THE UNIVERSITY OF MELBOURNE

Assignment 2

The University of Melbourne
COMP90024: Cluster and Cloud Computing
Semester 1, 2020

Eric Sciberras 761250
Jake Musiker 694178
Akhmetzhan Kussainov 10263001
Dhairya Kamal Shah 1105509
Mei-Chen Lin 1018399

Assignment 2 **1**
Introduction 3
Our Scenario 3
System Architecture 3
Data Flows 4
Deployment View 5
I. Twitter Harvester and Couchdb Instance 5
II. Dashboard instance 6
Strengths and weaknesses of the system 6
An idealised system 6
Our findings 7

1. Introduction

The purpose of this assignment is for students to build a non-trivial software system using cluster and cloud computing for the implementation. Our team has developed the system for the specific scenario using CouchDB, Ansible, and Express frameworks for development. The team used Unimelb Research Cloud as a software environment. Moreover, the team used the Aurin network to gather the information about population for the project.

2. Our Scenario

Recently one of the biggest and most significant events to occur has been the coronavirus pandemic, within a few months period it has quickly altered what those around the globe consider to be everyday life. With Australian residents ordered into lockdown, a large proportion of the public concerns have been focused on the economic impact that this pandemic will have. To minimize the impact that this pandemic will have on the country, the government has introduced some economic response initiatives such as Jobseeker and Jobkeeper payments. However, although these initiatives were intended to help Australians, support for these relief packages have been mixed. The responses have ranged from some feeling as though the wrong members of the public are benefitting through loopholes to others feeling as though the payments are incredibly helpful, and with plenty of differing viewpoints across the board.

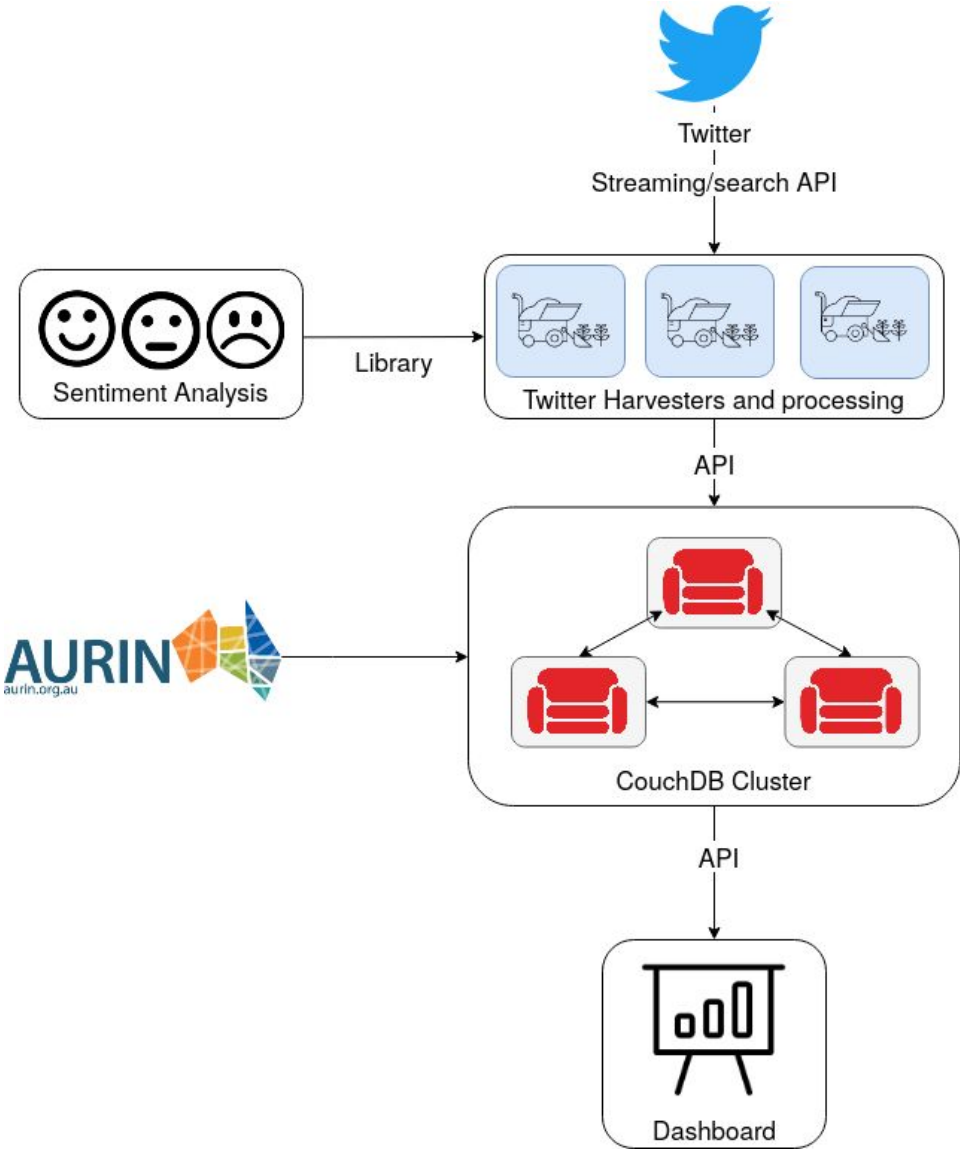
For this reason, we decided to investigate whether tweets mentioning #jobseeker or #jobkeeper had an overall positive or negative inclination using a sentiment analysis tool that was created for the analysis of social media. The sentiment analysis tool utilizes a range of rules to determine the overall sentiment of a piece of text, associated with a normalized score between -1 and 1. It is possible to adjust the sentiment of the given term or sentence relative to its environment (usually on the type of tweets harvested). With this we were able to analyze the differing attitudes of Australians towards the recent economic initiatives announced and implemented by the Australian government. By aggregating the tweet sentiment for each major city and then comparing these results to a data set from AURIN containing the labor market breakdown by the industry for the statistical area 4 levels we hope to be able to infer which industries are most benefited by these economic schemes. Furthermore, by analyzing the change in tweet sentiment over time we can view the change in the public's attitudes towards the policies and hopefully get an indication as to the effectiveness of these newly implemented economic response initiatives.

