# Cloud Computing P4 Report
## *Spark for Real Problems*

*Laxmi Janakiraman*
*Sayali Pendharkar*
*Suprabha Hegde*

**Google 1 gram data:**

**Q3:**

Plot the average word length for all unique words for all years available. Year on x-axis, average word-length on y-axis.

**Technologies and languages:**

Spark
PySpark,Python

**Analysis method:**

The question required determining the average length of unique words in each year in Google 1 gram data that has been provided.
The data is of the format:

**Word Year Number of occurrences of the word Documents containing the word**

This could be done by calculating the lengths of all words in a particular year and dividing it by the number of words that have occurred in a particular year.

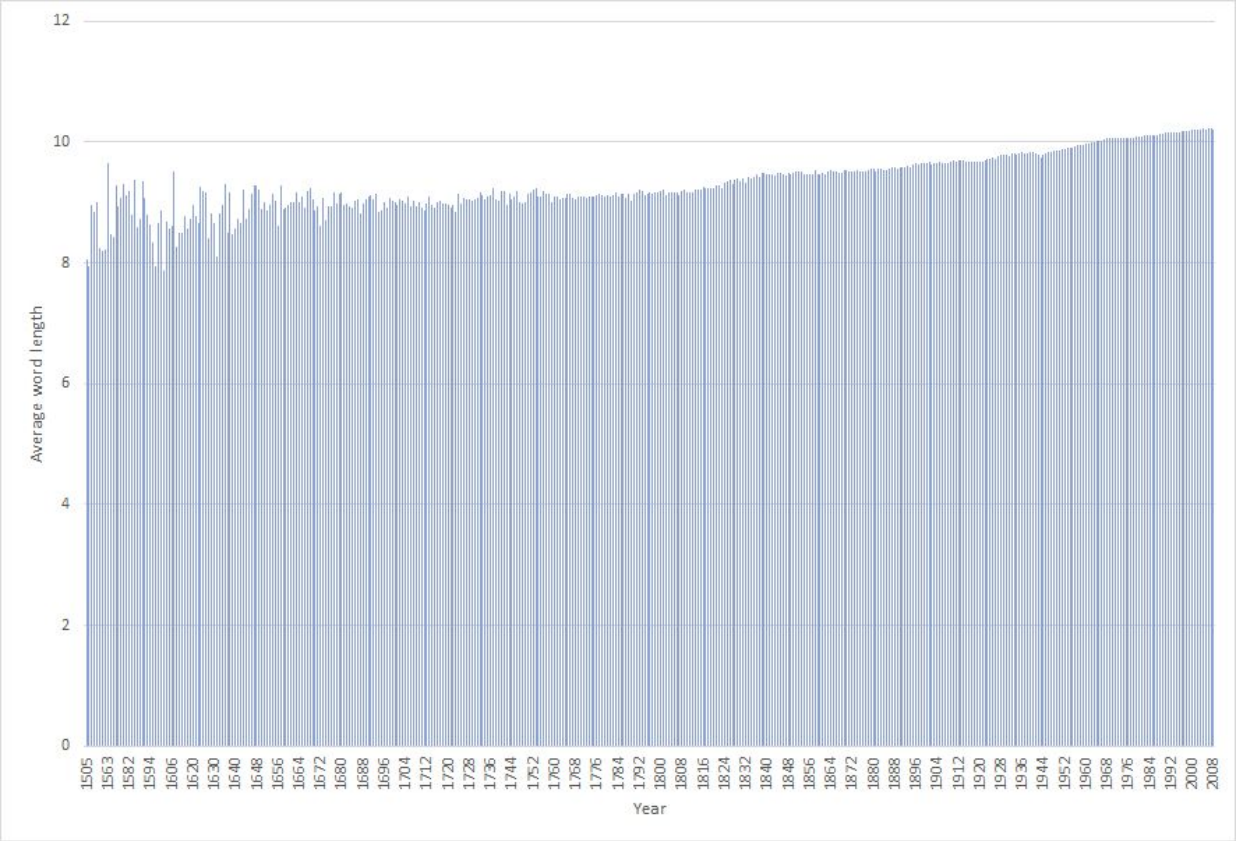
**Pseudo Code:**

1.Read each line of data & split based on tab
2.Retrieve the year, word and no of occurrences of each word
3.for each line of data:
    Call get_WLength function
    Store returned key value pairs in wordLength(RDD)
    <year,word length>
4.use reduceByKey to aggregate the word lengths for each year
    <year,sum of all word lengths in that year>
5.for each line of data:
    Call get_Occurences function
    Store returned key value pairs in wordCount(RDD)
    <year,occurrences of word in that year>
6.use reduceByKey to aggregate all the occurrences for each year
    <year,total number of words in that year>
7.for each year calculate average by taking the quotient of sum of all word lengths and total number of words in that year
8.Store the results in a file
    <year,average word length>

**Results:**
The file contains a list of key value pairs with year as key and the value as the average word length for that particular year.

