**P4/P5 Project**

**CLOUD COMPUTING**

**Group Members:**

Prudhvi Rao Shedimbi

Dwipika Konduru

Pallavi Muthyala

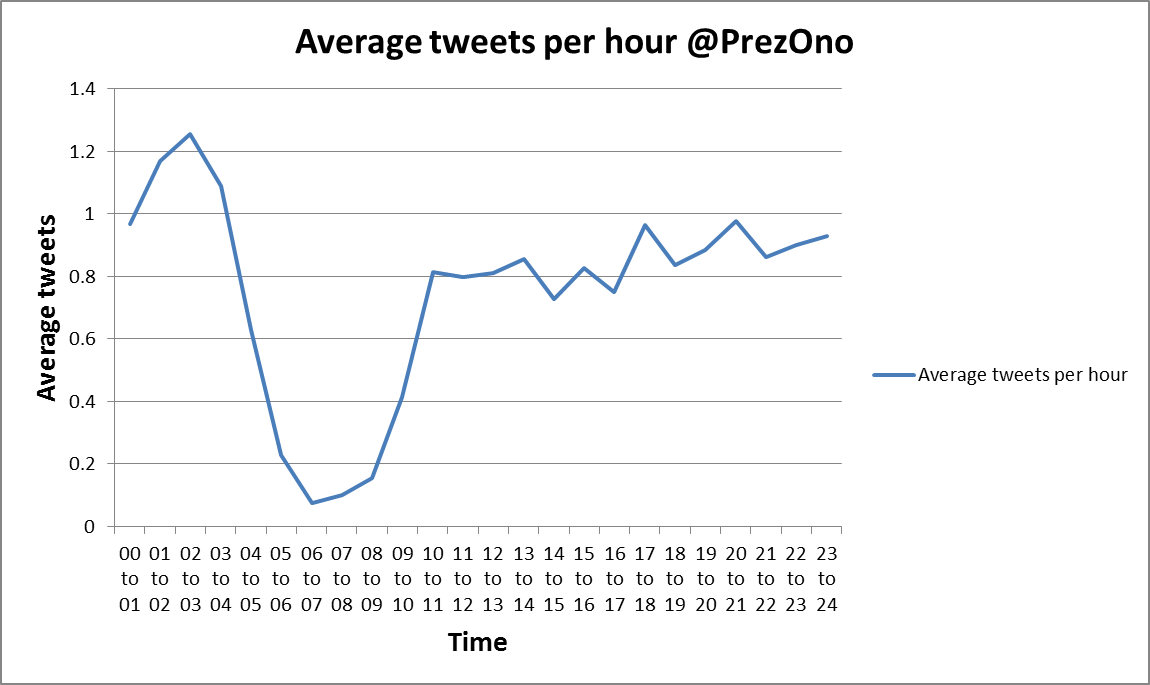
(Twitter – Question 1) The first question that we chose was to find out the hour of the day when @PrezOno tweets the most on average. We solved the problem by using Hadoop in streaming mode. To find out average number of tweets per hour, we need to find out the count of tweets per hour and the number of days for which we are analyzing the data. The data is in Json format, so we used Json package in python to load data. The map function (map1.py), returns a list of 24 integers, where each integer is the count of tweets by @PrezOno at a particular hour of the day and the position (index) of integer in the list is the hour of the day. For example, if the fourth element in the list is 24, it means that @PrezOno tweeted 24 times between 04:00 and 05:00 hours according to the data allocated to that particular mapper. The reduce (reduce1.py) function takes all the lists from all the mappers and aggregates them into one list to produce a final list containing total number of tweets by @PrezOno at each hour of the day. We are analyzing data for 394 days. So to find the average tweets per hour, we divide the total tweets per hour of the day by 394.