3. System Architecture

This section will talk about the design of the system architecture. Namely, we will describe the system by how data flows within the system and how the infrastructure is laid out. We will also address strengths and weaknesses in the architecture (redundancy, points-of-failure) and possible fixes are given an ideal situation (unlimited resources)

I. Data Flows

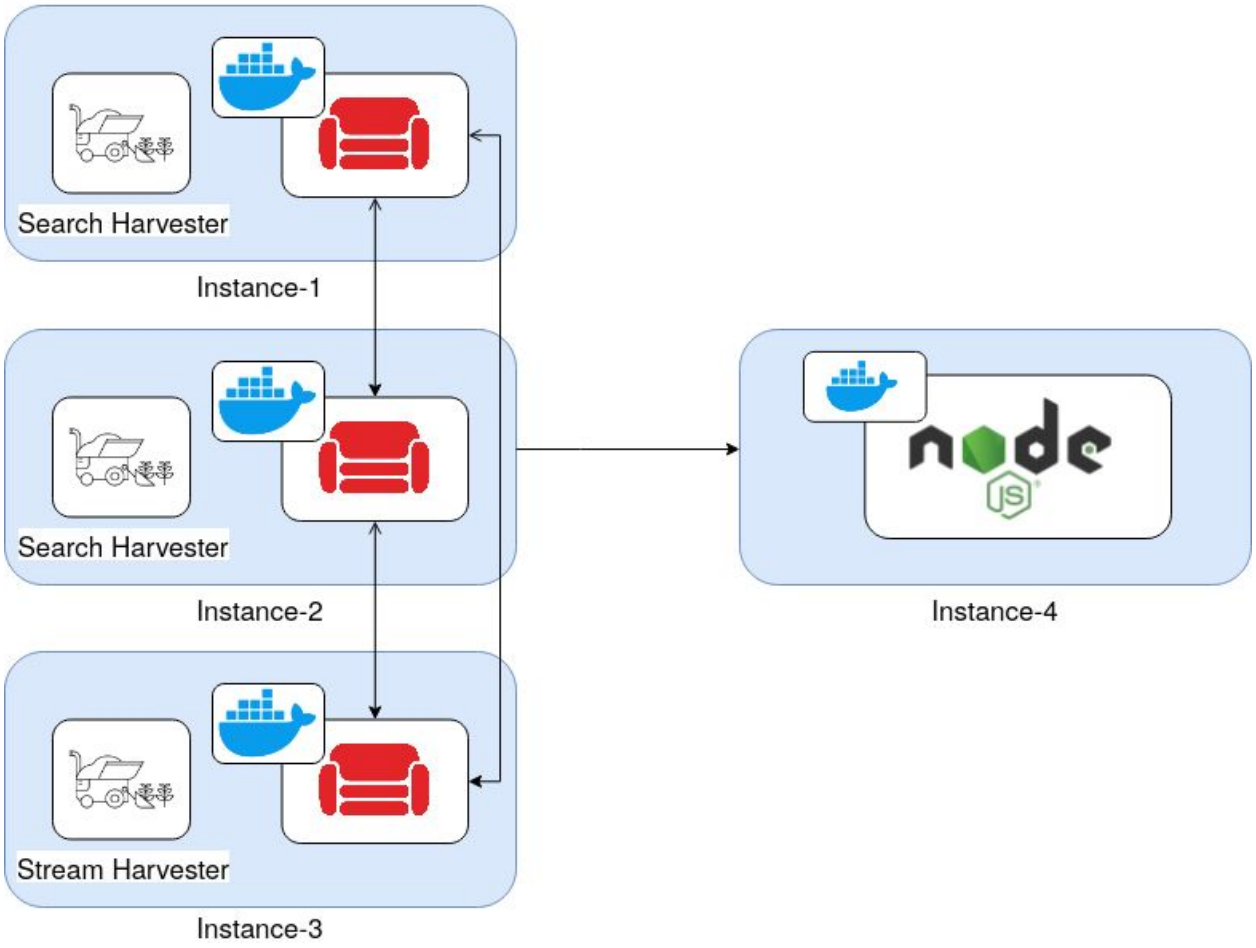
The diagram below represents how our data flows within our system.



Firstly, data is collected from Twitter using the search and stream endpoints. This allows us to collect both live and recent tweets (up to 7 days old). These tweets are enriched by adding sentiment information by the use of a sentiment analysis library VADER-Sentiment. The tweets are processed based on the sentiment analysis segregation of positive and negative tweets. Once the tweets are processed they are then sent to CouchDB where they are distributed amongst the cluster. Parallelly, we also incorporate the datasets from AURIN which will be used to further enrich the twitter data. Finally, when the data is enriched and processed within CouchDB it is then accessed by the dashboard by utilizing CouchDB views and the API.

II. Deployment View

The diagram below represents our system in terms of the infrastructure (instances)

