The global effects of Coivd-19 appearance and its relation to global events

Menghao Zhan
11708670

Abstract / Summary

At the end of 2019, a new type of coronavirus started spreading from China. COVID-19 symptoms can persist for months with a high probability and keep damaging the lungs, heart and brain, which increases the risk of long-term health problems. Therefore, this project aims to do research on the global short-term effects of Covid-19 and its relation to global events through an analysis of the GDELT 2.0 Events Database. The algorithms used in the analysis are the cosine-similarity algorithm and the Apriori algorithm. The expected result is that Covid-19 did not bring a huge effect around the world at the beginning and the most frequent global events still mostly relate to politics and tax. The expected result means that most people and governments around the world did not bring the appearance of Covid-19 to the forefront at the beginning of the Covid-19 spearing.

Introduction

Background

GDELT 2.0 is a database which contains a vast amount of world news and updates per fifteen minutes. In the database, each article contains its expression, topic, person, organization, place, theme, emotion, related images, etc. This project uses Pyspark to download, classify, compute, apply algorithms and analyze the data from the GDELT 2.0 database.

The algorithms used in this project include cosine-similarity and Apriori. The cosine-similarity algorithm is usually used to measure the similarity between two vectors in the inner product space. It is measured by the cosine of the angle between two vectors and determines whether the two vectors point in approximately the same direction. Therefore it is often used to measure document similarity in text analysis. The Apriori algorithm is the classical algorithm for finding the association of items, which finds the set of items that frequently appear together and computes the association of items within the

Motivation

I was in Wuhan before Covid-19 was posted by the mainstream media. Then, I was among the first to experience lockdown. However, until Covid-19 was confirmed to exist, it was not taken seriously. This led to the wide spread of Covid-19 after. Therefore, this project analyses global news reports through big data and algorithms to compare the changes in international news before and after Covid-19. After analyzing the results of the project objectively and subjectively, it can reflect the reality that I experienced. For this reason, the lecture notes exercises in Lab3 and Lab5 in DATA301 can be implemented in this project.

Research Question or Hypothesis

The research question is to figure out how the Covid-19 affects livings around the world at the beginning of Covid-19. The project addresses the research question through the analysis of the GDELT 2.0 Events Database. The cosine-similarity algorithm can use the V2Tone from the database to show the difference one month before and after Covid-19 was proved. The Apriori algorithm can show the relations between different V2Themes from the

database, Then we can find the supports, frequent items confidences, and interests to explore the facts they reflected.

Experimental Design and Methods

This project uses Google Colab to do the research. The first part of this project is to initialise the frame and the environment which the project uses. Setting up libraries, starting spark on the local server, then connecting the python driver to a local Spark JVM running on the Google Colab server virtual machine. After the steps above, the project starts to fetch data from the GDELT 2.0 Events database for two given date ranges and then downloads the data into files. This procedure uses multiprocess to process it more efficient. However, there is a bug when there is no data on some special date. To fix this bug, the codes are shown below to skip the vacant data.

```
for i in results_1:
 if i != None:
   results_1_fix_bug.append(i)
results_1_fix_bug
```

The data range before the Covid-19 proved is from 2019 Nov 27 to 2019 Dec 28, the date range after that is from 2019 Dec 28 to 2019 Jan 30. Both groups of data are separately defined as ‘data_before’ and ‘data_after’. Because they are in the Panda form, we can use ‘.show()’ to check the details in these two tables. To convert the form of both ‘data_before’ and ‘data_after’ from Panda to RDD with the required conditions, the map function is used and the codes shown below are implemented.

```
Locations_V2Tone_before = (data_before_rdd
                           .map(lambda a: (a.Locations, a.V2Tone)))
```

In this project four groups of data are required, they are ‘Locations_V2Tone_before’, ‘Locations_V2Tone_after’, ‘Locations_V2Themes_before’ and ‘Locations_V2Themes_after’. Because of the vast data around the world, only six countries, the United Kingdom, the United States, New Zealand, France, Canada, and China, are considered for this project. The ‘Location’ rows are a string containing several different explanations of counties, the function shown below is used to split out the required country codes in this project.

```
countrys = ['UK', 'US', 'NZ', 'FR', 'CA', 'CH']
#  Countrycode, Tone
def country_tone(string):
 if string != None:
   country_code = string.split('#')
   if len(country_code) >= 3:
     return country_code[2]
   else:
     return None
```

After filtering the data, the cosine-similarity algorithm can be used with these datasets ‘Countrycode_V2Tone_before’, ‘Countrycode_V2Tone_after’. First, the function ‘tidy_data’ converts strings into floats, and it only collects the floats without the last one. Because the last one is the count of tones in one day, not the actual tone from any article. Then, it accumulates all the tones and the sum of tones is divided by the number of total tones through the map function to get an integral number.

```
def tidy_data(data):
 result = []
```

```
 for string in data:
   floats = string.split(",")
   tone = 0
   for f in floats[:-2]:
     tone += float(f)
   result.append(tone)
 return result

Countrycode_V2Tone_before_t = Countrycode_V2Tone_before.map(lambda x:
(x[0], int(sum(tidy_data(x[1]))/ len(tidy_data(x[1])))))
```

