DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

CSCI 5408 – Data Management, Warehousing, and Analytics

Assignment 4

Anqi Chen

B00838586

Instructor: Dr. Saurabh Dey

Date of submission: December 4, 2020

Business Intelligence Reporting using Cognos

1. Download the weather dataset on Kaggle [1]

2. & 3. Explore the dataset

Identify dimensions on selected data fields

On the website, we can easily get an idea about the general information about this dataset – it contains 17 climate parameters (continuous hourly values) from 122 weather stations in Southeast Brazil. Start with this, we classify that the data fields can be measured by 3 dimensions.

- Location dimension (8 fields)

  wsid, wsnm, elvt, lat, lon, inme, city, prov

- Time Dimension (6 fields)

  mdct, date, yr, mo, da, hr

- Weather Dimension (17 fields)

  Prcp, stp, smax, smin, gbrd, temp, dewp, tmax, dmax, tmin, dmin, hmdy, hmax, hmin, wdsp, wdct, gust

Select the measurable fields

- Location dimension: State (Province), City, Station
- Time Dimension: Year, Month, Date, Hour
- Weather Dimension: Each **factor** can be divided into 3 ways: instant, max, min (Temperature, Humidity, Dew Point, Pressure, Wind, Solar Radiation, Precipitation)

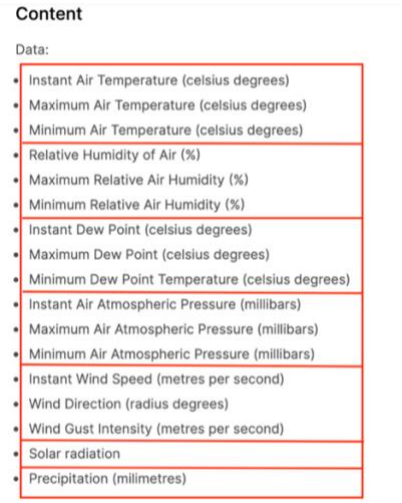


Figure 1 Weather Parameters Classification

Possible scenarios

- On Time Dimension:

Hourly changes in a day → When is the hottest, wettest, windiest time in a day?

Monthly changes in a year → When is the hottest, wettest, windiest time during a year?

Yearly changes of one parameter at the same time of each year → Trends of changes? Like global warming?

- On Location Dimension:

Where are these stations located?

What are the differences of same time data observed between these stations?

- On Weather Parameter Dimension:

What is the possible relationship between dew point and wind, or between temperature and atmospheric pressure?

These are some possible dimensions that a meteorologist may be interested in. The real concern will depend on the actual business domain.

4. Clean and format the dataset

- df.isnull().sum()/df.info()

```
prcp    8371184
stp           0
smax          0
smin          0
gbrd    4108820
temp         31
dewp        475
tmax         26
dmax        310
tmin         34
dmin        807
hmdy          0
hmax         12
hmin         44
wdsp     925561
wdct          0
gust     316474
dtype: int64
```

Figure 2 Before cleaning

```
Data columns (total 15 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   wsnm    884920 non-null  object
 1   lat     884920 non-null  float64
 2   lon     884920 non-null  float64
 3   city    884920 non-null  object
 4   prov    884920 non-null  object
 5   mdct    884920 non-null  object
 6   yr      884920 non-null  int64
 7   mo      884920 non-null  int64
 8   da      884920 non-null  int64
 9   hr      884920 non-null  float64
 10  stp     884920 non-null  float64
 11  temp    884920 non-null  float64
 12  dewp    884920 non-null  float64
 13  hmdy    884920 non-null  float64
 14  wdct    884920 non-null  float64
dtypes: float64(8), int64(3), object(4)
memory usage: 108.0+ MB
```

Figure 3 After cleaning

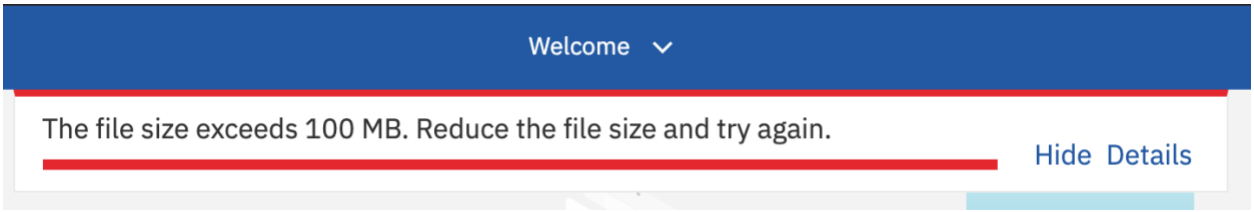


Figure 4 Error info: file size exceeds

Since it is too slow to clean such a huge dataset with more than 9700000 records, also Cognos cannot deal with file exceeds 100 MB, so I decided to drop some rows and columns to make the file size less than 100MB.