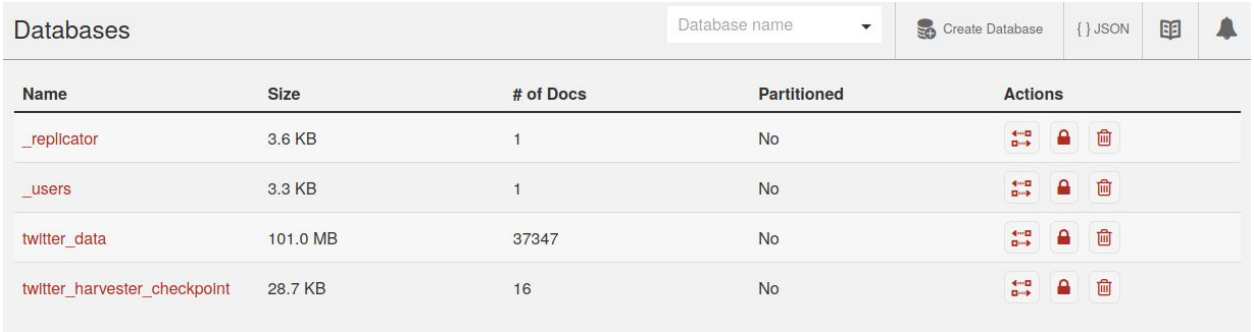


In our architecture, our instances are split into 2 types

- Twitter Harvester and Couchdb Instance

    - 2 search harvester
    - 1 stream harvesters
- Dashboard instance

I. Twitter Harvester and Couchdb Instance



Name	Size	# of Docs	Partitioned	Actions
_replicator	3.6 KB	1	No	
_users	3.3 KB	1	No	
twitter_data	101.0 MB	37347	No	
twitter_harvester_checkpoint	28.7 KB	16	No	

We have a couple of instances that contain a CouchDB and Twitter harvester service running. There were a number of factors that went into combining these services into single instances, Namely

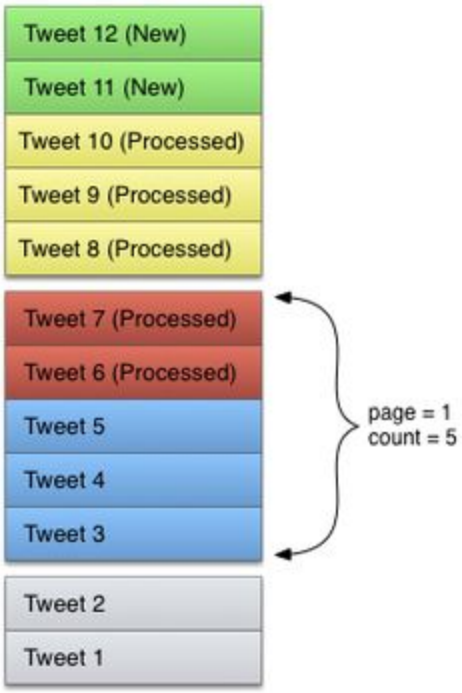
1. **The limited amount of instances:** For the architecture, we wanted to have a 3 node CouchDB cluster to effectively leverage the benefits associated with clustering such as availability. This left us with one remaining instance, therefore putting the Twitter harvester with CouchDB made sense.
2. **Being unable to use a load balancer:** Due to not being able to implement a load balancer; each CouchDB instance had a different IP address. This creates an issue when maintaining a functioning system in the case that a CouchDB instance fails. If a CouchDB instance fails that a harvester is exporting to, then the Twitter Harvester would need to have a mechanism to switch to another instance. Adding this logic adds complexity to the harvesters which we want to avoid.

Due to combining the services, in the event that an instance goes down we lose both a CouchDB node and Twitter harvester. While this isn't ideal it means that we don't have the situation where a Twitter harvester is orphaned, cannot call its CouchDB instance (assuming that we didn't implement a mechanism to switch to another CouchDB node).

Dealing with duplicate Tweets

As per our configuration 1 stream harvester and 2 search harvesters, there are some possibilities for duplicate tweets getting scraped. In the case of a search harvester grabbing a tweet that the stream harvester already has, this isn't an issue since that tweet is now further enriched (the tweet may have new information such as an updated retweet count).

As for the search harvesters clashing with each other or within their own runtime, we have set checkpoints that prevent us from scraping tweets.



As this image demonstrates (Image source Twitter API) The search harvesters will start processing tweets (starting from tweet 10) in this instance. Once they reach the oldest tweet (tweet 1), they then go back to the newest tweets but this time stop once they reach the newest tweet they processed from the last round (in this case tweet 10).

Furthermore, the search harvesters randomly pick a city to grab tweets from after each round to reduce the chances of them both scraping the same city at the same time.