**Following is the solution obtained:**

[0.96700507614213194, 1.1675126903553299, 1.2563451776649746, 1.0888324873096447, 0.62690355329949243, 0.22842639593908629, 0.073604060913705582, 0.098984771573604066, 0.1548223350253807, 0.4137055837563452, 0.81472081218274117, 0.79695431472081213, 0.80964467005076146, 0.85532994923857864, 0.72842639593908631, 0.82487309644670048, 0.75126903553299496, 0.96446700507614214, 0.8350253807106599, 0.88324873096446699, 0.97715736040609136, 0.86040609137055835, 0.90101522842639592, 0.92893401015228427]

From the above result, we can see that @PrezOno tweets the most between 02:00 - 03:00 (AM) hours of the day, with an average of 1.256 tweets per day during that time period.

Here is the graph for the expected number of tweets for each hour of the day:



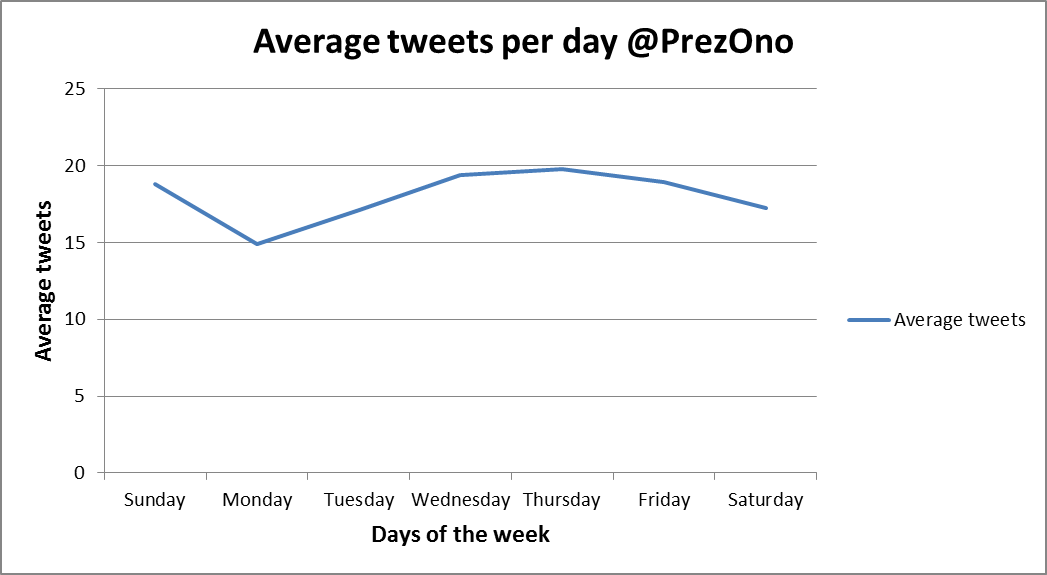
(Twitter – Question 2) The second question that we chose was to find the day of the week when @PrezOno tweet the most on average. We solved this problem by using Hadoop in streaming mode. To find out the average number of tweets per day, we need to find out the count of tweets per day and the number of times each day occurred in the dataset. The data is in Json format, so we used Json package in python to load data. The map function (map2.py), returns a list of 7 integers, where each integer is the count of tweets by @PrezOno on a particular day of the week and the position (index) of integer in the list is the day of the week starting from Sunday, i.e. 0 is Sunday, 1 is Monday, 2 is Tuesday and so on. For example, if the fourth element in the list is 214, it means that @PrezOno tweeted 214 times on a Thursday according to the data allocated to that particular mapper. The reduce (reduce2.py) function takes all the lists from all the mappers and aggregates them into one list to produce a final list containing total number of tweets by @PrezOno on each day of the week. We are analyzing data for 394 days and there are 7 days a week, so to find the average tweets per day, we divide the total tweets per hour of the day by 394/7, i.e. 56.286.

**Following is the solution obtained:**

[18.814625306470525, 14.888249298226912, 17.091283800589846, 19.36538393206126, 19.774011299435028, 18.903457342856129, 17.215648651529687]

From the above result, we can see that @PrezOno tweets the most on Thursday, with an average of 19.774011299435028 tweets.