The script doing all the simple clean-up can be found on GitLab – format.py.

https://git.cs.dal.ca/anqi/csci5408-a1-oceantracking/-/tree/master/A4

5. Create/import the dimension tables

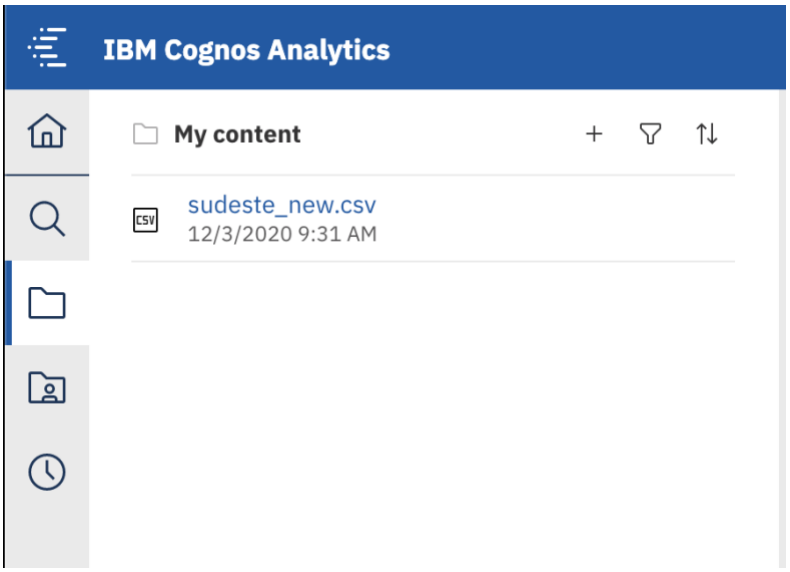


Figure 5 The data file is imported

6. Justification of your model creation

As mentioned before, to create the data model, I want to have information on both Time and Location for the core weather data. After normalizing the dimensional tables, creating other csv, the star schema will be built like the screenshot shown below (Fig 6).

7. Screenshots of the model

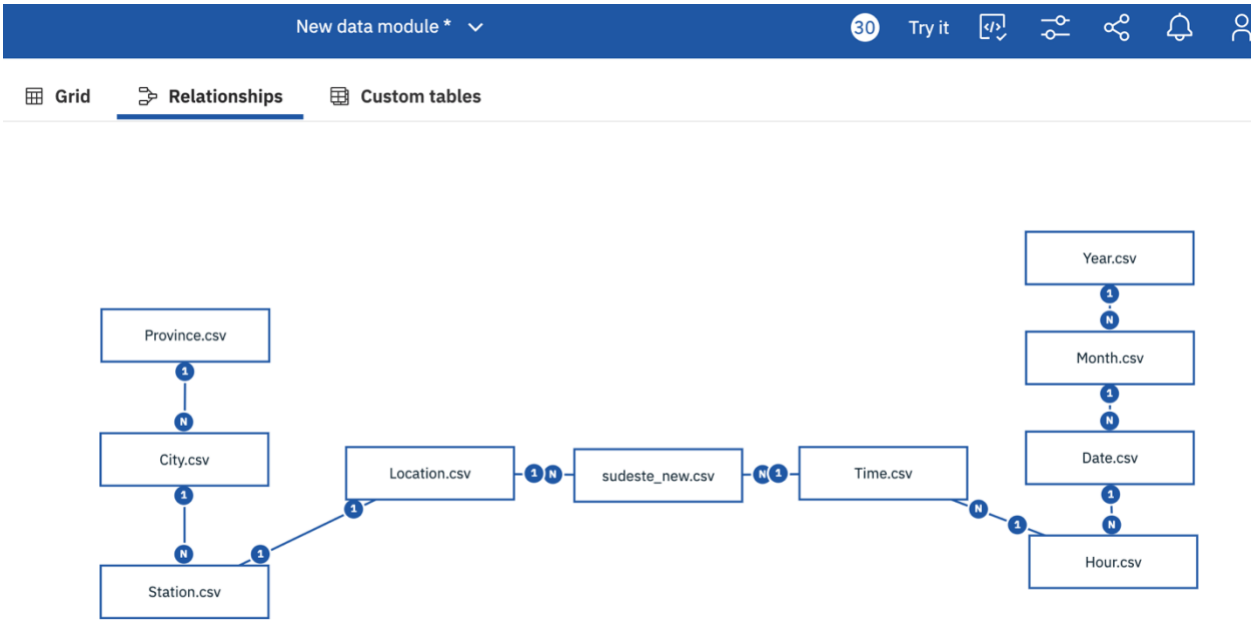


Figure 6 Normalized Snowflake Schema

8. Screenshot of the analysis

Note: details of this part can also be seen in another file named “dashboard.pdf”.

