PROJECT PROGRESS REPORT

What is the similarity in average sentiment of health-related events over the last few years between New Zealand and other countries?

May 18, 2022

Lily Williams
42415299

Contents

0.1 Overview . . . 2

0.1.1 Issues with Initial Proposal . . . 2

0.1.2 New Proposal . . . 2

0.1.2.1 Methods . . . 3

0.2 Completed Tasks . . . 4

0.2.1 Identified the New Plan . . . 4

0.2.2 Acquired Data . . . 4

0.2.3 Analyse the Tone use Map-Reduce Algorithms . . . 8

0.2.4 Calculate Similarity Between Vectors . . . 9

0.3 Next Steps . . . 10

0.3.1 Complete getTone Function . . . 10

0.3.2 Analyse All Data . . . 10

0.3.3 Analyse Similarities . . . 10

0.3.4 Present Results . . . 10

0.4 Difficulties . . . 11

0.1 Overview

0.1.1 Issues with Initial Proposal

The initial proposal stated that the GDELT 2.0 database was going to be downloaded and the data filtered by the CAMEO "type" code, identifying only “HLH” or health-related events. Following this, the data would be sorted into countries using the “Actor1CountryCode” entry. This turned out to be impossible, as there is no “type” code; the only filterable “type” is that of the Actor codes. This would result in only being able to analyze data put out by health-related organisations, rather than filtering by health-related events. Clearly, this is an issue as it implies that the entire crux of this project is not possible by the initially proposed method. Fortunately, there was another method found for determining the same information.

0.1.2 New Proposal

The new proposal makes use of the GDELT 2.0 Summary feature, which allows the user to search specific keywords appearing in news reports. Additionally, it has a built in country, and date selecting filter. Additionally, this summary has a built-in tone function which buckets the approximate tone of the articles on that day, producing an easily readable and, most-importantly, a downloadable CSV which contains these buckets. 1 displays an example of this tone chart. The keywords for this example was “health -mental”, the country was New Zealand, and the date was 01 March 2019, the first day of my proposed sample range. 1 shows a small section of the data conatined within the associated CSV file.

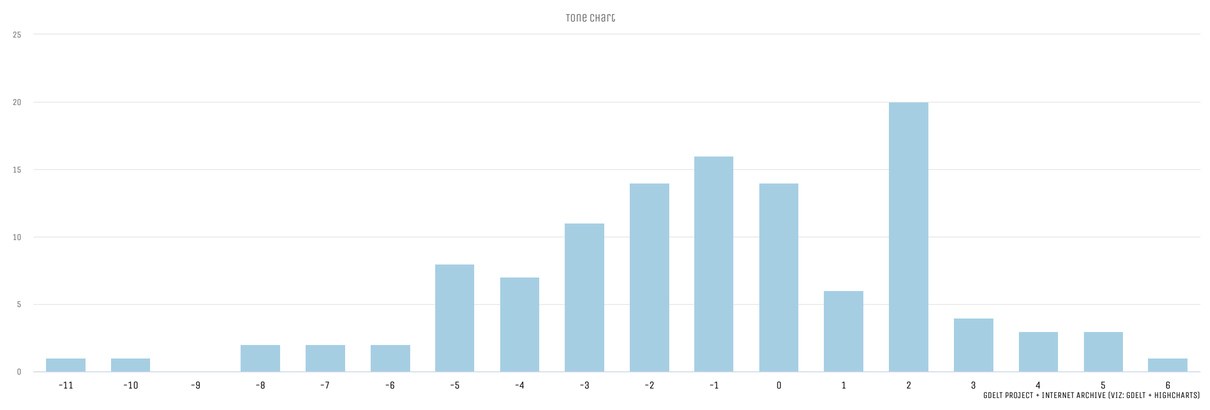

Figure 1: This is the tone chart of health-related events on 01/03/2019 in New Zealand.

Table 1: A small section of the data contained within the CSV for the example tone data shown in 1.

