COMP900024 Cluster and Cloud Computing Assignment 2

Team 14
Kranthi Kumar Kommuri 1049507, James Ng 1316315, Emmanuel Pinca 1080088, Matteo Rossi 1222468, Thanadol Ultarnpatumros 1323866

Front-End: http://172.26.129.193:8080/
Github Repo: https://github.com/Jamescwng/COMP90024A2
Youtube Video: https://youtu.be/FFKjTw1GWvg

Abstract

The advent of social media as a channel of communication has enabled individuals to broadcast opinions in a way that is immediately accessible by other users of the platform at large. None more so that of twitter which enables mass soapboxing to the general public. As such we aim to explore twitter as a tool to analyse sentiment over the Melbourne Metropolitan area to infer any correlation between polluting entities; a measure of livability, and general sentiment based on tweet data. Various livability factors were considered however pollutant data was most greatly distributed across the region.

Data was primarily collected across 2014 to 2022, however was most represented over the 2014-2016 period. Sentiment was determined through a machine learning process by assigning weight to different words and phrases taken from the tweet text. The polluting livability factors were taken from the AURIN database for urban, regional and social science datasets.

Difficulties were found running comparisons between twitter data in AURIN primarily due to data quality mismatch however it was found that although weak, there exists an inverse correlation between sentiment based-on suburbs and number of polluting entities across the melbourne metropolitan area. For example more heavily industrialised suburbs had a stronger correlation with negative sentiment.

From these findings we can conclude that an increase in polluting entities leads to an increase in expressing negative messaging on social media and in the future it would be prudent to consider the placement of industrial zoning when considering policies related to improving livability in metropolitan areas.

Introduction

The liveability scenarios that we explored for the city of Melbourne are based on data gathered from Twitter and AURIN. Twitter is a social networking service where users post 280-character messages known as “tweets”. As of 2019, Twitter had more than 300M monthly active users [1], making it an ideal platform for performing sentiment analysis and extracting meaningful insights on various topics related to liveability. Our interests centred around liveability indicators such as environment and health; thus, we aimed to explore tweets related to these contexts. We were provided with a large corpus of historical Twitter data for the city of Melbourne, and we also relied on Twitter APIs to fetch more relevant data points. Overall, our analysis considered 300k+ tweets. AURIN is a platform funded by the Australian Government that gives social science researchers access to data and tools to interrogate the current state of Australia’s cities and use them for sustainable development [2]. Thus, AURIN is an ideal platform to gather intelligence into the liveability conditions of the city of Melbourne. Since we focused on the liveability metrics mentioned above, we collected datasets about food access, pollution, and green space availability.

Approach

To evaluate the liveability of the city of Melbourne, we first decided to compare how suburbs in the Greater Melbourne area [3] performed against each other according to some metrics. We performed sentiment analysis on all the tweets we collected and calculated each suburb's percentage of tweets classified with a positive sentiment. The tweets were classified either as "positive", "neutral", or "negative", and we also recorded a confidence score for each tweet. We then performed a topic analysis only on the positive tweets in which we ran a clustering algorithm to extract relevant topics. This last analysis aimed to guide our search for relevant datasets on the AURIN platform. The results from our topic analysis suggested that keywords related to food were strongly correlated with a positive sentiment. Unfortunately, we could not find a rich dataset on AURIN related to food production, food security, or food access. However, we found a dataset about restaurants, although it only included data for the suburbs of the City of Melbourne [4]. Furthering our analysis of our sentiment classifier results, we noticed that some industrial suburbs were associated with a lower positive sentiment. Thus, our focus shifted to finding a dataset on AURIN that could support our scenario related to environment and health liveability indicators. After careful research, we came across a dataset highlighting the number of pollution reports within each suburb that we could relate to our observed results. In the following sections, we will discuss our findings.

Sentiment Classifier

We ran a pre-trained sentiment classifier algorithm on all the collected tweets to perform sentiment analysis. The classifier is known as “cardiffnlp/twitter-roberta-base-sentiment-latest” [5]. It is trained on ~124M tweets from January 2018 to December 2021 [6] and fine tuned for sentiment analysis with the TweetEval benchmark [7]. Before feeding our tweet instances into the pre-trained classifier, we also pre-processed the tweets to reduce noisy instances and improve the quality of our dataset. We used the following methods: stop word removal, emoticon to text translation, username removal, hyperlink removal, non-Ascii character removal, email address removal, and special character removal. We ran the algorithm before feeding the data into CouchDB; thus, every document in CouchDB has a sentiment associated with it.

Topic Classifier

We used a BERT and pre-trained transformer-based model [8] to classify the tweets into topics as similar approaches have shown promising results over the last years [9]. As mentioned in the Approach section, we only considered positive sentiment tweets for this analysis. The inner workings of the algorithm are beyond the scope of this paper. The results of the classifier are presented below. Figure 1 shows the generated clusters, while Figure 2 shows the most prominent words belonging to the second largest cluster, the red one in our graph. The largest cluster, the blue one, contains positive sentiment tweets that are very general. For example, they are classified as positive because they contain "beautiful", "lovely", or other similar words; thus, we don't consider them relevant to our analysis. Figure 3 shows the sizes of the ten largest clusters, where "-1" is the blue cluster and "84" is the red cluster. Overall, we can conclude that the data gathered is very sparse and noisy and that it is difficult to find topics related to health and environment liveability in general. However, the "food" related cluster is an interesting finding.