Figure 7 Dashboard Tab1 (Weather changes on Location, Time)

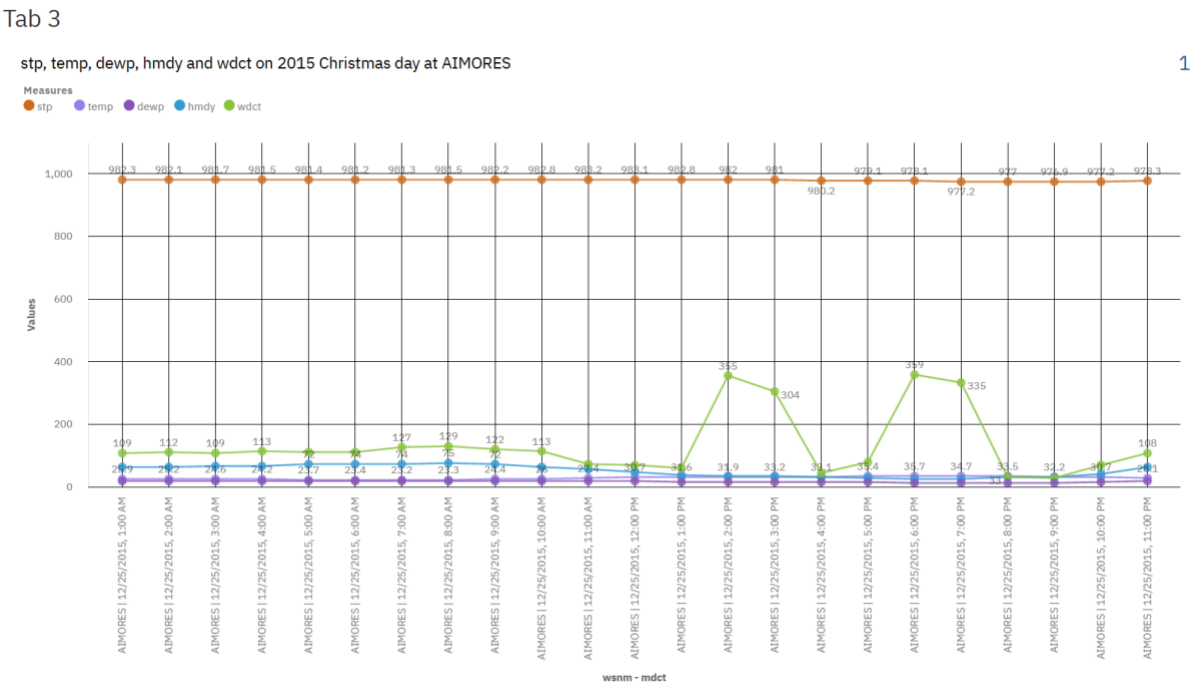


Figure 8 Dashboard Tab3 (Hourly weather changes on a specific day)

Basic Data Analysis and Presentation using R

1. Convert the value of tag “created_at” from String into Timestamps (numeric)

2. Data Preparation (rescale variables for comparability)

3. Determine the optimal number of clusters for k-means clustering

As the figure shown below (Fig 9), it seems 2 is the elbow point in the curve.