The last step of this part is to use the cosine－similarity function to get the result.

```
def cosine_similarity(list_a,list_b):
 """a regular python function that computes cosine similarity between
two lists of floats"""
 numerator = 0
 denom_l = 0
 denom_r = 0
 for i in range(len(list_a)):
   numerator += list_a[i]*list_b[i]
   denom_l += list_a[i]**2
   denom_r += list_b[i]**2
 return numerator / (((denom_l)**(1/2))*((denom_r)**(1/2)))

before = [tone for country, tone in
Countrycode_V2Tone_before_t.collect()]
after = [tone for country, tone in
Countrycode_V2Tone_after_t.collect()]

cossim_before_after = cosine_similarity(before, after)
```

The second part of this project is to find how V2Themes of articles posted before and after Covid-19 proved relations to each other through the Apriori algorithm. When the project tries to find the frequent V2Themes in different countries, it can be found that there is a huge difference in the number of V2Themes. Hence, the minimum frequencies are set in different sizes in different countries. The outputs from this function are frequent items in RDD form, frequent items in dictionary form and baskets which are built by articles.

```
min_frequency = 500
def frequentitems_baskets(country_code, dataset):
 if country_code == 'NZ' or country_code == 'FR':
   min_frequency = 50
 elif country_code == 'US':
   min_frequency = 5000
 else:
   min_frequency = 500
```

```
 baskets = (dataset.filter(lambda x: x[0] == country_code).map(lambda
x: convert_baskets(x[1])))
frequentitems = (sc.parallelize(events_tidy(dataset
.filter(lambda x: x[0] == country_code).map(lambda x: x[1]).collect()))
.reduceByKey(lambda a, b: a+b).filter(lambda a: a[1] >= min_frequency))
 return frequentitems, frequentitems.collectAsMap(), baskets

def events_tidy(data):
 result = []
 for d in data:
   result += d
 return result

def convert_baskets(data):
 result = []
 for d in data:
   result.append(d[0])
 return result
```

Because the dataset of each country is still quite huge, the project decides to only check and use the top 5 data.

```
dbg(Themes_UK_be_nf.takeOrdered(5, lambda s: -1 * s[1]))
```

As for the Apriori algorithm, the project only uses it once and analyses the relation between two V2Themes. Moreover, the number of baskets in each country is still a lot. To speed up the computation, the function uses broadcast to make read-only copies of the huge data. Then, the copies are sent to all actuators at once and are cached for reference.

```
def combinate(broadcast_value , line):
 ans = []
 for i, word1 in enumerate(line):
   if word1 in broadcast_value:
     for j, word2 in enumerate(line):
       if i > j and word2 in broadcast_value:
         ans.append(tuple(sorted([word1, word2])))
 return ans

def apriori(frequentitems, baskets):
 broadcast_value = sc.broadcast(frequentitems).value
 themes_groups = (baskets
                  .flatMap(lambda x: combinate(broadcast_value, x))
                  .map(lambda a:(a, 1))
                  .reduceByKey(lambda a,b:a+b))
 return themes_groups
```

The Support of each 2VTheme is output by the function, 'support'.

```
def support(group_rdd, baskets):
 total = baskets.count()
```

```
 return group_rdd.map(lambda x: (x[0],x[1]/total))
```

The last second part of this project focuses on the confidence and interest of each frequent pair. Because the frequent items are filtered by different minimum values, also the size of data and baskets are different, there is an extra boolean sentence to set different proportions for different countries. These proportions can be used to adjust the accuracy of confidence and interest given by the functions before.

```
def confidence_interest(individual_pairs, Frequent_Themes, baskets):
 # interest = |confidence - Pr(y)|
 # Pr(y) = y_counts/total_baskets
 count_of_total_pairs = baskets.count()
 if count_of_total_pairs <= 500:
   proportion = 100
 elif count_of_total_pairs <= 5000:
   proportion = 20
 else:
   proportion = 10
 sup_individual = individual_pairs.map(lambda a:(a[0],
a[1]/count_of_total_pairs))
 sup_ij = Frequent_Themes.map(lambda a:(a[0][0], (a[0][1],
a[1]/count_of_total_pairs)))
 sup_ji = Frequent_Themes.map(lambda a:(a[0][1], (a[0][0],
a[1]/count_of_total_pairs)))
 sup_all_ij_and_ji = sup_ij.union(sup_ji)
 all = sup_all_ij_and_ji.join(sup_individual)
 confidence = all.map(lambda a: ((a[0], a[1][0][0]),
a[1][0][1]/a[1][1]/proportion))
 interest = confidence.map(lambda x: (x[0][1],
x)).join(sup_individual).map(lambda x: (x[1][0][0], x[1][0][1] -
x[1][1]))
 return confidence, interest
```

Moreover, during the implementation of this program, the time elapsed should be recorded to show how the efficiency of parallel computing is while facing the vast data. The function shown below helps the project record the elapsed time while the program running.

```
import time
start_time = time.monotonic()
end_time = time.monotonic()
print("The time used to allocate Country, 2VTone and 2VThemes before
and after Covid",end_time-start_time)
```

show how the efficiency of parallel computing is while facing the vast data. The function