The Graph 1,plots the results derived from the data.

**Graph:**




**Graph 1: Average Word Length for each year**

**Google 1 gram data:**

**Q1:**

For each year available, plot the size of the set of words used. Year on the x-axis, number of words on y-axis.

**Technologies and languages:**

Spark
PySpark,Python

**Analysis method:**

The question required determining the number of unique words in each year in Google 1 gram data that has been provided.
The data is of the format:

**Word Year No_Of_Occurences No_Of_Documents**

This could be done by calculating the number of times a particular year has occurred in the data since every word will be associated with a year. So to find the number of unique words in the data set according to the year we have to find the number of times a year has occurred.

**Pseudo Code:**

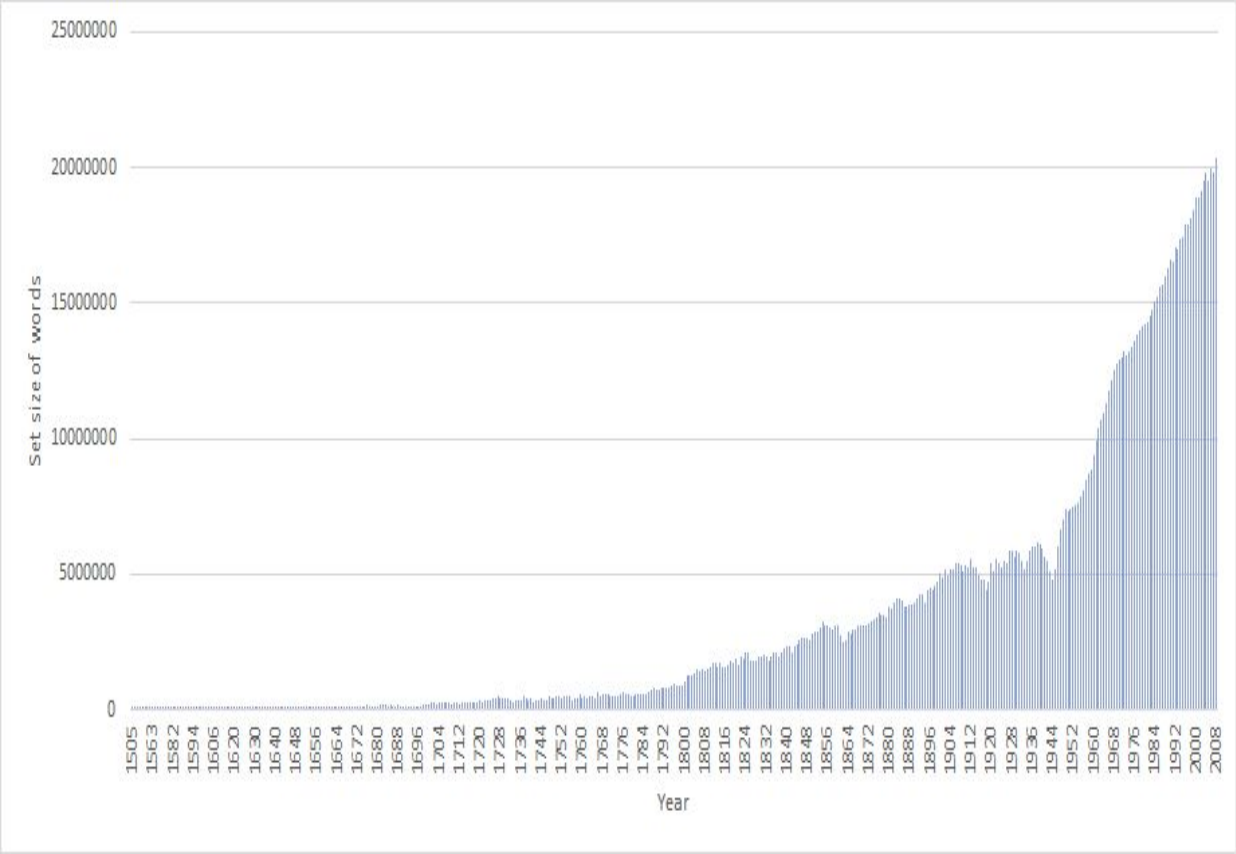
1. Read each line of data & split based on tab
2. Retrieve the year
3. for each line of data:
    Store the key value pair as <year,1>
4. Use reduceByKey to aggregate the words for each year
    <year,total number of words in that year>
5. Sort the results according to the key (year)
6. Store the results in a file
    <year,total number of words in that year>

**Results:**

The file contains a list of key value pairs with year as key and the value as the number of unique words for that particular year.