II. Dashboard instance

In our system we have one Dashboard instance. This instance contains a node.js web application that is running inside a docker container. Due to time constraints the node.js application only talks to one CouchDB instance. This is certainly a flaw in our system.
The dashboard instance is for revealing the data patterns we have observed from harvested tweets. By making API calls to CouchDB instances, the CouchDB views generated by MapReduce functions will be collected and displayed on the frontend web server for data visualization purposes. We have provided different aspects of observation including the number of tweets related to #jobseeker and #jobkeeper in each state, how the average sentiment of tweets differs between states, how the overall sentiment changes over time, and finally we made comparisons with Aurin data for further correlated findings. The selection of Aurin data is the labor market statistics of the Statistical Areas Level 4 (SA4) regions, which shows reliable employment data among various kinds of industries. The detailed analysis regarding the above will be elaborated later in Section 4 Our findings.

III. Strengths and weaknesses of the system

Strengths of the system:

- It can develop more insightful data-based positive negative tweet harvesting/filtering strategy.
- At any point of failure to the dashboard the system can create a duplicate dashboard with a different IP.
- CouchDb cluster: better scalability due to sharding and better availability in the case of a node going down

Weaknesses of the system:

- Lack of labeled data can be a barrier in the area of advancement.
- If any instances go down there will be hampering on CouchDB and twitter harvester.
- Dashboard only connects to one CouchDB node (there is no client side load balancer)

IV. An idealised system

Due to limitations with time, expertise in cloud technologies, and the Melbourne research cloud resources our system does have some weaknesses. In this section, we will try to address this weakness by talking about what an ideal architecture would look like if constraints didn't exist.

Running Services on single instances: In our system, we have coupled a twitter harvester with a couchdb instance. This means if an instance goes down we lose both these services. To address this we could place these services in separate containers.

Adding redundancy for the dashboard: Our system contains one dashboard instance which creates a single point of failure. To address this we can create duplicate dashboard instances. However, this means that all our dashboard instances will have a different IP address, therefore, users would need to go to different URLs if one goes down. Of course, large companies cannot run their website in one instance so there is a solution to address this which leads to our next point.

Using Load balancers for the dashboard and CouchDB instances: A common pattern in production systems is to place duplicate services behind a load balancer that adds redundancy and scalability. It also reduces complexity for other services since they don't need to implement client-side load balancing.

V. Pros and Cons of Melbourne Research Cloud

While the application is being implemented and run on Melbourne Research Cloud (MRC) virtual machines, there are some particular behaviours in terms of its usage we would like to address. We will talk about what the application could benefit from MRC and in which ways MRC may bring down the performance of the application.

Pros:

1. Provides free cloud computing resources for researchers to implement data analysis applications and is competitive for having similar capabilities as other commercial cloud vendors such as Amazon Web Services.
2. The availability of allowing multiple users accessing the same virtual machine at the same time.
3. The capability of enabling single or multiple cores for building customized infrastructures depending on different volumes required.
4. The capability of building instances with internal network providers that assist in achieving better data security by limiting access from external networks.

Cons:

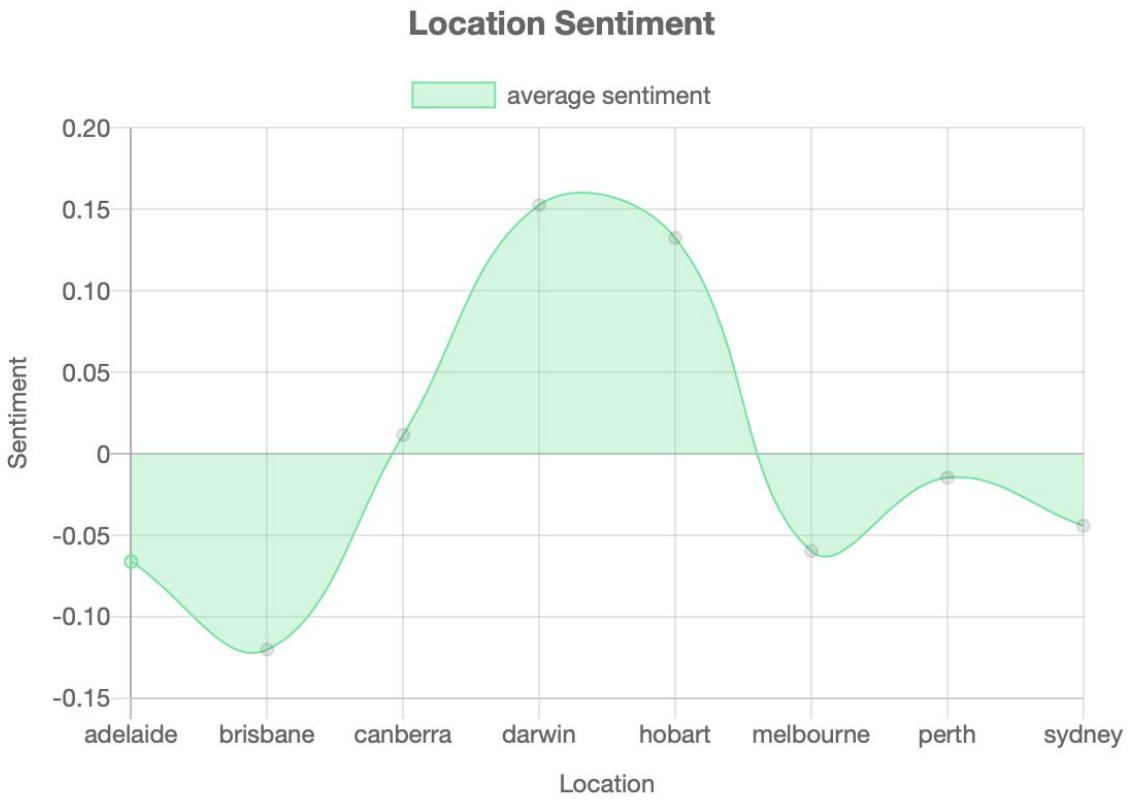
1. The internal networks may fail unexpectedly especially when accessing CouchDB instances.
2. Cloud users running on remote servers may have limited control over their works due to resources