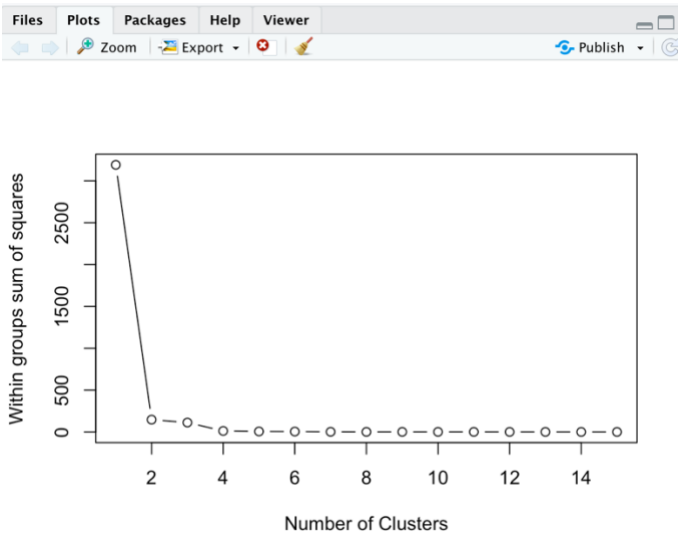
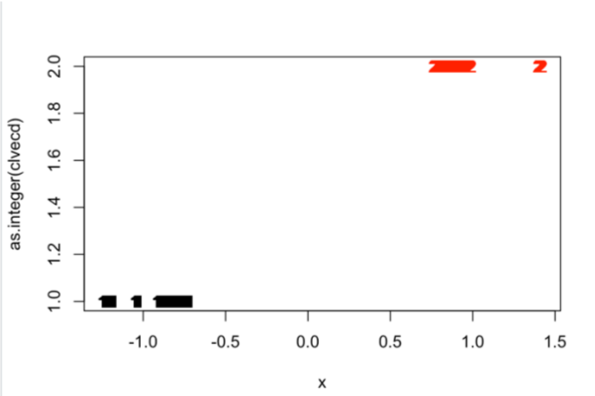
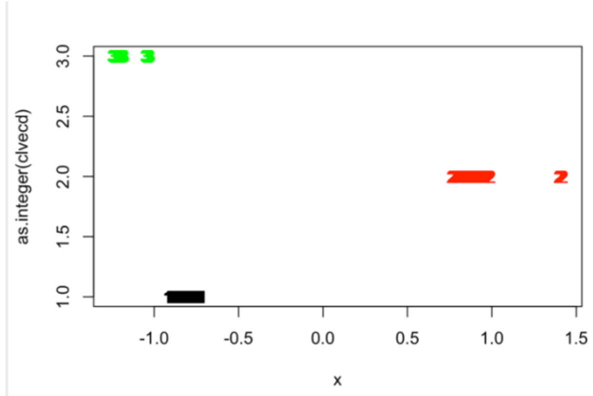
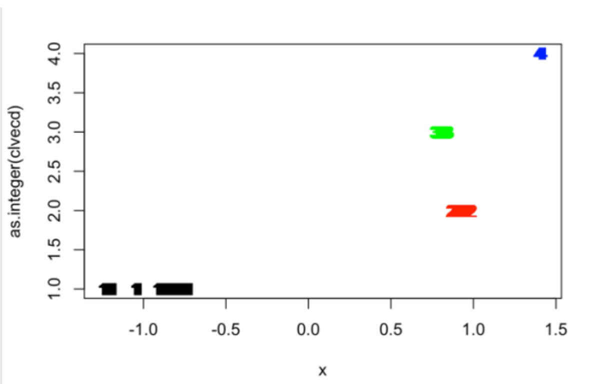


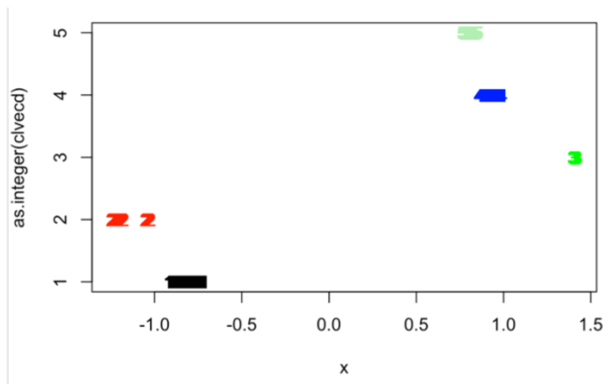
Figure 9 Finding optimal number of clusters

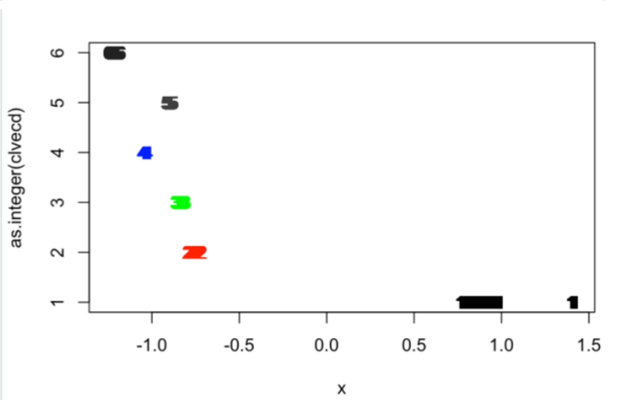
4. Plotting cluster images (K=2,3,4,5,6,7,8)

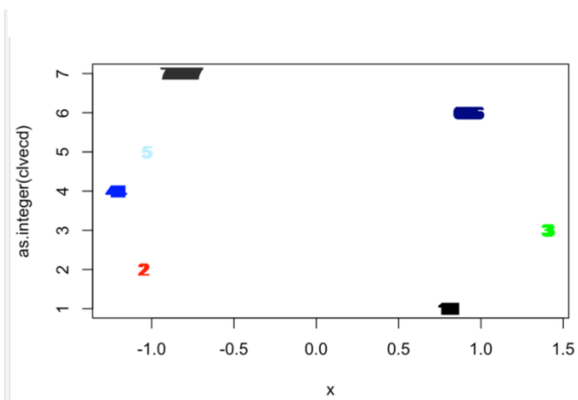


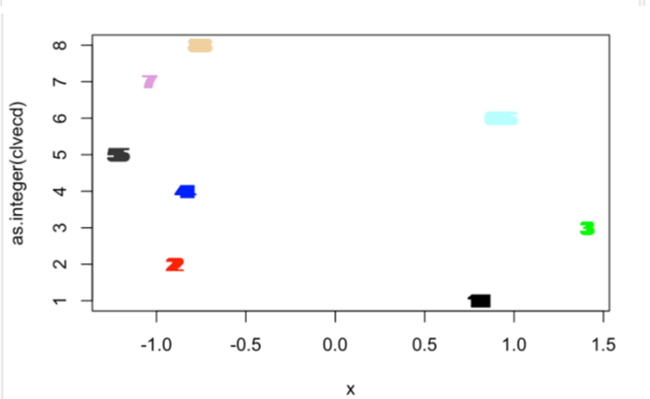












The script can be found on GitLab and in zipped file (KMeans.R)