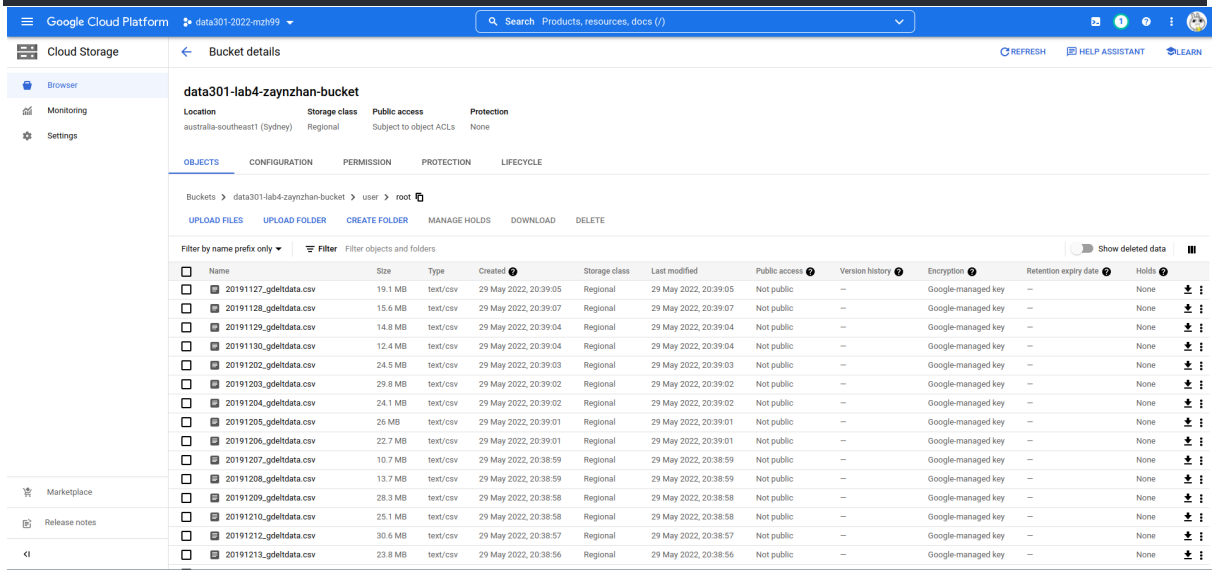
In addition, this project also explores cloud computing and its scalability by using the Google Cloud Platform. For this purpose, the project follows the instruction in the DATA301 course and applies the content of lab exercises to the cloud computing in this project.

In Google Colab the code below zips all data required for cloud computing and makes it easy to download them together at once.

```
!zip -o mydata.zip *.csv
```

When the downloading is finished, the zipped file is unzipped and uploaded to the bucket created by the code shown below in the Cloud Storag.

```
REGION=australia-southeast1
ZONE=australia-southeast1-a
PROJECT=data301-2022-mzh99
CLUSTER=data301-lab4-zaynzhan-cluster
BUCKET=data301-lab4-zaynzhan-bucket
```



In the Cloud Shell, copy the codes form Google Clolab and paste them into ‘pyspark_wc.py’ file and add the following codes to guarantee that the files in the bucket can be read.

```
test_data_before = ['20191127_gdeltdata.csv', '20191128_gdeltdata.csv', '20191129_gdeltdata.csv', '20191
test_data_after = ['20191229_gdeltdata.csv', '20191230_gdeltdata.csv', '20191231_gdeltdata.csv', '202001

test_data_before_n = ['gs://data301-lab4-zaynzhan-bucket/' + filename for filename in test_data_before]
test_data_after_n = ['gs://data301-lab4-zaynzhan-bucket/' + filename for filename in test_data_after]
sqlContext = SQLContext(sc)

data_before = sqlContext.read.option("header", "true").csv(test_data_before_n)
data_before_rdd = data_before.rdd
data_after = sqlContext.read.option("header", "true").csv(test_data_after_n)
data_after_rdd = data_after.rdd
```

Through changing the value ‘x’ of "n1-standard-x" in the "worker_config" in the python file ‘submit_job_to_cluster.py’, the project can achieve that data can be allocated in to different number of processors for computation. The number of processors against twice of value ‘x’. After running the code to submit the job to the clusters, the result will show the time it spent as ‘elapsed time’.

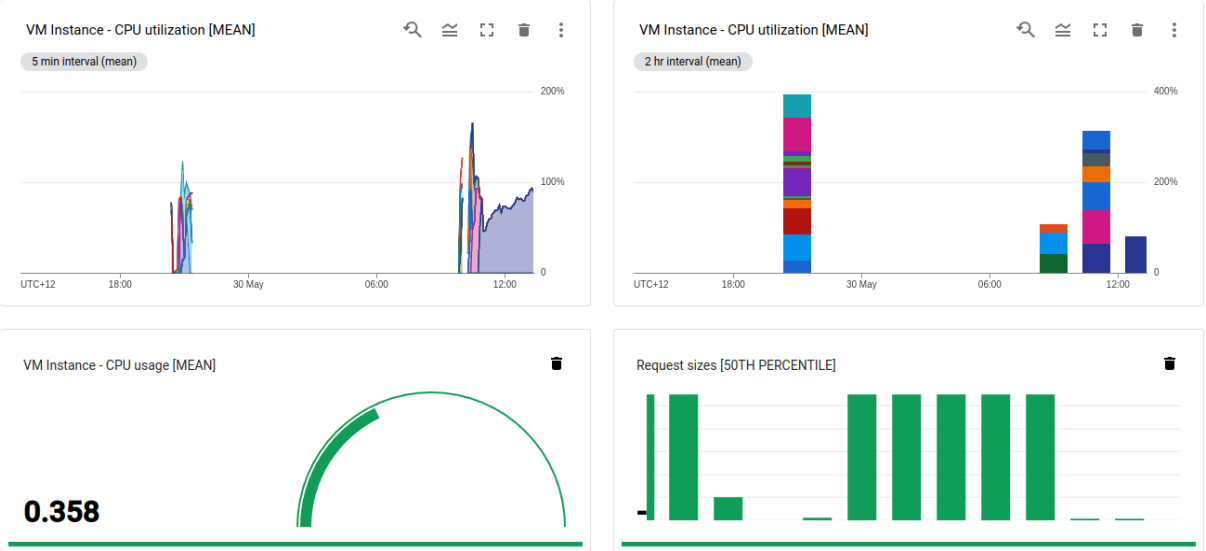
```
zaynzhan@cloudshell:~ (data301-2022-mzh99)$ python3 submit_job_to_cluster.py --project_id=$PROJ
 new_cluster --pyspark_file pyspark_wc.py
Creating cluster...
Waiting for cluster creation...
Cluster created.
Uploading pyspark file to Cloud Storage.
State.RUNNING
data301-lab4-zaynzhan-cluster - state: RUNNING
state_start_time {
  seconds: 1653863453
  nanos: 862824000
```