completely owned and managed by cloud service providers.

4. Our findings and Snapshots of the twitter data :-

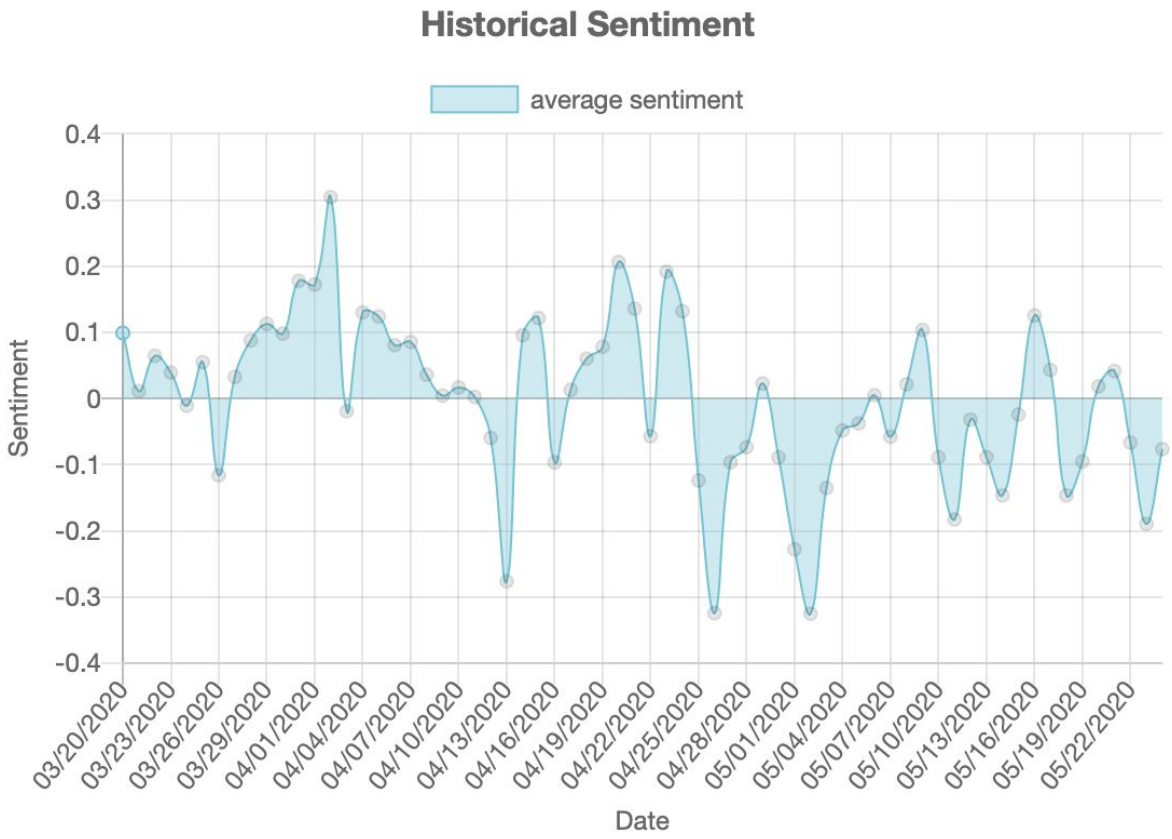
For a live view, our dashboard is available at http://172.26.134.121/