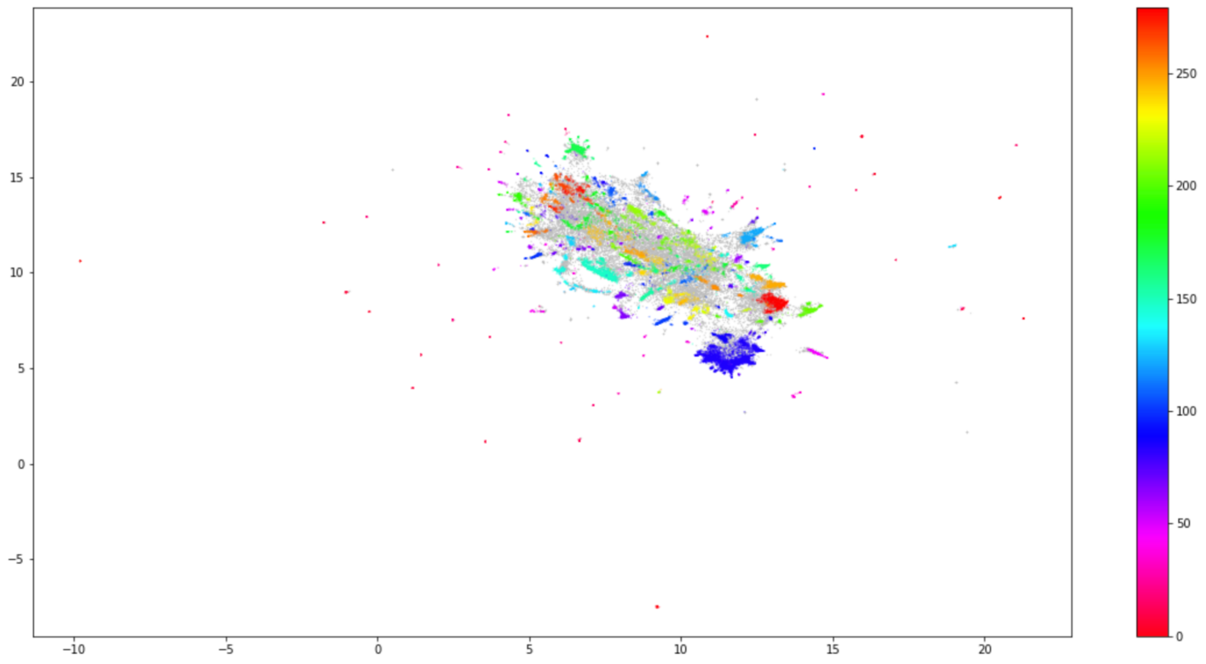
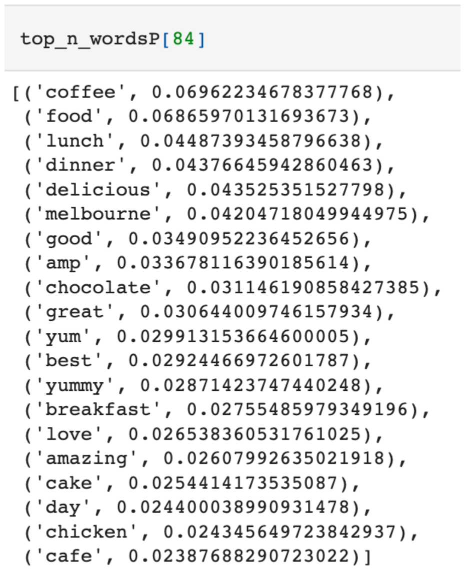


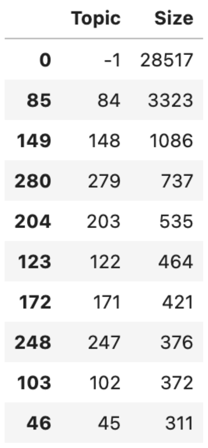
Figure 1. Topic Clusters for Positive Tweets



```
top_n_wordsP[84]
```

```
[('coffee', 0.06962234678377768),
 ('food', 0.06865970131693673),
 ('lunch', 0.04487393458796638),
 ('dinner', 0.04376645942860463),
 ('delicious', 0.043525351527798),
 ('melbourne', 0.04204718049944975),
 ('good', 0.03490952236452656),
 ('amp', 0.033678116390185614),
 ('chocolate', 0.031146190858427385),
 ('great', 0.030644009746157934),
 ('yum', 0.029913153664600005),
 ('best', 0.02924466972601787),
 ('yummy', 0.02871423747440248),
 ('breakfast', 0.02755485979349196),
 ('love', 0.026538360531761025),
 ('amazing', 0.02607992635021918),
 ('cake', 0.0254414173535087),
 ('day', 0.024400038990931478),
 ('chicken', 0.024345649723842937),
 ('cafe', 0.02387688290723022)]
```

Figure 2. Most prominent words for the second largest cluster

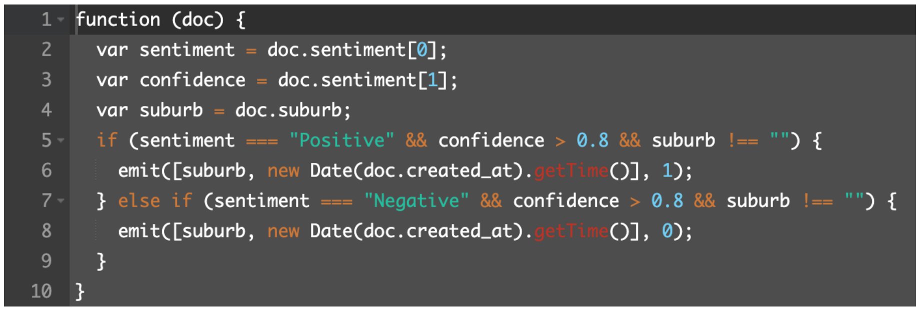


	Topic	Size
0	-1	28517
85	84	3323
149	148	1086
280	279	737
204	203	535
123	122	464
172	171	421
248	247	376
103	102	372
46	45	311

Figure 3. Sizes of the 10 largest clusters for positive tweets

Twitter Sentiment Analysis Findings

Using CouchDB's built-in MapReduce capabilities, we extracted only positive and negative sentiment tweets from our database with a confidence score greater than 80%. The purpose was to create datasets that we could use to generate high-quality maps for our front-end. The script that achieves this is shown in Figure 4 below. These tweets represent approximately 20% (or 70k+) of total tweets stored in the CouchDB database.



```
function (doc) {
  var sentiment = doc.sentiment[0];
  var confidence = doc.sentiment[1];
  var suburb = doc.suburb;
  if (sentiment === "Positive" && confidence > 0.8 && suburb !== "") {
    emit([suburb, new Date(doc.created_at).getTime()], 1);
  } else if (sentiment === "Negative" && confidence > 0.8 && suburb !== "") {
    emit([suburb, new Date(doc.created_at).getTime()], 0);
  }
}
```

Figure 4. MapReduce script

Furthermore, our analysis only considered suburbs with more than 50 tweets; otherwise, our results could become skewed. Figure 5 shows the suburbs with more than 50 tweets, while Figures 6 and 7 show all the available tweets and only the ones considered.