elapsed time is 388.68203616142273\n22/05/29 22:25:14

Results

The project analyses the efficiency and scalability of cloud computing with different numbers of processors used through a custom dashboard which is offered by the Google Cloud Platform.

Overall dashboard

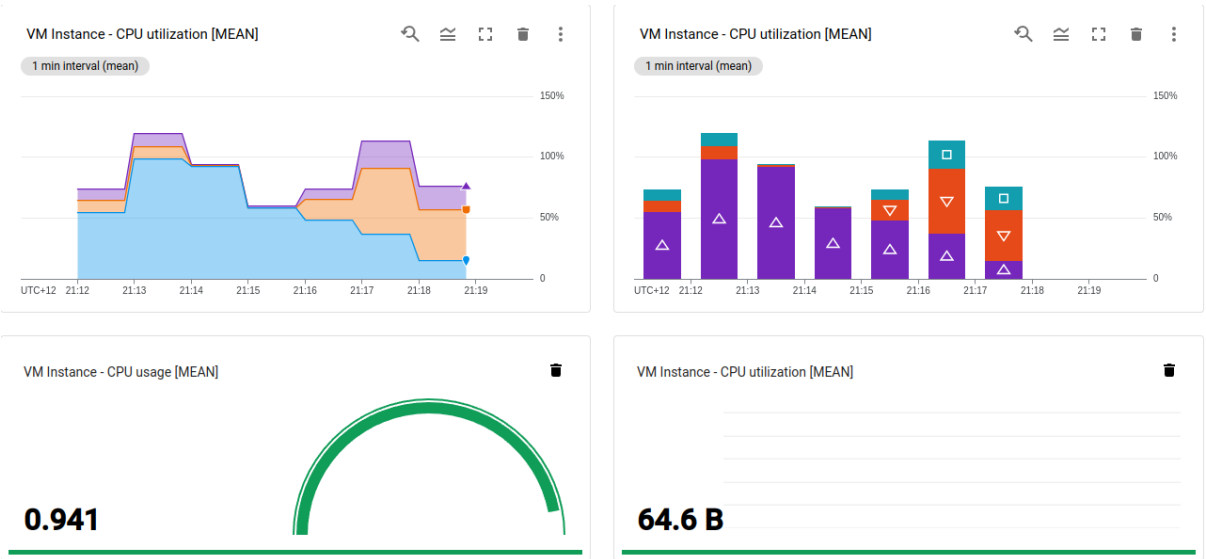


The overall dashboard shows all details of VM Instance-CPU utilization as a line chart and a stacked bar chart at top left and right corner separately. Two charts on the second row shows more information about the CPU usage and request sizes.