To validate the KMeans clustering, I compute cluster validation statistics as below.

cluster.number	cluster.size	cluster.ceters	separation
2	1596 1596	1 -0.9764937 2 0.9764937	1.468462 1.468462
3	1197 399 1596	1 -0.8294241 2 0.9764937 3 -1.1235634	0.3965278 1.468462 2.103508
4	399 798 1197 798	1 -0.9764937 2 0.9166095 3 0.8014500 4 1.4107097	0.39652780 0.09682406 0.39652780 0.09682406
5	399 399 399 1197 798	1 -0.8294241 2 -1.1235634 3 1.4107097 4 0.9166095 5 0.8014500	0.39652780 0.11636062 0.09682406 0.39652780 0.09682406
6	399 599 399 199 399 1197	1 0.9764937 2 -0.7555912 3 -0.8448379 4 -1.0374250 5 -0.8972834 6 -1.2097018	0.116360620 0.004723125 0.396527795 0.004723125 0.096824058 0.396527795
7	199 318 170 798 599 709 399	1 0.8014500 2 -1.0475964 3 1.4107097 4 -1.2097018 5 -1.0272025 6 0.9166095 7 -0.8294241	0.004723125 0.000429375 0.002576250 0.096824058 0.004723125 0.000429375 0.396527795

8	315 84 882 399 199 515 399 399	1 0.8014500 2 -0.8972834 3 1.4107097 4 -0.8448379 5 -1.2097018 6 0.9166095 7 -1.0374250 8 -0.7555912	0.0002146875 0.0004293750 0.0002146875 0.3965277952 0.0047231248 0.0004293750 0.1163606199 0.0968240583

Sentiment Analysis

1. Bag-of-words for Each Tweet

```
    print (dict)
```

```
{'rt': 1, 'frank': 1, 'chimienti': 1, 'handfuls': 1, 'biden': 1, 'rally': 1,
'pa': 1, 'cars': 2, 'honking': 1, 'least': 1, 'trump': 1, 'voters': 1}
{'usually': 1, 'wicker': 1, 'furniture': 1, 'used': 1, 'outdoors': 1, 'porches':
1, 'decks': 1, 'using': 1, 'dining': 1, 'room': 1, 'table': 1}
{'underthewitness': 1, 'hi': 1, 'city': 1, 'law': 2, 'amp': 1, 'requires': 1,
'everyone': 1, 'wear': 1, 'mask': 1, 'face': 1, 'c': 1}
{'dawnroseturner': 1, 'bonjour': 1, 'unroll': 1, 'smillssk': 1, 'sk': 1,
'covid': 1, 'update': 1, 'new': 1, 'restrictions': 1, 'expansion': 1, 'mandat':
1}
{'rt': 1, 'fox': 1, 'dallas': 1, 'county': 1, 'issuing': 1, 'warning': 1,
'increase': 1, 'covid': 1, 'cases': 1, 'like': 1, 'never': 1, 'seen': 1,
'judge': 1, 'clay': 1, 'jenkins': 1, 'urged': 1, 'peop': 1}
{'rt': 1, 'fox': 1, 'dallas': 1, 'county': 1, 'issuing': 1, 'warning': 1,
'increase': 1, 'covid': 1, 'cases': 1, 'like': 1, 'never': 1, 'seen': 1,
'judge': 1, 'clay': 1, 'jenkins': 1, 'urged': 1, 'peop': 1}
{'spend': 1, 'weekend': 1, 'mulligans': 1, 'take': 1, 'advantage': 1,
'takeaway': 1, 'options': 1, 'heat': 1, 'serve': 1, 'dinner': 1, 'package': 1,
'includi': 1}
{'rt': 1, 'katinalynn': 1, 'please': 3, 'prioritize': 1, 'schools': 2, 'bars':
1, 'restaurants': 1, 'gyms': 1, 'far': 1, 'less': 1, 'transmission': 1}
{'rt': 1, 'fox': 1, 'dallas': 1, 'county': 1, 'issuing': 1, 'warning': 1,
'increase': 1, 'covid': 1, 'cases': 1, 'like': 1, 'never': 1, 'seen': 1,
'judge': 1, 'clay': 1, 'jenkins': 1, 'urged': 1, 'peop': 1}
{'rt': 1, 'fox': 1, 'dallas': 1, 'county': 1, 'issuing': 1, 'warning': 1,
'increase': 1, 'covid': 1, 'cases': 1, 'like': 1, 'never': 1, 'seen': 1,
'judge': 1, 'clay': 1, 'jenkins': 1, 'urged': 1, 'peop': 1}
{'current': 2, 'indoor': 1, 'temperature': 2, 'humidity': 1, 'forestville': 1,
'time': 1}
{'rt': 1, 'drericding': 1, 'breaking': 1, 'entire': 1, 'country': 1,
'coronavirus': 1, 'pandemic': 1, 'another': 1, 'new': 1, 'time': 1, 'record': 1,
```