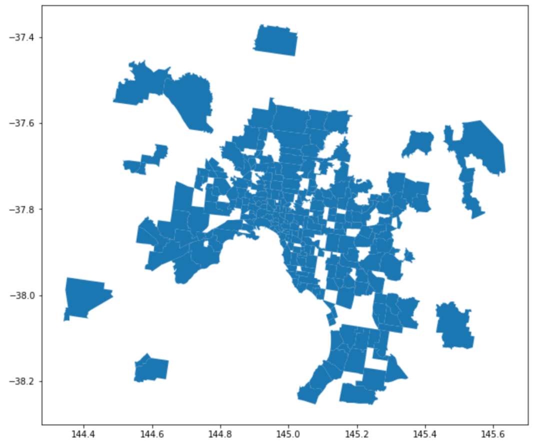


Figure 5. Suburbs with more than 50 tweets

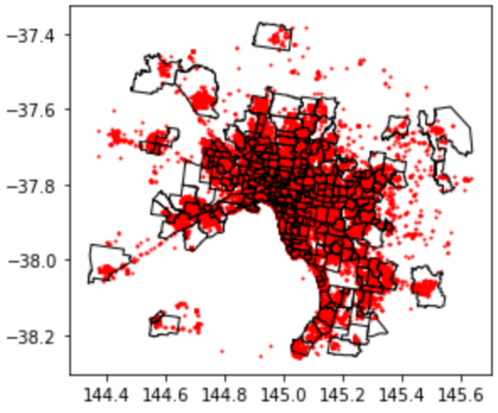


Figure 6. All positive and negative tweets with confidence greater than 80%

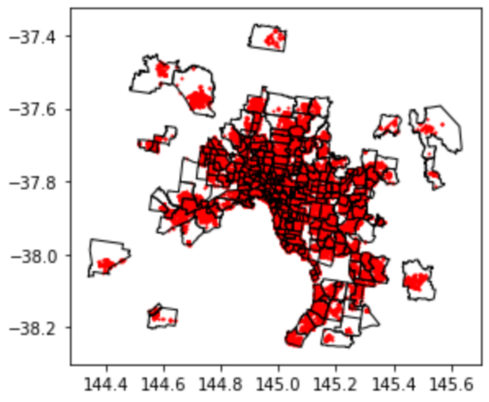
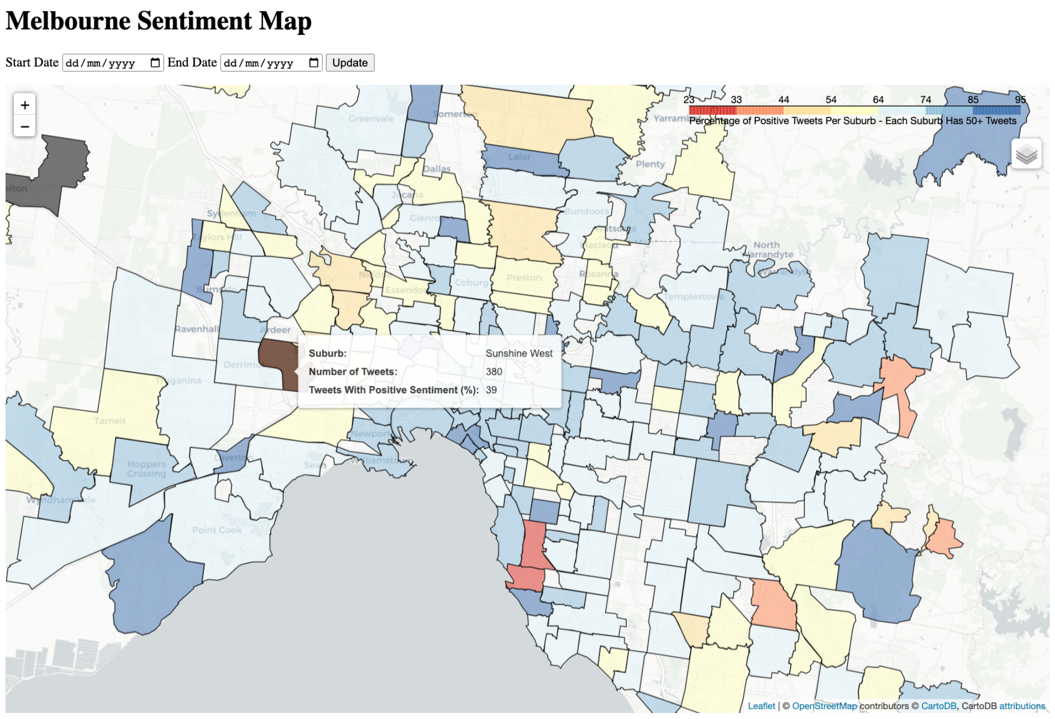


Figure 7. Tweets that fall within a suburb that has at least 50 tweets

Each tweet document in CouchDB has coordinates associated with it. To map each tweet data point to a suburb, we first extracted the locality boundary polygons from the state of Victoria [10]. We then ran a mapping algorithm to associate each coordinate point with a suburb polygon. Finally, each suburb's percentage of positive tweets is computed and displayed in an interactive map shown directly to the client when accessing the front-end. Figure 8 below shows the map displayed on the front end which is accessible at IP address 172.26.129.193:8080. As you may notice, we have also included a functionality to filter the sentiment results by time period.