Elapsed time and carts of 16 vCPUs

Elapsed time is 140.41171669…s

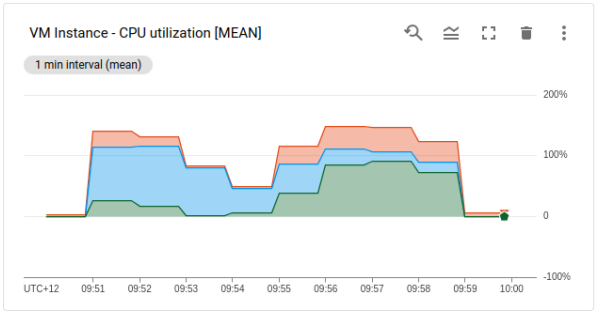
unknown; jvm 1.8.0_332-b09\n22/05/29 09:15:24 INFO org.spark_project.jetty.server.Server: Started @6278ms\n22/05/29 09:15:24 INFO org.spark_project.jetty.server.AbstractConnector: Started ServerConnector@67036991{HTTP/1.1, (http/1.1)}{0.0.0.0:34459}\n22/05/29 09:15:26 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at data301-lab4-zaynzhan-cluster-m/10.152.0.54:8032\n22/05/29 09:15:27 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at data301-lab4-zaynzhan-cluster-m/10.152.0.54:10200\n22/05/29 09:15:27 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found\n22/05/29 09:15:27 INFO org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find 'resource-types.xml'.\n22/05/29 09:15:27 INFO org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE\n22/05/29 09:15:27 INFO org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE\n22/05/29 09:15:32 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted application application_1653815576353_0001\n22/05/29 09:16:17 WARN org.apache.spark.scheduler.TaskSetManager: Stage 24 contains a task of very large size (232 KB). The maximum recommended task size is 100 KB.\n22/05/29 09:16:22 WARN org.apache.spark.scheduler.TaskSetManager: Stage 28 contains a task of very large size (198 KB). The maximum recommended task size is 100 KB.\n22/05/29 09:16:28 WARN org.apache.spark.scheduler.TaskSetManager: Stage 32 contains a task of very large size (2624 KB). The maximum recommended task size is 100 KB.\n22/05/29 09:16:35 WARN org.apache.spark.scheduler.TaskSetManager: Stage 36 contains a task of very large size (3232 KB). The maximum recommended task size is 100 KB.\n22/05/29 09:16:58 WARN org.apache.spark.scheduler.TaskSetManager: Stage 56 contains a task of very large size (237 KB). The maximum recommended task size is 100 KB.\n22/05/29 09:17:03 WARN org.apache.spark.scheduler.TaskSetManager: Stage 60 contains a task of very large size (239 KB). The maximum recommended task size is 100 KB.\n22/05/29 09:17:07 WARN org.apache.spark.scheduler.TaskSetManager: Stage 64 contains a task of very large size (201 KB). The maximum recommended task size is 100 KB.\n22/05/29 09:17:12 WARN org.apache.spark.scheduler.TaskSetManager: Stage 68 contains a task of very large size (350 KB). The maximum recommended task size is 100 KB.\nelapsed time is 140.41171669960822\n22/05/29 09:18:04 INFO org.spark_project.jetty.server.AbstractConnector: Stopped Spark@67036991{HTTP/1.1, (http/1.1)}{0.0.0.0:0}\n"

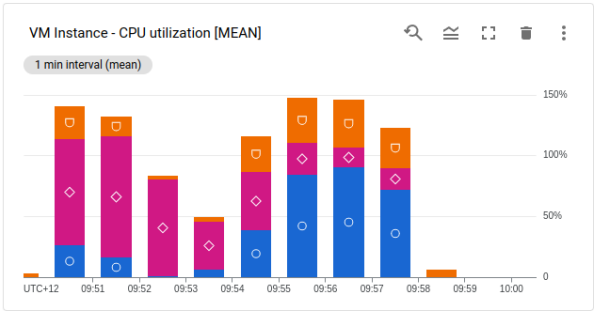


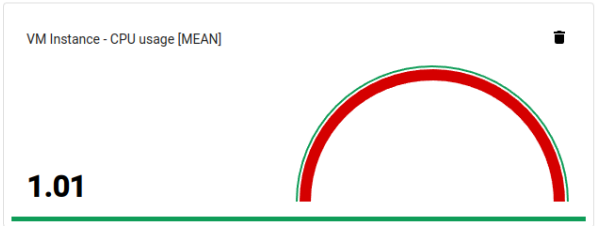
Elapsed time and carts of 8 vCPUs

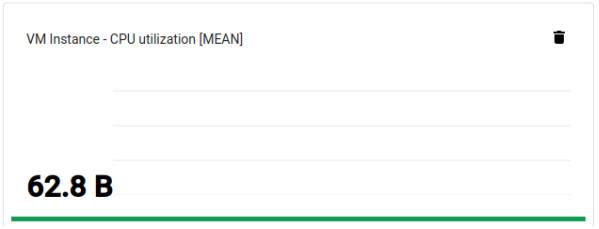
Elapsed time is 218.39725160…s

```
Adding resource type - name = vcores, units = , type = COUNTABLE\n22/05/29 21:54:04 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted applicatio
n application 1653861090545 0001\n22/05/29 21:55:01 WARN org.apache.spark.scheduler.TaskSetManager: Stage 24 contains a task of very large size (494 KB). The maximum
 recommended task size is 100 KB.\n22/05/29 21:55:10 WARN org.apache.spark.scheduler.TaskSetManager: Stage 28 contains a task of very large size (388 KB). The maximu
m recommended task size is 100 KB.\n22/05/29 21:55:19 WARN org.apache.spark.scheduler.TaskSetManager: Stage 32 contains a task of very large size (5282 KB). The maxi
mum recommended task size is 100 KB.\n22/05/29 21:55:29 WARN org.apache.spark.scheduler.TaskSetManager: Stage 36 contains a task of very large size (6507 KB). The ma
ximum recommended task size is 100 KB.\n22/05/29 21:55:37 WARN org.apache.spark.scheduler.TaskSetManager: Stage 40 contains a task of very large size (110 KB). The m
aximum recommended task size is 100 KB.\n22/05/29 21:56:08 WARN org.apache.spark.scheduler.TaskSetManager: Stage 56 contains a task of very large size (465 KB). The
maximum recommended task size is 100 KB.\n22/05/29 21:56:16 WARN org.apache.spark.scheduler.TaskSetManager: Stage 60 contains a task of very large size (507 KB). The
 maximum recommended task size is 100 KB.\n22/05/29 21:56:22 WARN org.apache.spark.scheduler.TaskSetManager: Stage 64 contains a task of very large size (431 KB). Th
e maximum recommended task size is 100 KB.\n22/05/29 21:56:30 WARN org.apache.spark.scheduler.TaskSetManager: Stage 68 contains a task of very large size (739 KB). T
he maximum recommended task size is 100 KB.\nElapsed time is 218.397251605987555\n22/05/29 21:57:56 INFO org.spark_project.jetty.server.AbstractConnector: Stopped Spa
rk@3dc6967e{HTTP/1.1, (http/1.1)}{0.0.0.0:0}\n"
Tearing down cluster.
zaynzhan@cloudshell:~ (data301-2022-mzh99)$
```