The graph below,plots the results derived from the data.

**Graph:**




**Graph 2: Size of the set of words used for each Year**

**Twitter data:**

**Q10:** Detect the proportion of bad words in a tweet. Plot bad word proportion by hour for all 24 hours.

**Technologies and languages:**

Spark
PySpark,Python

**Analysis method:**

The question demands for a proportion of bad words in a tweet and to plot a graph with bad word proportion by hour for all 24 hours. Initially a bad word list is built. In our case we have taken the list from http://www.hyperhero.com/en/insults.htm .

**Assumption:**

All the words present in the list (http://www.hyperhero.com/en/insults.htm) are considered to be bad. From 'created_at' value, if the hour is 07:00:00 to 07:59:59 we have considered it as 7th hour.

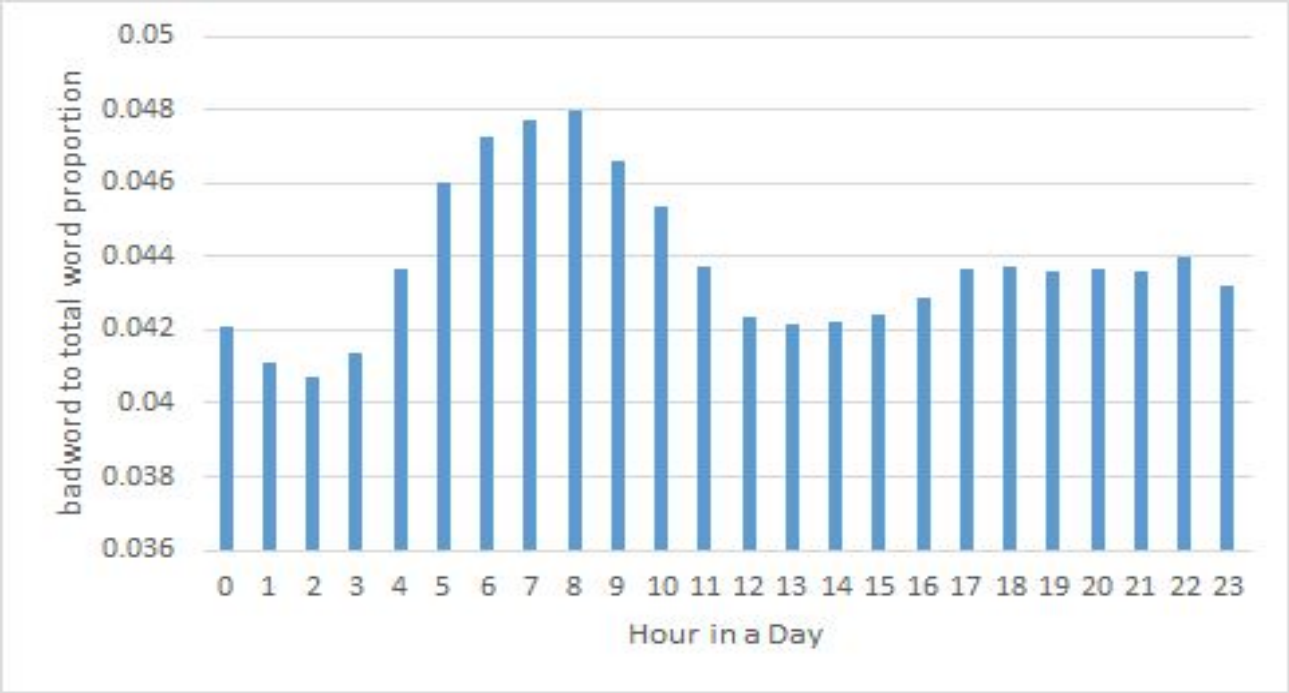
**Pseudo Code:**

1. Read the twitter data.
2. Decode json and get the tweet hour and tweet text.
3. Calculate total word present in a tweet and in which hour. Convert this to a dictionary and add all the values for a key i.e for every hour get the total word count.This is implemented using reduceByKey.
4. Calculate the total bad word present in a tweet and in which hour.Convert this to a dictionary and add all the values for a key i.e for every hour get the total bad word count.This is implemented using reduceByKey.
5. The two dictionary created in the above step is divided by one another to get the bad word proportion in a tweet.

**Results:**

The result is a list of proportion of bad words present in all tweets for each hour (0-23) in a day. A graph plotting the expected bad word proportion to hour is shown in graph 3.

**Graph:**




**Graph 3: Bad word proportion by hour**