Melbourne Sentiment Map

Start Date dd/mm/yyyy End Date dd/mm/yyyy Update

Suburb: Sunshine West
Number of Tweets: 380
Tweets With Positive Sentiment (%): 39

23
33
44
54
64
74
85
95
Percentage of Positive Tweets Per Suburb - Each Suburb Has 50+ Tweets
+
−
Leaflet | © OpenStreetMap contributors © CartoDB, CartoDB attributions

Figure 8. Sentiment results displayed when accessing the front end

To provide an example, in Figure 8, we can see that the suburb Sunshine West has a total of 380 tweets - with confidence greater than 80% - and that only 39% of these are associated with a positive sentiment.

Aurin Findings

As mentioned in the Approach section, we gathered a dataset from AURIN related to pollution, the National Pollutant Inventory dataset [11]. The dataset, which was loaded to the database via the CouchDB APIs, comprises 754 documents, where each document specifies the number of reports from a given pollutant source. This information is also enriched with geolocation data of the given source. As an example, one such source is a Bulk Petroleum Storage Facility.

Similarly to how we dealt with the Twitter data, we mapped the point coordinate of the pollutant source to the suburbs that we considered in our sentiment analysis. Thus, if a suburb is considered heavily polluted, but the number of tweets in that suburb is fewer than 50, we would disregard this data point. From Figure 9 below, one can see the pollutant documents that we omitted from our analysis.

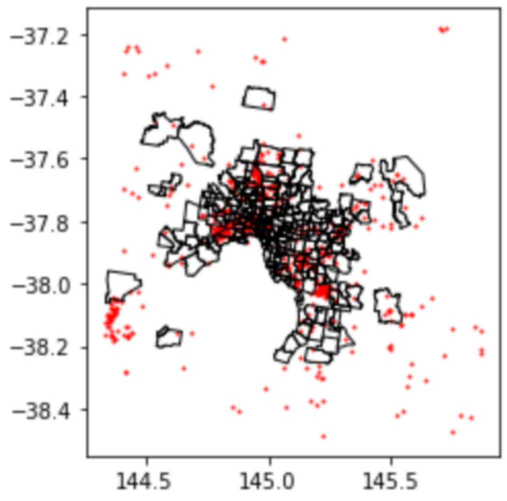


Figure 9. Pollutant documents in CouchDB relative to the suburbs considered

Finally, we summed the number of reports per suburb and created an interactive map that we displayed on our front end. The result offers interesting insights. For example, someone familiar with the city of Melbourne would expect the suburb of Port Melbourne to be a somewhat polluted area, which is indeed what we see in our results. Figure 10 below shows the map in question.

Pollution Map

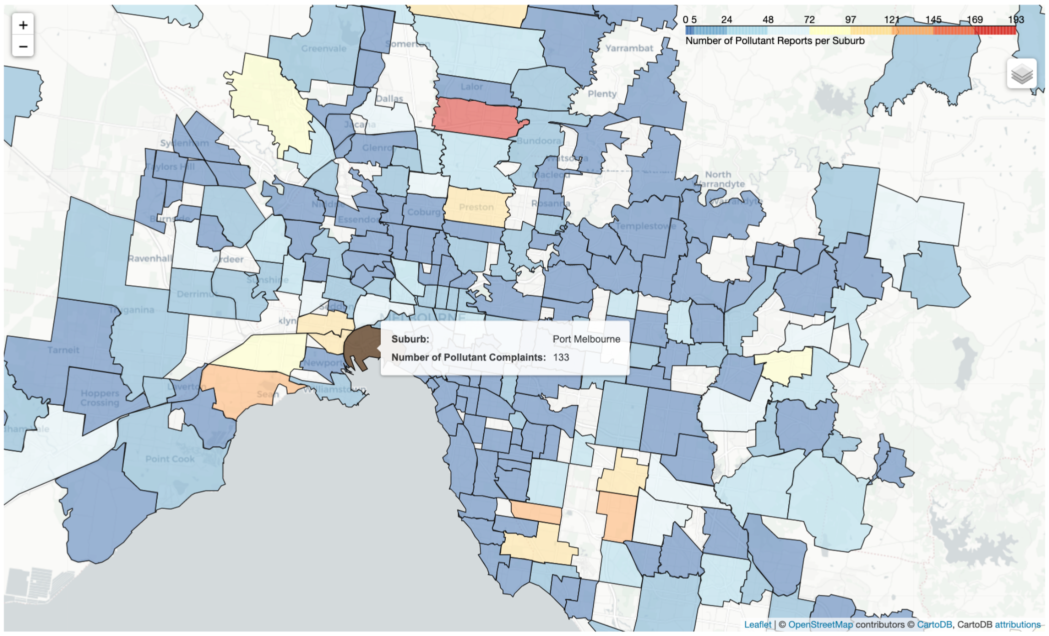

0 5 24 48 72 97 121 145 169 193
Number of Pollutant Reports per Suburb
Suburb: Port Melbourne
Number of Pollutant Complaints: 133
Leaflet | © OpenStreetMap contributors © CartoDB, CartoDB attributions

Figure 10. Pollution map for the suburbs considered

Correlation Findings

The final step in our analysis has been to compare the Twitter dataset with the AURIN dataset and derive meaningful liveability insights. Below we plot two graphs that show the correlation between the percentage of positive tweets and the number of pollutant reports per suburb.