Here is the graph for the number of tweets for each day of the week:



(Twitter – Question 3) The third question that we chose was to compare the tweet length of @PrezOno with other Twitter users. We solved this problem by using Hadoop in streaming mode. To find out the average tweet length of @PrezOno and other Twitter users, we need to find out the total length of all tweets and the count of tweets by @PrezOno, and the total length of all tweets and the count of tweets by other Twitter users. The data is in Json format, so we used Json package in python to load data. The map function (map3.py), returns the required data in the following format:

Key: Ono Value: [total tweet length by @PrezOno, number of tweets by @PrezOno]

Key: Other Value: [total tweet length by all other Twiiter users, number of tweets by all other Twiiter users]

The reduce function (reduce3.py), aggregates the total length of tweets and number of tweets by @PrezOno from all the mappers. It does the same for all other Twitter users. Once we have the total length of tweets and number of tweets for @PrezOno and other Twitter users, we can calculate the average tweet length by dividing length of tweets by number of tweets.

Tricks Used:

In order for map function to return more than one type of values, key used is the string ‘Ono’ and ‘Other’. The value is a list: [total length of tweets, number of tweets]. To read these values in reduce function, we stripped the square braces ‘[’ and ‘]’ and then split the remainder string using ‘,’ as seperator.

**Here is the solution obtained:**

Avg length of PrezOno is 103.322903453

Avg length of Other is 81.694189469

Observation:

We can see that average length of a tweet by @PrezOno is 103.323, which is much higher than the average length of a tweet by other Twitter users whose average tweet length is 81.694.

@PrezOno’s tweet length is about 26.5% longer when compared to tweet length of all other users.

(Twitter – Question 5) The fourth question that we chose was to find the Twitter user who tweeted the most and top 5 longest tweeters along with bottom 5. We solved this problem by using Hadoop in streaming mode. The data is in Json format, so we used Json package in python to load data. The map function (map4.py) returns username as key and a list as value. That list contains count of tweets as first element and maximum length of a tweet by that user as second element. An empty dictionary is initialized in the beginning. If a username does not exist in the dictionary as key, then it is added to the dictionary with value being a list whose first element is 1 and second element is the length of the tweet. If a username already exists in the dictionary, then its value (a list) is updated. The first element of the list is incremented by 1 and the second element is changed if the length of the current tweet is greater than older length.

The reduce function (reduce4.py) finds the total tweet count and maximum tweet length of a user by summing up the values returned by all mappers. The reduce function has 2 dictionaries. The first dictionary has username as key and total number of tweets as value. The second dictionary has username as key and maximum tweet length by that user as value. Initially both the dictionaries are empty.

Dictionary 1: If a username does not exist in the keys of dictionary 1, then it is added to the dictionary 1 with tweet count of the user as value. If a username already exists in the dictionary 1, then its value is updated as the sum of value in the dictionary and value returned from map function.

Dictionary 2: If a username does not exist in the keys of dictionary 2, then it is added to the dictionary 2 with maximum tweet length as value. If a username already exists in the dictionary 2, then its value is updated to the new value if the new value is greater than the previous value in the dictionary.

Both the dictionaries are sorted based on the values. The key with highest value in dictionary 1 is the user who tweets the most. The 5 keys with top 5 highest values are the top 5 longest tweeters and the 5 keys with least 5 values are the bottom 5.

PS – For this problem, the entire Twitter data set was not used. It was taking too long and it was failing at the reduce step (memory issues). However, map-reduce was run for subset of the Twitter data set and the result shown below is for that subset.