Figure 10 Sample BOW of tweets

2. Positive.txt & Negative.txt

Online sources used [4].

3. Polarity Tags

A	B	C	D
Tweet	Text	Match	Polarity
1	['rt', 'frank', 'chimienti', 'handfuls', 'biden', 'rally', 'pa	trump	positive
2	['usually', 'wicker', 'furniture', 'used', 'outdoors', 'porches', 'decks', 'using', 'dining', 'room', 'table']		neutral
3	['underthewitness', 'hi', 'city', 'law', 'amp', 'law', 'requires', 'everyone', 'wear', 'mask', 'face', 'c']		neutral
4	['dawnroseturner', 'bonjour', 'unroll', 'smillssk', 'sk', 'covid', 'update', 'new', 'restrictions', 'expansion', 'mandat		neutral
5	['rt', 'fox', 'dallas', 'county', 'issuing', 'warning', 'incre	warning,like	neutral
6	['spend', 'weekend', 'mulligans', 'take', 'advantage', '	advantage	positive
7	['rt', 'katinalynn', 'please', 'please', 'please', 'prioritize', 'schools', 'bars', 'restaurants', 'gyms', 'far', 'less', 'tran		neutral
8	['current', 'indoor', 'temperature', 'humidity', 'temperature', 'forestville', 'current', 'time']		neutral
9	['rt', 'drericding', 'breaking', 'entire', 'country', 'coron	breaking	negative
10	['appletontech', 'journalsentinel', 'protests', 'outside'	protests,problems,selfish	negative
11	['rt', 'proflwiley', 'go', 'indoor', 'gym', 'dine', 'indoors'	risk	negative
12	['rt', 'titialayour', 'tunmiike', 'kastro', 'whatever', 'loc	cry	negative
13	['nygovcuomo', 'agreed', 'spreading', 'widely', 'schoo	risk,worsening	negative
14	['jim', 'jordan', 'differences', 'indoor', 'outdoor', 'wearing', 'mask', 'wearing', 'mask', 'indoor', 'maskless', 'gath		neutral
15	['deitymicrophone', 'bruh', 'get', 'every', 'wanted', 're	right	positive

Figure 11 Sample Polarity Results

More information can be found in the following appendix (after reference) and GitLab Repo.

- bagOfWords.ipynb
- sentimenta_analysis_result.csv

Semantic Analysis

1. TF-IDF on “Canada”, “rain”, “cold”, “hot”

Table 1 Words Occurrence in New Articles

Search Query	Document containing term(df)	Total Documents(N)/ number of documents term appeared (df)	Log10(N/df)
Canada	25	40.0	1.6020599913279623
rain	85	11.764705882352942	1.0705810742857074
cold	1	1000.0	3.0
hot	7	142.85714285714286	2.154901959985743

2. Frequency Count of “Canada” per document

A	B	C	D	E
Article	Total Words(	Frequency(f)	Relative Frequency (f/m)	
article111	105	2	0.066666667	
article213	194	1	0.036082474	
article261	58	1	0.120689655	
article279	254	3	0.027559055	
article320	39	2	0.179487179	
article357	448	1	0.015625	
article412	190	3	0.036842105	
article479	84	2	0.083333333	
article542	94	1	0.074468085	
article565	45	1	0.155555556	
article609	274	2	0.025547445	
article674	223	1	0.031390135	
article700	89	1	0.078651685	
article674	223	1	0.031390135	
article733	147	2	0.047619048	
article739	243	3	0.028806584	
article756	75	2	0.093333333	
article772	82	1	0.085365854	
article778	29	1	0.24137931	
article858	505	1	0.013861386	
article863	271	4	0.025830258	
article890	775	1	0.009032258	

Figure 12 Sample Result Table

3. News article with the highest relative frequency

It is article991 in reuster14.json, and f/m is 0.

More information can be found in the following appendix (after reference) and GitLab Repo.

- TF-IDF.ipynb
- semantic_analysis_results.csv

Reference

[1] “Hourly Weather Surface - Brazil (Southeast region).” https://kaggle.com/PROPPG-PPG/hourly-weather-surface-brazil-southeast-region.