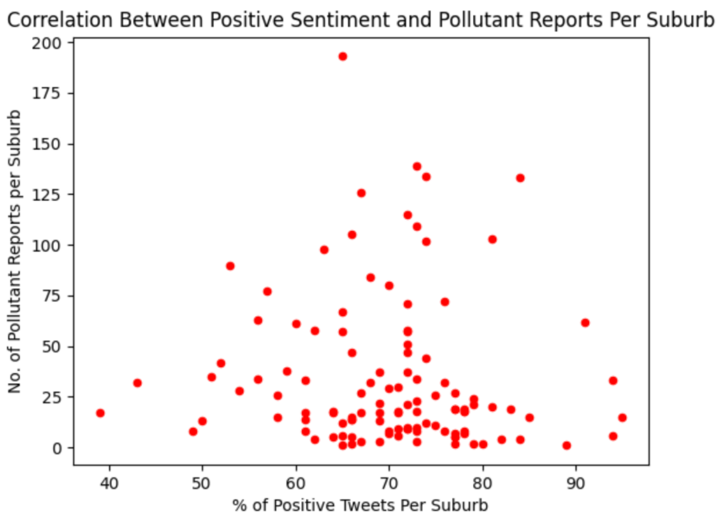


Figure 11. Correlation between positive tweets and pollutant reports per suburb - scatter plot

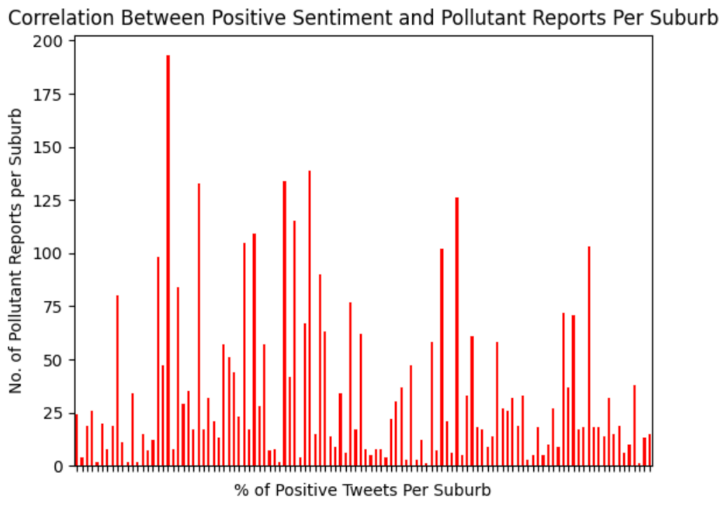


Figure 12. Correlation between positive tweets and pollutant reports per suburb - histogram

Although it is challenging to draw straightforward conclusions, it is evident that the suburbs with the highest percentage of positive tweets are associated with a lower level of pollutant reports. These conclusions are most apparent in the histogram chart. However, if someone compares the 60%-80% and 80%-100% intervals in the scatter plot, most of the “polluted” suburbs are within the former interval. This suggests that a positive correlation is present for certain suburbs. It is also worth mentioning that of the 70k tweets considered, only 20k were associated with a negative sentiment. Thus, the percentages for the suburbs are naturally skewed in favour of positive sentiment.

Challenges

Overall we found it quite challenging to find a rich corpus of data on AURIN that related to our research interest. The high-quality datasets we found were small and reflected a very niche topic. Besides the pollution reports datasets, we also analysed sports facilities and green space availability datasets. However, these datasets correlated poorly with our Twitter data. We believe this emerged because most of the suburbs with a high positive sentiment are clustered within the inner suburbs of the City of Melbourne, often where sports facilities and green spaces are fewest. Furthermore, the AURIN data was difficult to merge with our Twitter data in some cases as it lacked geo coordinates or suburb identification fields. For these reasons, we found it ideal to use pollution as the ideal metric to analyse the health and environment liveability indicators of the Greater Melbourne area.

Limitations of Mining Twitter Content

Geo-enriched Twitter Data

Geocoded tweets were the primary limiting factor in regards to collecting and collating data via the python endpoint. Access to the tweets with attached location data was limited to Academic

Research and elevated developer accounts, of which elevated accounts were subject to aggressive throttling for premium queries.

Effectively for this project and due to only possessing elevated privileges, geocoded tweets were limited to the past 30 days. This dataset was enriched with an existing twitter dataset that covered the 2014-2017 period. Due to the fact this pre collated data was not subject to rate limiting, unlike the twitter API, this made up the bulk of the data set.

Furthermore geolocated data is turned-off by default and thus would introduce a level of bias into the dataset as it would not represent a true representation of the general twitter population. Effectively this would mean only users that were open to geo-enriching data would be shown. Contentious topics would generally be underrepresented in the set of twitter users willing to publish tweets that could be used to identify themselves.

Skewness of Twitter Demographics

Twitter users typically are a small subset of the Australian population which would mean that data collected from twitter could not be used to generalise across the Melbourne population, and to further compound this issue, tweets are not distributed evenly across the twitter user base. A smaller subset of active twitter users typically make the bulk of tweets hence the opinions of more active users are represented more strongly in this dataset.

Location data not representative of community

Twitter location data is set to the location at the point of publishing. Thus issues will arise when trying to correlate sentiment by publishing location to the community from which the user resides. Tweets originating from the Melbourne CBD would be a greater indicator of commuter behaviour rather than evidence of residence; from the data collected, more than half of the total tweets were published in an area that was within three kilometres of the Melbourne Central Business District. This is unlikely to be due to population distribution and more likely due to location of occupational posting.

Determining Intent from tweet text

Due to the non-standard nature of twitter usage as compared to more formalised forms of data collection, i.e. surveys and interviews, twitter text can be fraught with linguistic features that may muddle intent, for example sarcasm. Another example is that an initial topic of tweet analysis was crime however this was muddled as it was difficult to determine context of the tweets that related to crime and/or criminal activities, i.e. James Hird killed it on the weekend, would register as crime related despite the original intent not being the case. As such, we were forced to pivot to a topic that considered tweets more generally.