**Here’s the solution obtained:**

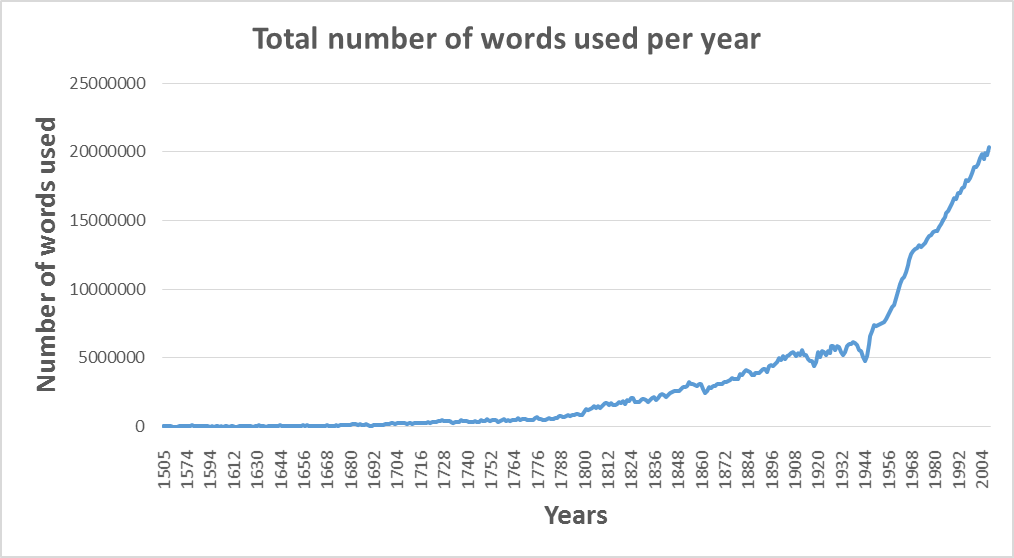
User that tweeted the most: OhioHealthJobs      
Top 5 longest tweeters (with length of tweet): [('HappyCaraT', 254), ('CyborgScreams', 258), ('Atmvn', 327), ('Rubenslb13', 366), ('HarveyMCMXCIV', 446)]      
Bottom 5 longest tweeters (with length of tweet): [('Tayorrann\_12', 1), ('mmitchelldaviss', 1), ('christi2nOK', 1), ('DreamsxHoops', 1), ('myialynette', 1)]

(Google 1-gram – Question 1) The fifth question that we chose was to plot the total number of words used for each year available in Google 1-gram data. We solved this problem by using Hadoop in streaming mode. We need to find out the total number of words used for each year available in the google 1-gram data. In the map function (map5.py) we have a dictionary where year is the key and the count of words in that year is the value. Initially this dictionary is empty. If a year is not found in the keys of the dictionary, that year is added with the value of the year as 1. If a year is already present in the keys, then its value is incremented by 1. The mapper finally returns all the years present in the chunk of data available to the mapper as keys and their corresponding word count as values. The reduce function (reduce5.py) aggregates all the years and their counts. It works in a similar manner to the map function. It has a dictionary that is initially empty. If a year is not found in the dictionary, then it is added to the dictionary with the count of words from the mapper as value. If a year is already present in the dictionary, then its value is incremented by the count of words of the year returned by the mapper function.

**Here is the output (Few lines of the output):**

1948    6975779  
1949    7369983  
1942    5489436

**Here is the plot of total number of words against the year:**



(Google 1-gram – Question 3) The sixth question that we chose is to plot the average word length for all years available in Google 1-gram data. We solved this problem by using Hadoop in streaming mode. To calculate average word length for every year, we need to find the total length of all the words and total count of all the words in a year. In the map function(map6.py), we have a dictionary where year is the key and value is a list with length as the first element and count as the second element. Initially the dictionary is empty. If a year is not found in the keys of the dictionary, the year is added with value as a list with length of the word being the first element of the list and 1 being the second element. If a year already exists in the dictionary, then its value is updated accordingly. The reduce function (reduce6.py) aggregates the values from all the mappers. Similar to map function, the reduce function also has a dictionary that is empty in the beginning. If a year is not found in the keys of the dictionary, the year is added with value as a list with total length of all words, from a specific mapper, being the first element of the list and total count of all words in the mapper being the second element. If a year already exists in the dictionary, then its value is updated by summing up elements of the list with first and second element of the value returned by map function (which is also a list).

**Here is the output (few lines of the output):**

1948    9.88216785538  
1949    9.87452576214  
1942    9.82614151982  
1943    9.76251310739  
1940    9.85197999909

**Here is the plot of average word length for all years:**