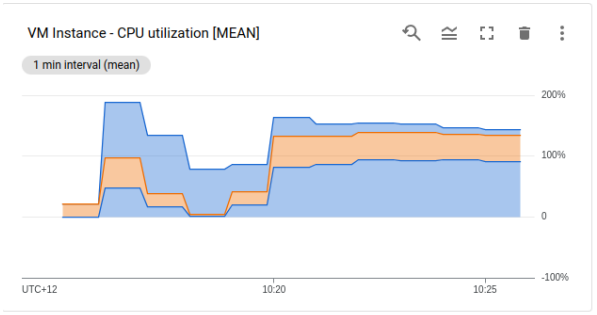


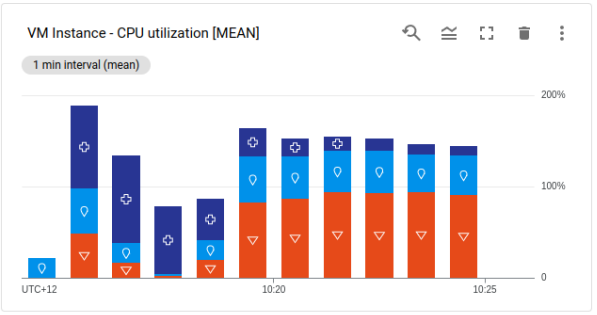


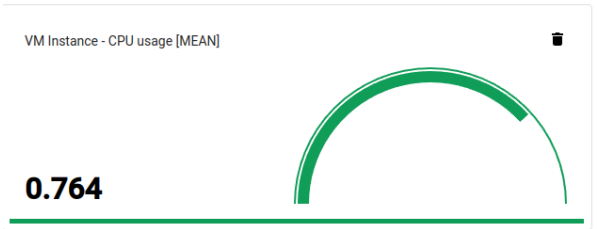
Elapsed time and carts of 4 vCPUs

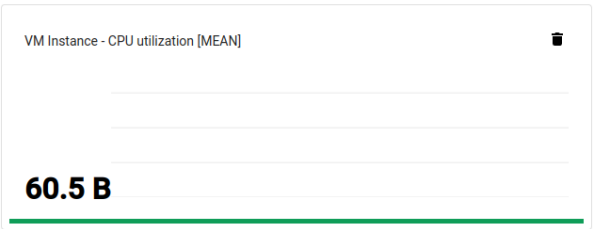
Elapsed time is 388.68203616…s

```
m recommended task size is 100 KB.\n22/05/29 22:20:17 WARN org.apache.spark.scheduler.TaskSetManager: Stage 32 contains a task of very large size (10605 KB). The max
imum recommended task size is 100 KB.\n22/05/29 22:20:34 WARN org.apache.spark.scheduler.TaskSetManager: Stage 36 contains a task of very large size (13039 KB). The
 maximum recommended task size is 100 KB.\n22/05/29 22:20:49 WARN org.apache.spark.scheduler.TaskSetManager: Stage 40 contains a task of very large size (163 KB). The
 maximum recommended task size is 100 KB.\n22/05/29 22:21:04 WARN org.apache.spark.scheduler.TaskSetManager: Stage 44 contains a task of very large size (161 KB). Th
e maximum recommended task size is 100 KB.\n22/05/29 22:21:16 WARN org.apache.spark.scheduler.TaskSetManager: Stage 48 contains a task of very large size (120 KB). T
he maximum recommended task size is 100 KB.\n22/05/29 22:21:31 WARN org.apache.spark.scheduler.TaskSetManager: Stage 52 contains a task of very large size (121 KB).
The maximum recommended task size is 100 KB.\n22/05/29 22:21:44 WARN org.apache.spark.scheduler.TaskSetManager: Stage 56 contains a task of very large size (928 KB).
 The maximum recommended task size is 100 KB.\n22/05/29 22:21:59 WARN org.apache.spark.scheduler.TaskSetManager: Stage 60 contains a task of very large size (1008 KB
). The maximum recommended task size is 100 KB.\n22/05/29 22:22:12 WARN org.apache.spark.scheduler.TaskSetManager: Stage 64 contains a task of very large size (895 K
B). The maximum recommended task size is 100 KB.\n22/05/29 22:22:27 WARN org.apache.spark.scheduler.TaskSetManager: Stage 68 contains a task of very large size (1482
 KB). The maximum recommended task size is 100 KB.\nElapsed time is 388.682036161422273\n22/05/29 22:25:14 INFO org.spark_project.jetty.server.AbstractConnector: Stop
ped Spark@779c0b1f{HTTP/1.1, (http/1.1)}{0.0.0.0:0}\n"
Tearing down cluster.
zaynzhan@cloudshell:~ (data301-2022-mzh99)$
```