Managing tweets processing

Due to retweets effectively increasing the relative representation of specific tweets in the data analysis, these retweets were filtered and not included in the database. This was done through tags within the tweet metadata.

Tweets were inserted into the database with the unique identifier of the tweet (the tweet id) used to mitigate any dedup issues. This was done by checking the existence of the key in the database.

Various server sided limitations were enforced on the twitter api, as mentioned above. Other than throttling done specifically on premium twitter API calls, another limiting factor was having tweets shaped after returning 450 tweets per 15 minute window [15].

Melbourne Research Cloud

The Melbourne Research Cloud (MRC) exists as a platform for University of Melbourne researchers, staff and students to timeshare computing infrastructure. The platform functions similarly to other Infrastructure as a service (Iaas) platforms, e.g. Microsoft Azure, Amazon AWS and Google CP.

The research cloud provides on-demand resources to handle a variety of use cases such as data analysis, data storage, and data processing. Boasting approximately 20,000 virtual cores on a range of VM instance types (flavours) [11]. In addition the MRC is able to provision specialised resources such as GPU compute cores; for parallel processing, networking, load balancing and DNS name hosting in order to add security and fault tolerance to a project [11].

There are no direct costs associated with using the MRC with membership given on the provision of users being a member of a pre-approved project [14]. Funding of the platform is borne by the university which primarily derives its income from fees collected from students, government sources and grants.

Contemporary scientific research is increasingly becoming reliant on machine computation, likewise the need for performant hardware has risen proportionally. Given that such costly hardware could typically be purchased for a single research function before becoming redundant means that a centralised research platform such as the MRC is especially beneficial in a research setting. Thus a federated solution like the MRC allows users to timeshare compute resources, i.e. cores having a greater utilisation over a given time period.

System Design and Architecture

The system is hosted on the Melbourne Research Cloud across multiple instances. Each of these instances contain separate system components and require a platform for communication, ensuring each component contributes to a singular cohesive network. This system includes: a front-end web application, a twitter miner, database nodes and volumes.

System Components

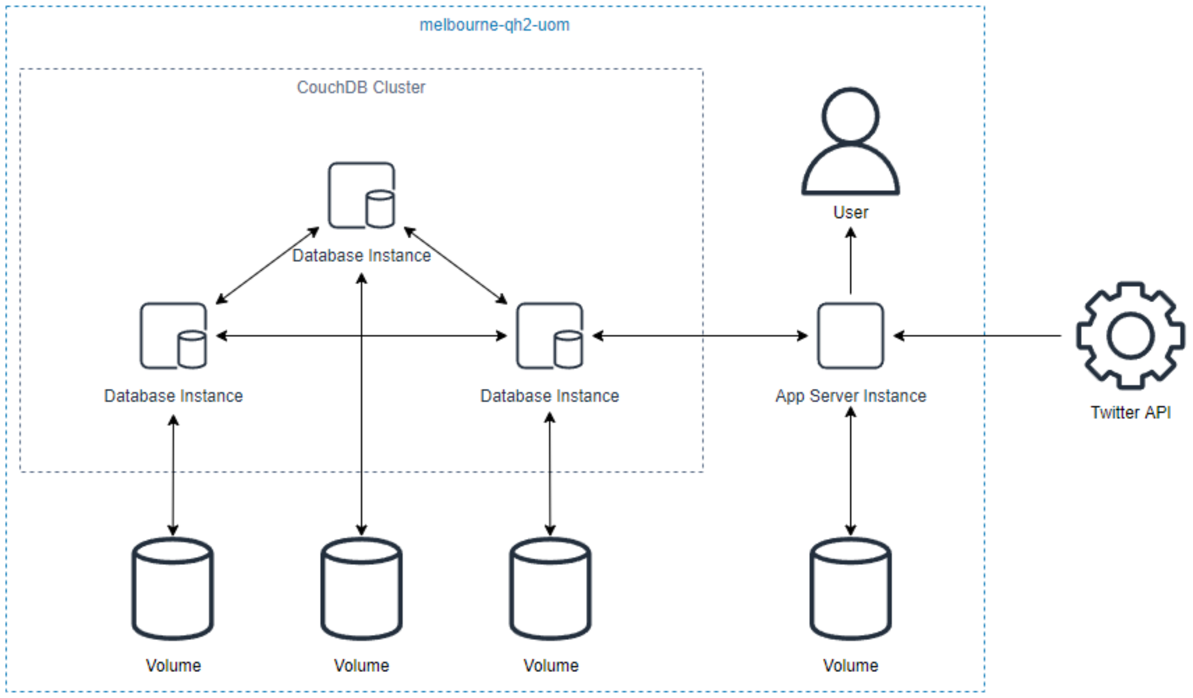


Figure 13. Architecture Diagram

Front-end Web Application

This is the primary endpoint for users of the system. It displays the collected information and findings inferred from the twitter and AURIN data. The application was built using the Flask framework through Python and the displayed data is sourced from the CouchDB database. This web application should be the only component exposed to the public (within the university's network) and runs from a single instance.

Twitter Miner

The twitter miner collects and deposits tweets from Twitter through the Twitter API to the system's database. Each collected tweet has their location in Melbourne and is classified by a sentiment analyser. The miner runs on a single instance and feeds data into a single database node.

As it is set up, the miner shares an instance with the front-end application to separate the database nodes from the other components. Ideally, each node should contain a single component to ensure each is assigned a singular task; however with a maximum of four nodes, this design decision was not implemented. Having each node complete a single function would have minimised coupling, increasing the system's maintainability.

Database Nodes

Multiple instances collaborate and create a CouchDB cluster to store the collected data. Initially, three instances are assigned to host cluster nodes. The cluster has two shards with three replicas, creating a total of six shard replicas distributed across the cluster. Each node can operate as an entry point to the database, however the data scraped from twitter is gathered via the master node.

