Project 4(Spark) Report

Problem 1:
Find the day of the week does @PrezOno tweet the most on average?

Data: Twitter
The data contains minute by minute tweets in JSON format.

Tools: Pyspark

Analysis:

- First all the tweets are read into an RDD.
- Among these tweets, all the tweets by PrezOno (screen_name) are filtered and stored in another RDD.
- Now from PrezOno's tweets, created time is extracted using a "map" action and day of week is taken from this time stamp.
- The day of week along with a count "1" are sent as key/value pairs.
- Later these key/value pairs are "reduced" based on the key and adding up all the "1's" thus giving the total counts of all tweets on each particular day.
- Later this RDD is sorted based on the counts to get the top 3 days of week where PrezOno tweets most.

Output:
Total tweets by PrezOno -> 341
Sunday -> 59 tweets
Wednesday-> 55 tweets
Thursday-> 54 tweets

The code for this question can be found at :
https://github.uc.edu/ghandisv/P4Spark/blob/master/DayOfWeek.py

Problem 2:
How does @PrezOno’s tweet length compare to the average of all others? What is his average length? All others?

Data: Twitter
The data contains minute by minute tweets in JSON format.

Tools: Pyspark

Analysis:

- First all the tweets are read into an RDD.
- From all the tweets, tweet text is extracted by using “map” action on the earlier RDD. The new RDD now contains all tweet texts.
- Then from the first RDD again, only PrezOno’s tweets are filtered and tweet texts from his tweets are also extracted using map function.
- The length of tweets texts of both the RDDs (PrezOno’s and everyone else) are found using a map action on each and every tweet text which finds the length of the text
- The “stats()” method is then used to find the basic statistics like average, count etc.

Output:
Everyone’s Tweets -> count: 6080302; average length: 79.84
PrezOno’s Tweets -> count: 341; average length: 103.88

The code for this question can be found at :
https://github.uc.edu/ghandisv/P4Spark/blob/master/TweetLength.py

Problem 3:

For those tweets with location information, what lat/long (or city/state) is the centroid? What was the proportion of tweets with location to those without?

Data: Twitter

The data contains minute by minute tweets in JSON format.

Tools: Pyspark

Analysis:

- First all the tweets are read into an RDD.
- Then the total tweet count is found by applying the "count()" function on this RDD.
- From the RDD of all the tweets, the tweets having geolocation are filtered.
- The tweet count of tweets having geolocation is also found again by applying the "count()" function.
- Then from the tweets containing geolocation, the coordinates(latitude and longitude) are extracted using the "map" action.
- These coordinates are then split into two lists latitudes and longitudes to make it easy to find the centroid.
- The centroid value is found for each of the list and they are again combined to find the final centroid coordinate.

Output:

Total Tweet Count: #6080302

Tweets with geolocation: #1868302

Proportion of tweets with geolocation: 30.72 %

Centroid(latitude,longitude): (38.99,-82.45)

Note: Based on these coordinates, the name of the place was ***Vinton Rd, Thurman, OH 45685, USA***

The code for this question can be found at : https://github.uc.edu/ghandisv/P4Spark/blob/master/geoTweets.py

Problem 4:
What twitter user tweeted the most? What is the top 5 longest tweeters over each's average tweet length? Bottom 5?

Data: Twitter
The data contains minute by minute tweets in JSON format.

Tools: Pyspark

Analysis:

- First all the tweets are read into an RDD.
- From this RDD, user's screen names are extracted using "map" actions.
- Along with screen name, count "1" is emitted as a key value pair.
- These key value pairs are reduced based on key by adding up all these counts which indicates the total number of tweets from each individual user.
- These "reduced" counts are then sorted in descending order and then the first user is taken based on the count values(highest tweeter).
- Then from the RDD containing all the tweets, screen_name and tweet text lengths are extracted again.
- This values are emitted as key/value pairs using "map()" function.
- These key-value pairs are reduced on key by finding the average tweet length of each individual user(key).
- These averages are now sorted in both ascending and descending orders.
- From the sorted lists top 10 and bottom 10 tweeters are extracted based on average tweet length.