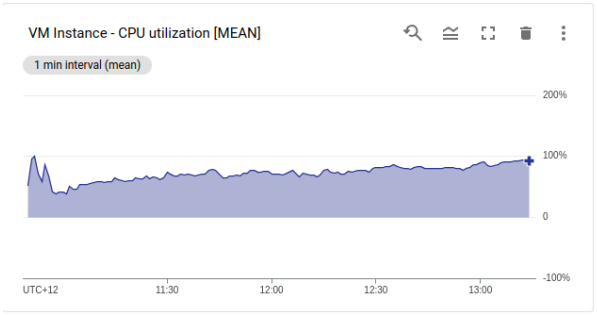


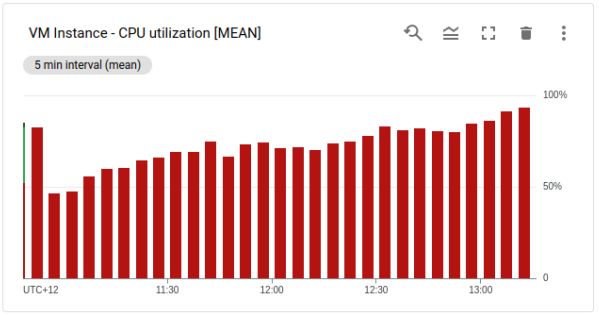


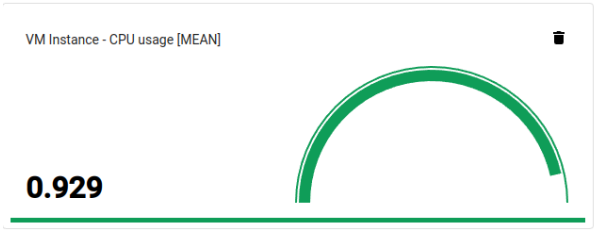
Elapsed time and carts of 2 vCPUs

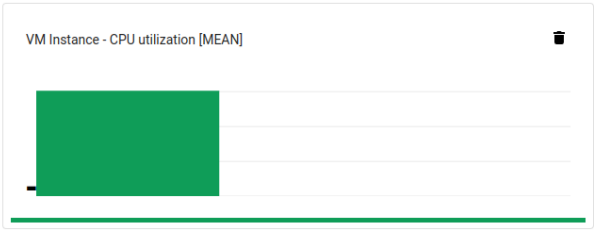
Elapsed time is 794.1356618…s

```
maximum recommended task size is 100 KB.\n22/05/30 02:01:09 WARN org.apache.spark.scheduler.TaskSetManager: Stage 32 contains a task of very large size (15890 KB). T
he maximum recommended task size is 100 KB.\n22/05/30 02:01:47 WARN org.apache.spark.scheduler.TaskSetManager: Stage 36 contains a task of very large size (19562 KB)
. The maximum recommended task size is 100 KB.\n22/05/30 02:02:17 WARN org.apache.spark.scheduler.TaskSetManager: Stage 40 contains a task of very large size (240 KB
). The maximum recommended task size is 100 KB.\n22/05/30 02:02:50 WARN org.apache.spark.scheduler.TaskSetManager: Stage 44 contains a task of very large size (237 K
B). The maximum recommended task size is 100 KB.\n22/05/30 02:03:17 WARN org.apache.spark.scheduler.TaskSetManager: Stage 48 contains a task of very large size (159
KB). The maximum recommended task size is 100 KB.\n22/05/30 02:03:50 WARN org.apache.spark.scheduler.TaskSetManager: Stage 52 contains a task of very large size (158
 KB). The maximum recommended task size is 100 KB.\n22/05/30 02:04:18 WARN org.apache.spark.scheduler.TaskSetManager: Stage 56 contains a task of very large size (13
86 KB). The maximum recommended task size is 100 KB.\n22/05/30 02:04:51 WARN org.apache.spark.scheduler.TaskSetManager: Stage 60 contains a task of very large size (
1546 KB). The maximum recommended task size is 100 KB.\n22/05/30 02:05:20 WARN org.apache.spark.scheduler.TaskSetManager: Stage 64 contains a task of very large size
 (1361 KB). The maximum recommended task size is 100 KB.\n22/05/30 02:05:53 WARN org.apache.spark.scheduler.TaskSetManager: Stage 68 contains a task of very large si
ze (2221 KB). The maximum recommended task size is 100 KB.\nelapsed time is 794.1356618404388\n22/05/30 02:11:52 INFO org.spark project.jetty.server.AbstractConnecto
r: Stopped Spark@67036991{HTTP/1.1, (http/1.1)}{0.0.0.0:0}\n"
Tearing down cluster.
zaynzhan@cloudshell:~ (data301-2022-mzh99)$
```