Utilising multiple replicas allows the system to become fault resistant when a node fails. This is due to the ability for the database to still be available while the nodes are down via access to other nodes. Three database nodes were implemented to minimise downtime and complexity [12].

Volumes

Volumes add additional storage to their mounted instances. Each can persist after dismounting and contain files that are required by the other components to execute.

Benefits of MRC/IaaS Solutions

Scalability

Opposed to typical physical hardware solutions, MRC and other IaaS solutions are provisioned from a larger pool of resources, which means it is easy to supply additional computing cores/ volumes on request if free resources exist in the pool of available resources which in most cases is effectively immediate [11].

Multiple instances can exist on a single rack which leads to improvement in latency over an equivalent system architected over multiple physical servers.

Auto scaling also presents a way to dynamically adjust the amount of active resources during runtime based on the total throughput capacity of a system. Additional active resources are provisioned after a set of criteria is met and reduced when not required. This happens seamlessly without the interruption to end users. Benefits of auto scaling include reducing costs when not needed, reduction on electricity use, etc..

Fault tolerance

Load balancing is the act of distributing client requests over a pool of resources. Fault tolerance of the system is improved by ensuring that individual components of the system are not over utilised which can cause performance issues and thread lock-up which can negatively affect the user experience.

Similar to autoscaling, it acts to improve system performance, however it does it by distributing more evenly over multiple servers as opposed to creating more resources. Auto scaling and load balancing must be performed by a dedicated resource (either manually by the system architect, or managed by the platform) which means increased overheads and thus may not be suitable for smaller projects.

Backups

MRC, other IaaS platforms, provide multiple ways to create cold-storage data redundancy. Solutions such as creating images from virtual machines or database backups are trivial to accomplish.

Images are typically bitwise copies of already created environments and are a one-to-one (including operating system) of an existing or previously created instance and can be very useful when the user wants a baseline virtual machine i.e Arch Linux x64 with node installed.

Privately created images can then be distributed to the public via image repository which are inbuilt into a platform's VM deployment options. IaaS providers will also characteristically create, manage and maintain in-house images with updated packages/modules for increased security [11]. One limitation of publicly available images is that they tend to be fairly generic to suit more use cases however the image repository consists of many different images and hence servers types that are heavily deployed usually have an associated image, e.g. A server that is used to manipulate large data sets using python and pandas.

Database backups function similarly to images however can have two forms, a full backup of data within an existing database and incremental backups which function as a delta off an existing full snapshot. Incremental backups are smaller thus fewer resources are required to create multiple incremental backups [13]. Full backups are larger but can be used to completely restore the state of a database. Good practice is to create a full backup prior to any major release with deltas being produced on regular intervals.

Flexible Allocations

As stated above MRC supports a range of computing use cases, it does this by allowing the user to select VM sizes and types. For example users with a more computationally heavy workflow may opt in for flavour (MRC’s designation for VM sizes) with more cores, likewise users who aim to stitch larger data sets together may opt for larger attached volumes or greater ram allocations. Contrast this with physical hardware where the decisions made on acquisition are effectively locked-in for the life of the hardware and hence more consideration and planning must take place before selection. Furthermore researchers may opt to have ‘a little bit more’ when trying to avoid underprovisioning a system which can lead to buffer allocation which can be wasteful or alternatively restrict the ability for a project to pivot.

Security Configurations

IaaS platforms manage security via an active directory style interface. In MRC this occurs via configurable security groups. Security groups act to allow users to atomically set access ports and protocols on a group by group basis or instance by instance basis. Furthermore MRC provides users with a choice to deploy across one or many availability zones increasing fault tolerance should a network outage occur.

One issue with MRC and other IaaS providers is that by design the instances are always attached to the network, thus the attack surface cannot be fully mitigated. We can compare this to a fully offline server which presents no method of intrusion by malicious actors.

Disk storage

MRC can assign a variable-size disk based on user preferences either during instance creation, which is affixed to the instance and cannot be removed; this will typically contain the operating system and other essential software, and block storage which is attached to the disk, is resizable and can be transferred across instances. Furthermore a single block store can be accessed simultaneously across multiple instances through a server running an NFS or CIFS protocol.

Compared to a physical hardware system, the MRS volume sizes can be dynamically resized. In addition, creating a backup of a physical volume can be especially time consuming and costly as an additional disk is required for each subsequent backup. Secondly versioning across each backup can prove challenging, especially with physical hardware as disks typically cannot be hot swapped.

Image Creation

Image management on the Melbourne research cloud is divided into two major categories, public images that are supplied by the Research Computing Services. These images are designed to be general purpose to suit a range of potential use cases. The images provide the service layers that sit on top of the infrastructure provided by the MRC. In these images, an operating system is packaged with related software to perform and run an application. Alternatively users may generate an instance snapshot based on an existing running image which can then be deployed onto a new instance. This enables users to duplicate environments and also allows the easy parallelisation of servers.

Image Deployment

For the purposes of this assignment ansible was used as an automation tool to deploy server objects on the network. The ansible runbook was configured to deploy the frontend, backend and NoSQL database from a bare state instance. Anisble was configured to select and run code pulled from a github repository. Existing data state was stored on a volume block that was attached at creation.

Automation with Ansible

S. No.	File Name	Description
1	Common Level	Code in this level is used to install the required packages on the host.
2	Volume Level	Code in this level is used to create storage volumes on the cloud infrastructure. Code uses the variables defined under “Volumes” to size and name the created storage volumes.
3	Security Group Level	Code at this level is used to create the security groups on the cloud infrastructure. Security groups are created using the variables defined under “Security_groups” to determine the ports that need to be opened, protocol, ip addresses to which access needs to be allowed etc.
4	Instance Level	Code at this level creates the instances on the cloud infrastructure. Information in the host_vars files is used to assign security groups, attach volumes etc. that are already created. Apart from that, it creates in-memory inventory of the launched instances. This inventory has three groups: 1. ALL_SERVER_NODES : Contains ip addresses of all the nodes that are launched. Useful to execute code that needs to perform action on all the nodes.