Label	Count	TopArtURL1	...
-11	1	https://www.nzherald.co.nz/...	...
-10	1	https://thedailyblog.co.nz/2019/03/01/...	...
-9	0		...
-8	2	https://livenews.co.nz/2019/03/01/...	...
-7	2	https://www.stuff.co.nz/national/crime/110967765/...	...
-6	2	https://i.stuff.co.nz/national/education/110954487/...	...
-5	8	https://www.nzherald.co.nz/nz/news/...	...
-4	7	https://www.nzherald.co.nz/northern-advocate/...	...
-3	11	https://www.stuff.co.nz/national/110958855/...	...
...	...	...	...

0.1.2.1 Methods

Using GDELT's Summary feature, individual CSV files can be acquired for each country for each day within the desired date range. This will be done by editing the URL to contain the appropriate country code, keywords, and date range. The country codes will be "NZ", "US", "UK", "CH", "RS", and "SF", the keywords will be "health -mental", and the dates will range in sections of one day from "20190301000000" to "20220401235959" in the YYYYMMDDHHMMSS format. Each country will have its own RDD containing the CSV files of the tone buckets for each day, resulting in 6 RDDs of 1128 CSV files each. Once these files have been pulled from GDELT's database, each will be parallelised and mapped to only retain the first two columns of data, "Label" and "Count". Another map will find the product of these two columns, sum these up over every column in the CSV, and divide by the total number of articles, giving an average tone for each day. Finally, this average tone will be mapped to one of the five integers on the sentiment scale as mentioned in the initial proposal. From there, the method will remain the same as before; the tones for each country will be sorted by month and a vector created, then the similarity, S, between New Zealand and the five other countries' tone vectors will be calculated using the cosine similarity formula:

$$S = \frac{u \cdot v}{||u|| * ||v||}$$

0.2 Completed Tasks

As of **May 18, 2022**, the following tasks have been completed:

0.2.1 Identified the New Plan

The first task that was completed was identifying the deliverables in the course, and writing a new plan to answer the research question. This plan is detailed below:

- Using the GDELT Summary Search Function
  - Keywords= “health” -mental
  - Date Range = Monthly from 20190301000000 to 20220401235959
  - Countries = NZ, UK, US, CH, RS, SF
- Download the tone charts as a .csv
  - Naming convention "NZ_20220301_tones.csv"
- Read each country .csv into an RDD
- MapReduce to sum the tones over each month for each country
- Recall all the RDDs into lists (vectors)
  - Should be dimension 37 (37 months)
- Use the cosine similarity formula to calculate the similarity of each country to New Zealand
- Graph the results

0.2.2 Acquired Data

In order to acquire the data, the necessary libraries were setup; the code to do this was taken primarily from **James Atlas’s** example code Colabs, and were also used in all the labs. This code is shown below.

```
#library and code setup
!apt-get update
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!pip install -q pyspark
```

```

import pyspark, os, sys
from pyspark import SparkConf, SparkContext
from concurrent.futures import ProcessPoolExecutor
from datetime import date, timedelta
from pyspark.sql import SQLContext
import pandas as pd
import urllib.request
from operator import add
import time
import numpy as np


os.environ["PYSPARK_PYTHON"]="python3"
os.environ["JAVA_HOME"]="/usr/lib/jvm/java-8-openjdk-amd64/"
```

```
#start spark local server


os.environ["PYSPARK_PYTHON"]="python3"

#connects our python driver to a local Spark JVM running on the Google
    Colab server virtual machine
try:
  conf = SparkConf().setMaster("local[*]").set("spark.executor.memory", "1g
    ")
  sc = SparkContext(conf = conf)
except ValueError:
  #it's ok if the server is already started
  pass

def dbg(x):
  """ A helper function to print debugging information on RDDs """
  if isinstance(x, pyspark.RDD):
    print([(t[0], list(t[1]) if
            isinstance(t[1], pyspark.resultiterable.ResultIterable) else t
    [1])
           if isinstance(t, tuple) else t
           for t in x.take(100)])
  else:
    print(x)
```