Our CouchDB instances are available at, 172.26.134.28, 172.26.130.35, 172.26.133.54
Username: admin
Password: password

When looking at the results of the sentiment analysis that was performed on tweets containing #jobseeker & #jobkeeper we see some interesting results. Firstly, the average sentiment for all cities was relatively close to 0 because the sentiment value could range from -1 to 1. This tells us that across the board, there was no clear opinion by Australians on this issue. For an economic policy expecting to total $320 billion, equivalent to 16.4% of Australia's GDP, there would be an expectation that there would be more support by the Australian public for such an extreme stimulus package. Further, if we look further into the location sentiment findings we can see that there is a split between major cities that on average had positive tweets and those with an average of negative tweets. Canberra, Darwin, and Hobart's tweet sentiments all averaged out to a positive value while the remaining cities had an overall negative average sentiment. If we take into consideration the numbers of tweets from each city we can see that Darwin and Hobart have barely any tweets in comparison to the other cities so with almost all the other cities registering negative average sentiment of tweets we can make the claim that overall Australia has a negative attitude towards the newly implemented jobkeeper and jobseeker economic packages.



Another interesting result when looking at the historic trend of tweets sentiment over time is that we notice a generally positive sentiment attached to these tweets initially when the policies were announced in March and then slowly with time we start to see that the tweets become more negative in nature. It can be noted that there appears to be a switch in sentiment from positive to negative around the 24th of April onwards. This is significant because on the 24th of April there was an announcement by the Australian Treasurer Josh Frydenberg stating that all employees that qualify for jobkeeper payments must be paid $1500 a fortnight by employers. Now it could be said that based on this event, there was a shift in sentiment towards these schemes as businesses were forced to outlay $1500 per employee every fortnight while having a decrease in revenue of at least 30%, this may have been the reason for the shift in tweet sentiment beyond this date.



Finally, when comparing this to the Aurin data which shows which industry has the most employees in each major city, we see that in most major cities the most common industry of work is 'Health Care And Social Assistance', while Darwin has its most employees in 'Construction' industry and finally Sydney having the most employees in 'Professional, Scientific, And Technical Services'. Since this data shows that most major cities have healthcare and social assistance as their most common industry accompanied by the fact

that these healthcare workers are not the ones typically going to need jobkeeper or jobseeker payments this might explain the average negative sentiment coming from these cities. Healthcare workers have been some of the hardest working Australians throughout this time so it may be reasonable to think that they would have a negative attitude towards others receiving government handouts for no additional work. Finally with Sydney's most common industry being 'Professional, Scientific And Technical Services' this might explain Sydney having the most number of tweets on the issue. It could potentially be said that those within this industry would have strong enough opinions on the governments implemented policies to tweet about it.