		2. APP_SERVER_NODES : Has the ip addresses of the "Application Servers". Useful to execute code on Application servers only. 3. DB_SERVER_NODES: Has the ip addresses of all the nodes where database (replicas) are deployed. 4. DB_MASTER_NODE: Contains the ipaddress of the node that is the maser node for couchdb deployment.
5	Application Level	Code at this level is used to perform the tasks which are specific to the application being deployed. This includes installing/updating the required packages, installing docker etc. Applications being developed can be deployed in several ways hence code is separated so that we can easily change the application architecture without impacting the infrastructure.
6	Database Level	This level is used to deploy the database required for the application. Selected approach deploys docker container and configures the couchdb cluster, adds the databases as required, mounts storage volume which can be used to transfer any files from local host to the remote database host etc.
7	App Server Level	This level is used to configure the application server and copy the code and files from the local host to the application server. This includes the application to scrape tweets and the front end code.

Table 1. Ansible Deployment Sequence

User Instructions

Command to execute the playbook:

```
. ./Team14_openrc.sh ; ansible-playbook DeployInfra_playbook.yaml
```

Please note that the code needs to be executed from under the *MRCInfraPlayBook* directory.

To copy files from localhost to remote host DB Servers:

1. Ensure the files that need to be copied are under **MRCInfraPlayBook/roles/Database_Level/files/**
2. Update the task file **MRCInfraPlayBook/roles/Database_Level/tasks/load_data.yaml** by including a command to copy using existing sample as reference

To copy files from localhost to remote host App Servers:

1. Ensure the files that need to be copied are under **MRCInfraPlayBook/roles/AppServer_Level/files/**
2. Update the task file **MRCInfraPlayBook/roles/AppServer_Level/tasks/main.yaml** by including a command to copy using existing sample as reference

To create a new database on DB Servers at the time of node creation:

1. Modify the **add_databases.yaml** file under **MRCInfraPlayBook/roles/Database_Level/tasks/** and add another block similar to the one for *twitter* database

Database will be created when the instances are created.

Scalability

The solution built is scalable and can be deployed on any number of Database server nodes and Application servers by simply modifying the entries in host_vars.yaml. This allows us to have multiple application servers which can be used to serve different groups (based on geographical location etc.) using a load balancer to provide better user experience.

Given the restriction on the number of nodes we can spin-up we couldn't test the scalability beyond a point. However, we used ansible script to launch a varying number of nodes which worked as expected.

Fault Tolerance

Application developed is fault tolerant on the database server side and can continue serving users even if we lose up to 2 database nodes. Additional nodes can be created by increasing the number of nodes in host_vars.yaml and executing the ansible script.

Given the restriction on the number of nodes we can launch, we couldn’t really deploy a fault tolerant application as we have a single application server. If we have one more node, we can launch another application server providing tolerance against a faulty app server.

Team Member Contributions

Student	Contributions
Matteo	Performed sentiment, topic, and scenario analysis. Created charts, interactive maps, and front end. Written documentation relevant to these topics.
James	Python scraper and twitter api scripts. Data collection and filtering. Documentation relevant to MRC, Twitter API, General issues related to data collection.
Emmanuel	CouchDB cluster, couchdb ansible scripting and front-end. Documentation relevant to architecture.
Thanadol	Topic and scenario research, ETL from Aurin to CouchDB, Some MapReduce, Minor issue related to loading data to CouchDB, and Documentation
Kranthi	Ansible automation. Documentation relevant to benefits of MRC/IaaS and ansible automation.

References

[1] Number of monthly active Twitter users worldwide from 1st quarter 2010 to 1st quarter 2019, Accessed May 2022, (https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/)

[2] Aurin Website, Accessed May 2022, (https://aurin.org.au/)

[3] Melbourne Spatial Data, Accessed May 2022, (https://www.planmelbourne.vic.gov.au/maps/spatial-data)

[4] City of Melbourne Suburbs, Accessed May 2022, (https://www.melbourne.vic.gov.au/about-melbourne/melbourne-profile/suburbs/Pages/suburbs.aspx)

[5] Twitter-roBERTa-base for Sentiment Analysis - UPDATED (2021), Accessed May 2022, (https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest)

[6] Twitter 2021 124M (RoBERTa-base), Accessed May 2022, (https://huggingface.co/cardiffnlp/twitter-roberta-base-2021-124m)

[7] F. Barbieri, et al., TWEETEVAL: Unified Benchmark and Comparative Evaluation for Tweet Classification, 2020, Accessed May 2022, (https://arxiv.org/pdf/2010.12421.pdf)

[8] Topic Modeling with BERT, Accessed May 2022, (https://towardsdatascience.com/topic-modeling-with-bert-779f7db187e6)

[9] D. Angelov, Top2Vec: Distributed Representations of Topics, 2020, Accessed May 2022, (https://arxiv.org/abs/2008.09470)

[10] Locality Boundaries (Property) (polygon) - Vicmap Admin, Accessed May 2022, (https://discover.data.vic.gov.au/dataset/locality-boundaries-property-polygon-vicmap-admin)

[11] National Pollutant Inventory, Accessed May 2022, (https://data.aurin.org.au/dataset/au-govt-dee-national-pollutant-inventory-facilities-2018-na)

[12] Cluster Theory, Accessed May 2022, (https://docs.couchdb.org/en/3.2.0/cluster/theory.html#cluster-theory)

[13] MRC User Guide, Accessed May 2022, (https://docs.cloud.unimelb.edu.au/faq/)

[14] MRC Dashboard, Accessed May 2022, (https://dashboard.cloud.unimelb.edu.au/)

[15] NECTAR FAQ, Accessed May 2022, (https://ardc.edu.au/services/nectar-research-cloud/)

[16] Twitter API, Accessed May 2022, (https://developer.twitter.com/en/docs/twitter-api/)