After the necessary spark contexts were setup, the GDELT data was pulled down from the server using the following code. This code was adapted from **James Atlas's** sample project codes, but many changes were made to accommodate the different tasks. At the end of this block, all the CSV files were read into RDD files, with one RDD per country.

```
# Remove files from previous runs
!rm -rf articles
!rm *.csv

# Multiprocess the query
e = ProcessPoolExecutor()

### Example URL Format ###
# "https://api.gdeltproject.org/api/v2/doc/doc?format=csv&startdatetime
    =20190301000000&enddatetime=20220401235959&query=%22health%22%20-mental
    %20sourcecountry:NZ&mode=tonechart"
###

URLbase1 = "https://api.gdeltproject.org/api/v2/doc/doc?format=csv"
# startDate= "&startdatetime=" + queryDate + "000000"
# queryDate = "20190301"          (Example)
# endDate = "&enddatetime=" + queryDate +"235959"
URLbase2 = "&query=%22health%22%20-mental%20sourcecountry:"
#queryCountry = "NZ"
URLbase3 = "&mode=tonechart"

# queryURL = URLbase1 + startDate + endDate + URLbase2 + queryCountry +
    URLbase3




# Pull down the Tone files as .csv for each country and each date
def getFilename(x, countryCode):
  date = x.strftime('%Y%m')
  #print(date)
  #print(countryCode)
  return "{}_{}_tones.csv".format(countryCode, date)

def intoFile(filename):
    try:
        if not os.path.exists(filename):
          queryDate = filename.split("_")[1]
          queryCountry =filename.split("_")[0]
          startDate= "&startdatetime=" + queryDate + "01000000"
          endDate = "&enddatetime=" + queryDate +"28235959"

          queryURL = URLbase1 + startDate + endDate + URLbase2 + queryCountry + URLbase3
          print(queryURL)

          with urllib.request.urlopen(queryURL) as testFile, open(filename, 'w') as f:
            f.write(testFile.read().decode())
        return filename
    except Exception as inst:
        print(type(inst))    # the exception instance
        print(inst.args)     # arguments stored in .args
        print("Error occurred")

# Pull the data from GDELT into multi files; this may take a long time
countries = ['NZ', 'US', 'UK', 'CH', 'RS', 'SF']

resultList = []
for countryCode in countries:

  dateRange = [getFilename(x, countryCode) for x in pd.period_range('2019 March 1','2019 April 1', freq='M')]
  resultList.append(list(e.map(intoFile, dateRange)))
```

```
#dbg(resultList)

NZCSV = resultList[0]
USCSV = resultList[1]
UKCSV = resultList[2]
CHCSV = resultList[3]
RSCSV = resultList[4]
SFCSV = resultList[5]

dbg(NZCSV)

####
# Put downloaded files into RDDs
####

sqlContext = SQLContext(sc)

```

```
NZRDD = sqlContext.read.option("header", "true").csv(NZCSV)
USRDD = sqlContext.read.option("header", "true").csv(USCSV)
UKRDD = sqlContext.read.option("header", "true").csv(UKCSV)
CHRDD = sqlContext.read.option("header", "true").csv(CHCSV)
RSRDD = sqlContext.read.option("header", "true").csv(RSCSV)
SFRDD = sqlContext.read.option("header", "true").csv(SFCSV)
#dbg(NZRDD)
```

0.2.3 Analyse the Tone use Map-Reduce Algorithms

While not all the necessary functions have been completed, a number of the functions were written. This means that once all the map-reduce work has been done, the data needs only to be passed into the functions and all the required information will have been acquired to answer the research question. This code is shown below. The getTone function, which is arguably the most important, is still unfinished.

```
def getTone(data):
  '''
  Takes a dataframe containing the CSVs of each country
  Uses Map-Reduce algorithms to find the weighted average tone for each CSV
  Returns a list of the average tones for each csv
  '''
  #### INCOMPLETE ###
  # DOESNT FIND TONE FOR EACH CSV, FINDS TONE FOR ALL CSVs
  ####

  # Sum the total articles within each csv (sum of 'Count' column)
  totalArticles = data.rdd.map(lambda row: (1, int(row['Count']))).
    reduceByKey(lambda a, b: a + b).map(lambda x: x[1]).collect()

  # Sum the Label*Count Rows (Sum of Row1*Row2)
  countryTone = data.rdd.map(lambda row: (1, (int(row['Label'])*int(row['
    Count'])))).reduceByKey(lambda a, b: a+b)

  # Find the (weighted) average tone and allocate to and integer on the
    tone scale
  countryTone = countryTone.map(lambda tone: toneOnScale(tone[1]/
    totalArticles[0]))

  return countryTone.collect()