[2] “pandas.DataFrame — pandas 1.1.4 documentation.” https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html.

[3] “Quick-R: Cluster Analysis.” https://www.statmethods.net/advstats/cluster.html.

[4] Bing Liu, Minqing Hu and Junsheng Cheng. "Opinion Observer: Analyzing and Comparing Opinions on the Web." Proceedings of the 14th International World Wide Web conference (WWW-2005), May 10-14, 2005, Chiba, Japan.

[4] “An introduction to Bag of Words and how to code it in Python for NLP,” *freeCodeCamp.org*, Dec. 18, 2018. https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04.

[5] “(Tutorial) Simplifying Sentiment Analysis in Python,” *DataCamp Community*, Jan. 07, 2020. https://www.datacamp.com/community/tutorials/simplifying-sentiment-analysis-python.

[6] “Tutorial: Extracting Keywords with TF-IDF and Python’s Scikit-Learn.” https://kavita-ganesan.com/extracting-keywords-from-text-tfidf/#.X8vzARP0m8p.

bagOfWords

December 5, 2020

[1]:
```
import glob
import os
import pandas as pd
```

[2]:
```
inputfile = str(os.path.dirname(os.getcwd())) + "/A4/tweets/*.csv"
outputfile = str(os.path.dirname(os.getcwd())) + "/A4/tweets/all.csv"
csv_list = glob.glob(inputfile)
```

[3]:
```
filepath = csv_list[0]
df = pd.read_csv(filepath, encoding="gbk", low_memory=False)
df = df.text
df = df.to_csv(outputfile, encoding="gbk", index=False)
```

[4]:
```
# Combine csv files and only select the text field, save into all.csv
for i in range(1, len(csv_list)):
    filepath = csv_list[i]
    df = pd.read_csv(filepath, encoding="gbk", low_memory=False)
    df = df.text
    df = df.to_csv(outputfile, encoding="gbk", index=False, header=False,␣
 ↪mode='a+')
```

[5]:
```
# Change Dtype to String, Drop duplicated rows
import pandas as pd
import re

data = pd.read_csv("tweets/all.csv")
data['text'] = data['text'].astype('string')
data.drop_duplicates(inplace=True)
```

[6]:
```
# Download stopwords from NLTK
from nltk import download
download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/chenanqi/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[7]:
```
# Clean Data
def text_cleaner(text):
    new_string = text.lower() #lower case
    new_string = re.sub(r'\([^)]*\)', '', new_string)
    new_string = re.sub('"','', new_string)
    new_string = re.sub(r"'s\b","",new_string) # delete 's in the text
    new_string = re.sub("[^a-zA-Z]", " ", new_string) # Change punctuation into␣
 ↪a single space
    new_string = re.sub('[m]{2,}', 'mm', new_string)
    cleaned_text = [w for w in new_string.split() if w not in stop_words]
    return cleaned_text
```

[8]:
```
# Collect cleaned texts, save them into another column
words = []
for t in data['text']:
    word = text_cleaner(t)
    words.append(word)
```

[9]:
```
data['cleaned_text'] = words
# data['cleaned_text'] = data['cleaned_text'].astype('string')
data['cleaned_text'][:5]
```

[9]:
```
0    [rt, frank, chimienti, handfuls, biden, rally,…
1    [usually, wicker, furniture, used, outdoors, p…
2    [underthewitness, hi, city, law, amp, law, req…
3    [dawnroseturner, bonjour, unroll, smillssk, sk…
4    [rt, fox, dallas, county, issuing, warning, in…
Name: cleaned_text, dtype: object
```

[10]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2185 entries, 0 to 4787
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   text          2185 non-null   string
 1   cleaned_text  2185 non-null   object
dtypes: object(1), string(1)
memory usage: 51.2+ KB
```

[11]:
```
for r in data['cleaned_text']:
    dict = {}
    for w in r:
        dict[w] = dict.get(w, 0) + 1
#     print (dict)
```

[12]:
```
pWords = open("polarity/positive-words.txt", "r")
pWordsRead = pWords.read()
pWList = pWordsRead.split("\n")
nWords = open("polarity/negative-words.txt", "r")
nWordsRead = nWords.read()
nWList = nWordsRead.split("\n")
```

[13]:
```
def polarity_check(row):
    dict = {}
    for w in row:
        dict[w] = dict.get(w, 0) + 1
    positive = 0
    negative = 0
    match = []
    for key in dict:
        if key in pWList:
            # count positive words
            positive += dict[key]
            match.append(key)

        elif key in nWList:
            # count negative words
            negative += dict[key]
            match.append(key)

    if negative < positive:
        polarity = "positive"
    elif negative == positive:
        polarity = "neutral"
    else:
        polarity = "negative"

    return match, polarity
```