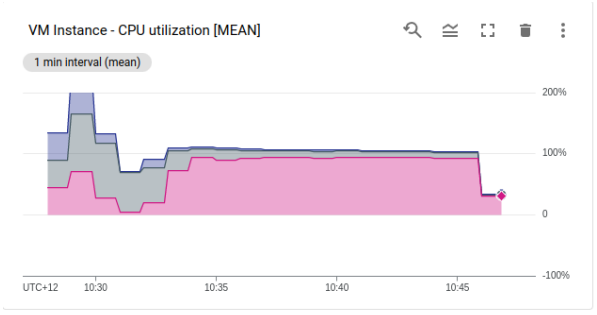


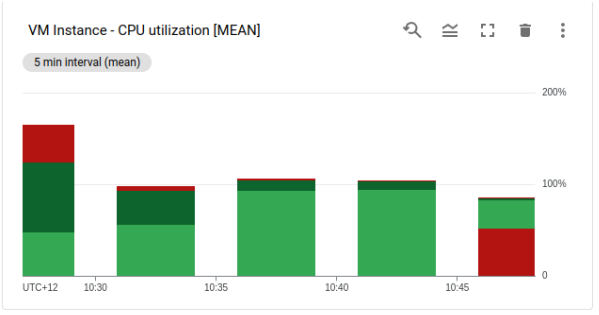


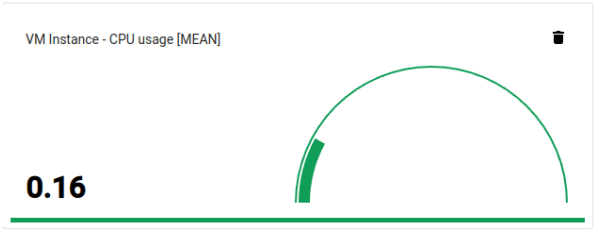


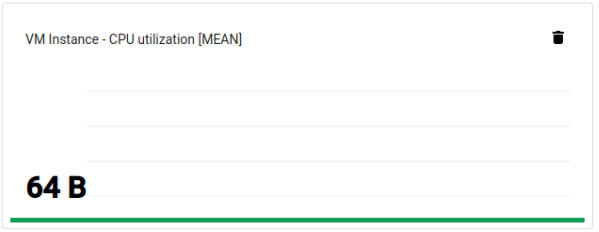
Elapsed time and carts of sequential computation

Elapsed time is given by the sum of elapsed time in each part of Colab, around 576.7254s









More processors vastly seedup the computing, however if the number of processors is too small, sometimes it is less efficient than the sequential computing.

The outputs from Colab are shown below.

The cosine-similarity between tones before and after the Covid-19 proved in the countries which this project based on is 0.9997569330516283. Hence, the result shows that in the short period, the Covid-19 did not affect people's livings alot around the world. The difference of negative and positive effects given by the globa events are quite similar between the one month before and after the Covid-19.

A month after the Covid-19 was proved, there is a new theme increased vastly in China, 'TAX_DISEASE_CORONAVIRUS' its support a month after Covid started is 0.857760691975012, it was the top theme in that period. However, in other countries, it is still an unimportant topic. For all countries, ‘CRISISLEX_CRISISLEXREC’ and ‘CRISISLEX_C07_SAFETY’ are always hot themes whenever it is. The supports for these two themes are always over 0.55. Moreover, themes about tax are also the top topic in most countries, most of them have support above 0.8. Most in these six countries have some changes in the themes of articles before and after Covid-19 proved, except America. It means the effect on America is the least one in the short term among all six countries.
The result of confidence and interest shows that the risk of impeachment of the leader in the US increased after the Covid-19 started. ‘MANMADE_DISASTER_IMPLIED’ increased. More and more themes of articles involve wellbeing, health, and medical around the words. As for education in Canada, ‘SOC_POINTSOFINTEREST_SCHOOL’ happens more in the educational articles. Energy problems were continuously related to policies and industries in China.

Conclusion

The results obtained from this project not only provide reliable answers to the research question set before the project began, but also yield additional information beyond the question under study. After analysing the results, it is clear that Covid-19 has only had a significant impact on the lives of the Chinese people within a month of its inception. However, other countries were not affected a lot and they were not alerted to this. It is even possible to detect here a foretaste of the worldwide spread that followed. The above conclusion can be figured out from the support of the article themes in the month before and after the confirmation of Coivd-19 and the confidence between the pair of two themes. It is also clear that the American election was also deeply affected by the festering epidemic of Covid-19.

My results are good evidence that previous events are indeed related to the Coivd-19 epidemic. When Covid-19 was just proved, if countries around the world had prepared in time for the disaster that was coming soon, the current epidemic situation might have been far better than the reality.

Based on this project, it is possible to validate some of the arguments by applying other algorithms to other databases to trace the origin of the Covid-19 outbreak. In addition, big data can be used to analyse this Covid-19 outbreak to predict when and where the next major outbreak will take place. As a result, better response decisions will be made in the future.

Critique of Design and Project

The cosine-similarity algorithm part is too sweeping and not particularly well used in the analysis of V2Tones. An optimised solution would be to first put all the emergent V2Themes in a list and compare the value sum of V2Tone for each V2Them in the month before and after the Covid-19 outbreak. Otherwise, there are still some scaling problems in the algorithms for confidence and interest computations that do not lead to the expected results.

Reflection

During this project, I have made extensive use of what I have learnt in DATA301. I used Pyspark parallel computing functions such as, ‘sc.’, ‘map’, and ‘Reducbykey’... They are enabled me to become more proficient in using the cosine-similarity algorithm and the Apriori algorithm in Pyspark that I have learnt in class. in Pyspark. I have also become more familiar with the use of Google Colab and Cloud Platform. In addition, I have gained a deeper understanding of the efficiency and convenience of Cloud computing.

References

COVID-19 (coronavirus): Long-term effects. (2021, October 22). Mayo Clinic. https://www.mayoclinic.org/diseases-conditions/coronavirus/in-depth/coronavirus-long-term-effects/art-20490351

Sharma, V. K., & Mittal, N. (2016). Exploiting Parallel Sentences and Cosine Similarity for Identifying Target Language Translation. Procedia Computer Science, 89, 428–433. https://doi.org/10.1016/j.procs.2016.06.092

Announcement (ScienceDirect Article in Press). (2004). Assessing Writing, 9(1), III–IV. https://doi.org/10.1016/s1075-2935(04)00016-9

Lab 3, Lab 4, Lab5 and Contents from slides, Zhenyuan He