```

```
def toneOnScale(tone):
  '''
  Takes a float of the tone
  Matches it to an integer tone on the scale
  Returns the integer tone
  '''
  if tone > 2:
    #Tone is Very Positive
    return 2
  elif tone <= 1.5 and tone > 0.5:
    # Tone is positive
    return 1
  elif tone <= 0.5 and tone > -0.5:
    # Tone is neutral
    return 0
  elif tone <= -0.5 and tone > -1.5:
    # Tone is negative
    return -1
  else:
    # tone is Very Negative
    return -2

rddNames = [NZRDD, USRDD, UKRDD, CHRDD, RSRDD, SFRDD]
toneVectors = []
for name in rddNames:
  toneVectors.append(getTone(name))

NZtone = toneVectors[0]
UStone = toneVectors[1]
UKtone = toneVectors[2]
CHtone = toneVectors[3]
RStone = toneVectors[4]
SFtone = toneVectors[5]

print(NZtone)
```

0.2.4 Calculate Similarity Between Vectors

While no similarities have yet been calculated (due to an incomplete implementation of the getTone function), the function to do this similarity calculation has been written.

```
def cosineSimilarity(u, v):
    '''
    Finds the similarities between two vectors
    Returns a float representing the similarity
    '''
    similarity = (u.dot(v)) / ((np.linalg.norm(u))*np.linalg.norm(v))
    return similarity
```

0.3 Next Steps

The next tasks to be completed, in order, are as follows:

0.3.1 Complete getTone Function

As the most important function to answer the research question, the main task will be to get this working so that it returns the appropriate data. Currently it can get the average tone, but it calculates the average tone of ALL the CSV files, not just for each individual CSV. This means it cannot be put into the vector to find the similarity. Fixing this issue is well-underway and will be finished, likely by the end of the week (by Friday 20 May).

0.3.2 Analyse All Data

Once the getTone function has been completed, all 37 months of data will need to be pulled in. Currently, testing has been done with only 2 or 3 files per country. Until the functions have been completed, it saves a lot of time when running the code to only pull a few files at a time.

0.3.3 Analyse Similarities

Once the tone vectors have been acquired, the similarities between New Zealand's tone progression and that of other countries will have to be found using the cosineSimilarity function. Once the vectors have been found, however, this will be very quick.

0.3.4 Present Results

Using Numpy and Pandas, the sentiment progression will be plotted and analysed. This analysis will complete the research question. This also should not take very long, though, and the figures this produces will be used in the final report.

0.4 Difficulties

While the plan is straight-forward, the challenge is implementing all the map-reduce algorithms to acquire all the data. The first main problem I have encountered is that pySpark throws errors after the first year of CSV daily files. I have yet to determine the cause of this, but I believe I will need to change my approach so that instead of bringing in the data day-by-day, I will pull in the data month-by-month. This should result in only pulling in 37 CSV files per country, which will solve this issue, if it is indeed a problem with the number of files I am accessing from GDELT's servers.

Another difficulty is determining how to get the tone of each CSV individually, rather than the average tone of all CSVs per country. I believe this issue is to do with the way I am manipulating the RDDs, and will be a simple fix once the issue is identified.

References

So far, no code has been copied or referenced apart from the Sample GDELT code provided by Professor James Atlas.

Atlas J.(2022). *Sample GDELT Project*, Accessed: 11 May 2022
Available: https://colab.research.google.com/drive/1sTsl_-f2ipgzqM6htsVdKjf4MZ3Ds2CW

Atlas J.(2022). *Sample GDELT Starter Only*, Accessed: 13 May 2022
Available: https://colab.research.google.com/drive/1hXAeG6yheFUQiHfc9Z5ISfNBqAQw47Dq