[14]:
```
import csv
result_csv = open("sentiment_analysis_results.csv", "w+", newline='',␣
 ↪encoding="utf-8")
result_csv = csv.writer(result_csv, delimiter=',', quotechar='"', quoting = csv.
 ↪QUOTE_MINIMAL)
result_csv.writerow(['Tweet', 'Text', 'Match', 'Polarity'])
cnt = 0
for row_text in data['cleaned_text']:
    cnt += 1
    match, polarity = polarity_check(row_text)
    result_csv.writerow([cnt, row_text, ','.join(match), polarity])
```

[ ]:

TF-IDF

December 5, 2020

[1]:
```
import pandas as pd

df=pd.read_json("data/reuster.json")
print("Schema:\n",df.dtypes)
```

```
Schema:
 _id        object
article    object
reuters    object
text       object
title      object
dtype: object
```

[2]:
```
import re
def pre_process(text):
    text=text.lower() # lowercase
    text=re.sub("","",text) #remove tags
    text=re.sub("(\\d|\\W)+"," ",text) # remove special characters and digits
    return text

df['text'] = df['text'].apply(lambda x:pre_process(x))

df['text'][:1]
```

[2]:
```
0     title johannesburg gold shares close mixed to…
Name: text, dtype: object
```

[3]:
```
import math
def wordCounter(word):
    count = 0
    for t in df['text']:
        if t.count(word) > 0:
            count += 1
    return count
words = ['canada', 'rain', 'cold','hot']
N=1000
for w in words:
    count = wordCounter(w)
```

```
    print(w)
    print("df: " + str(count))
    if count != 0:
        tmp = N / count
        print("N/df: " + str(tmp))
        print("Log10(N/df):" + str(math.log10(tmp))+"\n")
```

```
canada
df: 25
N/df: 40.0
Log10(N/df):1.6020599913279623

rain
df: 85
N/df: 11.764705882352942
Log10(N/df):1.0705810742857074

cold
df: 1
N/df: 1000.0
Log10(N/df):3.0

hot
df: 7
N/df: 142.85714285714286
Log10(N/df):2.154901959985743
```

[4]:
```
resultList=[]
maxRF = 0
word = "canada"

for t in df['text']:
    cnt = 0
    total = t.split()
    if t.count(word) > 0:
        cnt += len(re.findall(word, t))
        relativeFreq = count /len(total)
        articleName=df.loc[df['text']==t, 'article'].values[0]
        res = [str(articleName), str(len(total)), str(cnt), str(relativeFreq)]
        resultList.append(res)
        if maxRF < relativeFreq:
            maxRF = relativeFreq
            target =articleName
resultList
```

[4]:
```
[['article111', '105', '2', '0.06666666666666667'],
 ['article213', '194', '1', '0.03608247422680412'],
 ['article261', '58', '1', '0.1206896551724138'],
 ['article279', '254', '3', '0.027559055118110236'],
 ['article320', '39', '2', '0.1794871794871795'],
 ['article357', '448', '1', '0.015625'],
 ['article412', '190', '3', '0.03684210526315789'],
 ['article479', '84', '2', '0.08333333333333333'],
 ['article542', '94', '1', '0.07446808510638298'],
 ['article565', '45', '1', '0.15555555555555556'],
 ['article609', '274', '2', '0.025547445255474453'],
 ['article674', '223', '1', '0.03139013452914798'],
 ['article700', '89', '1', '0.07865168539325842'],
 ['article674', '223', '1', '0.03139013452914798'],
 ['article733', '147', '2', '0.047619047619047616'],
 ['article739', '243', '3', '0.02880658436213992'],
 ['article756', '75', '2', '0.09333333333333334'],
 ['article772', '82', '1', '0.08536585365853659'],
 ['article778', '29', '1', '0.2413793103448276'],
 ['article858', '505', '1', '0.013861386138613862'],
 ['article863', '271', '4', '0.025830258302583026'],
 ['article890', '775', '1', '0.00903225806451613'],
 ['article927', '98', '1', '0.07142857142857142'],
 ['article987', '372', '1', '0.01881720430107527'],
 ['article991', '14', '1', '0.5']]
```

[5]:
```
target
```

[5]:
```
'article991'
```

[6]:
```
maxRF
```

[6]:
```
0.5
```

[7]:
```
import csv
# write csv file
semantic_csv = open("semantic_analysis_results.csv", "w+", newline='',␣
 ↪encoding="utf-8")
semantic_csv = csv.writer(semantic_csv, delimiter=',', quotechar='"',␣
 ↪quoting=csv.QUOTE_MINIMAL)
semantic_csv.writerow(['Article', 'Total Words(m)', 'Frequency(f)', 'Relative␣
 ↪Frequency (f/m)'])
for result in resultList:
    semantic_csv.writerow([result[0], result[1], result[2], result[3]])
```

[ ]:
```

```