Nectar, e.g. **`~/Downloads/nectar-key.pem`**.
2. Before running the Ansible playbook, make sure you are using University of Melbourne's private network, either that be inside the campus or using VPN. If you are using the former, skip this step.
    a. Recommended VPN is Cisco AnyConnect.
    b. Fill the VPN host with: **`remote.unimelb.edu.au/student`**.
    c. Fill the credentials with your credentials at University of Melbourne.
3. Running Ansible Playbook. In this step, we will be running a script that will execute the whole playbook using our OpenRC credentials that we get from our Cloud Dashboard.
    a. In the project root, navigate into **ansible** folder.
    b. Run the file **run-nectar.sh**. This particular script will run **openrc.sh** (which uses our credentials from **nectar-credentials.sh**), then execute the Ansible playbook.
    c. Wait for the tasks to complete.
4. Checking the deployment.
    a. Check instances in Melbourne Cloud Dashboard.
        i. Log in to your Melbourne Cloud Dashboard.
        ii. In the **Compute** category of the sidebar, click on the **Instances** menu.
        iii. If there are 4 VM instances in the list, then your Ansible tasks have successfully deployed them in Nectar.
    b. Check CouchDB cluster setup
        i. Check using curl: **`curl -u admin:password`** **`http://{ip_address}:5984/_membership`**. Replace ip_address with one of your instance's actual IP address.
        ii. If it returns 4 array elements for **`all_nodes`** and **`cluster_nodes`**, with the 4 array elements are the IP addresses for your deployed instances, then CouchDB cluster setup is successful.
    c. Check the tweet harvesters
        i. Go to Fauxton's UI: **`http://{ip_address}:5984/_utils`**, replace ip_address with one of your instance's actual IP address.
        ii. If there is a database named **`victoria`** and its size/number of documents increases over time, then the tweet harvester is running.
    d. Check the web application
        i. Go to the IP address of the first instance: **`http://{ip_address_first_instance}`**. This will open port 80, which the Nginx web server is listening to based on our Nginx server block

configuration in **ansible/roles/setup-remote-nginx/templates/sites.conf.tpl**.

ii. If a dashboard page is loaded, then the Nginx web server deployment is successful as well.

Source code Link
https://github.com/davidsetyanugraha/c3-assignment2

Video Link
https://youtu.be/gOY1iIrPXxI