The code for this question can be found at :
https://github.uc.edu/ghandisv/P4Spark/blob/master/TopTweeters.py

Output:

```
User with highest tweets-> [(screen_name:‘marilyn9743’, tweets:3419)]
-------------------Top 10------------------
Screen Name                       Avg. Tweet Length
‘Niehime1210kayo’                      140
'Stevetheantler'                       140
'Jgelfuso'                             140
'Nathalietaylor7'                      140
'ItsBarrett'                           140
'FLATeuzim'                            140
'Brerochaa'                            140
'TombaHuddlestan'                      140
'ADailey15'                            140
'suzieperlsteinR'                      140
-------------------Bottom 10------------------
Screen Name                       Avg. Tweet Length
'Vanessare24'                          3
'angellicaa_rose'                      8
'NeverHate14'                          8
'Shee_Finessin'                        9
'laceylouise97'                        9
'anallii_'                             10
'Kwaller_cru11'                        12
'King_RuPERT_'                         12
'Raggdolljim'                          13
'Alikeerrs'                            15
```

Problem 5: (Google onegram)

Find the top 25 words and plot their relative frequency in a heatmap for 10 year increments. The y-axis should contain words, and the x-axis 10 year increments, where the box's color represents the popularity of that word to all others during that 10-year block. Only plot back to 1800.

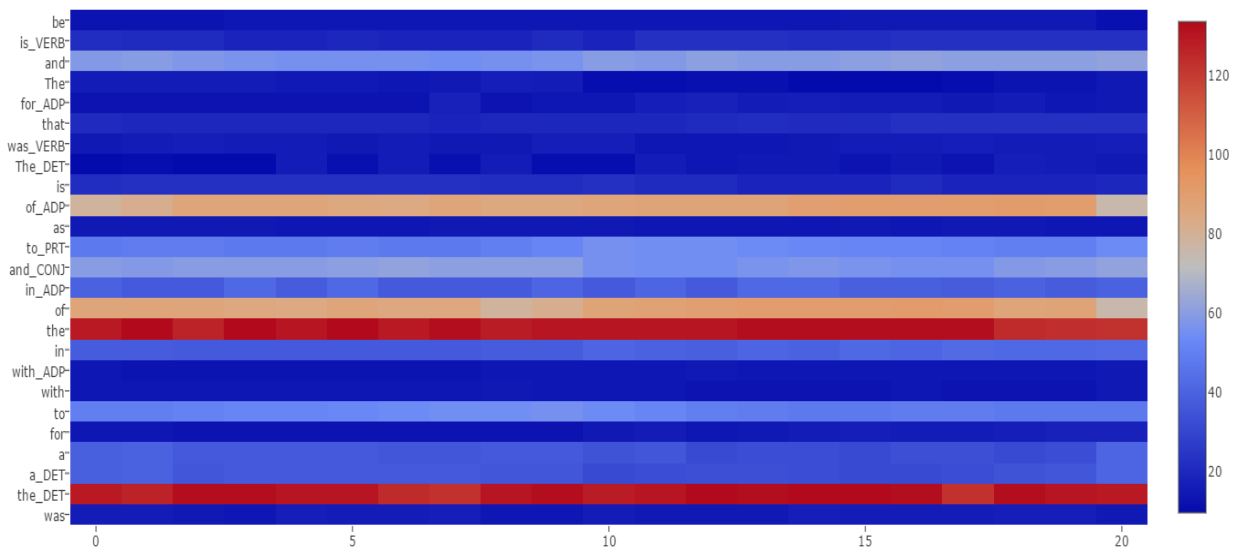
Dataset: Google 1gram data.
The data is organized as 4 string tuples on each line delimited by a TAB character. The order is word, year, frequency for that year and number of books it appeared in that year.

Tools: Python for Spark, plotly library for plotting the heatmap

Analysis:

- First the data is brought into an RDD and split into strings.
- Then we sum up the frequencies (3rd string) for each word and find the top 25 most frequent words and put them in a list.
- We then run through the data once more and find the individual frequencies per year, for the top 25.
- We bin these into bins of 10 years each.
- Now, we find relative frequencies per bin, for these 25 words. This data is used to plot the heatmap.

The code for this question can be found at :
For getting data:
https://github.uc.edu/ghandisv/P4Spark/blob/master/oneGram/oneGram.py
For plotting:
https://github.uc.edu/ghandisv/P4Spark/blob/master/oneGram/plot.py


be
is_VERB
and
The
for_ADP
that
was_VERB
The_DET
is
of_ADP
as
to_PRT
and_CONJ
in_ADP
of
the
in
with_ADP
with
to
for
a
a_DET
the_DET
was
0
5
10
15
20
120
100
80
60
40
20

Heatmap