Cloud Computing Project 4 ngram Problem 1

# Problem Statement

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

# Approach

The set of words used in each year was found in 5 stages. Initially, it was understood that the ngrams data set had a line with an ngram and a year, and that there could be duplications of the same ngram for the same year. So, a set of map and reduce tasks were created to find the distinct ngrams for every year. While it was later discovered that the ngrams data set does not have duplicates, the code still works for finding the set of ngrams used in each year.

The first stage of the solution mapped each line to an (<ngram>, <year>) tuple. The second stage of the solution used the “reduceByKey()” function to reduce all ngram year tuples to a tuple of the form (<ngram>, <tab delimited year list>). This reduce served to remove any duplicate ngrams for a year. The third stage used the “flatMap()” function, and mapped each ngram year list tuple to a tuple of the form (<year>, 1). The fourth stage used the “reduceByKey()” function to sum up the count for each year, thus producing the total number of ngrams for a year. The fifth stage used the “sortByKey()” function to sort all of the year tuples by year. The result was then saved on hdfs in a local folder.

# Results

**The ngrams set size for the latest years was:**

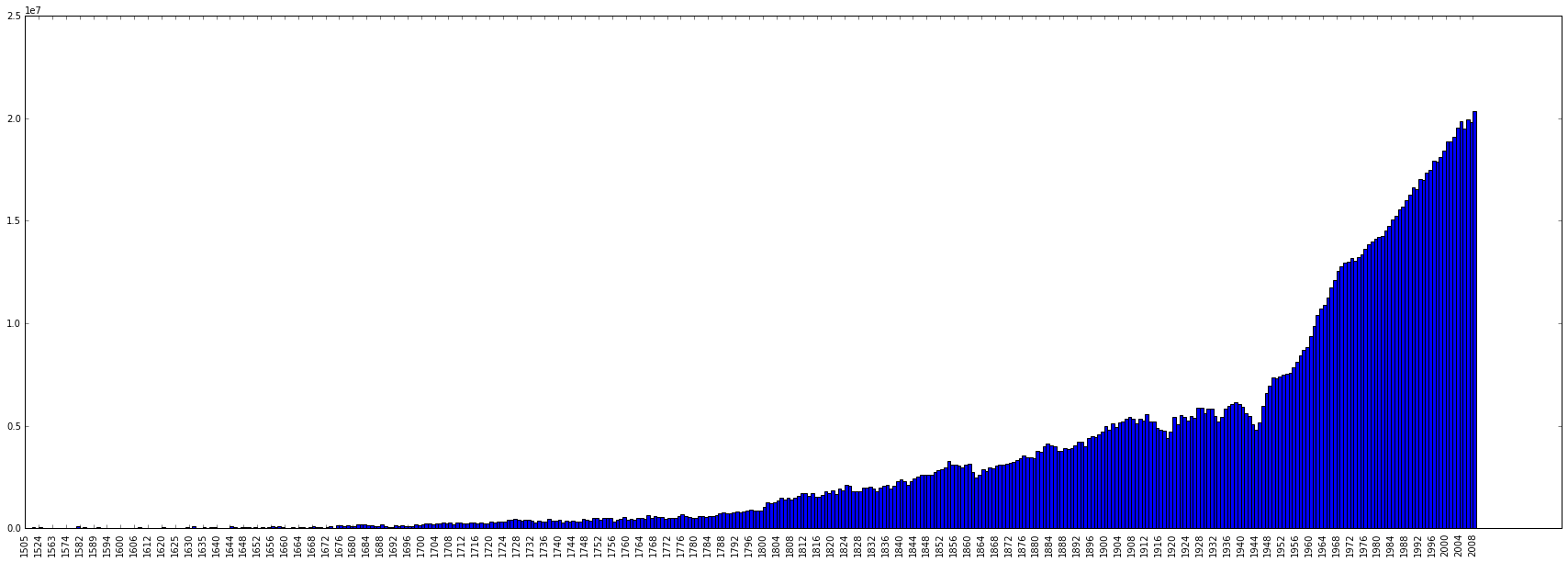
2005: 19510978 ngrams

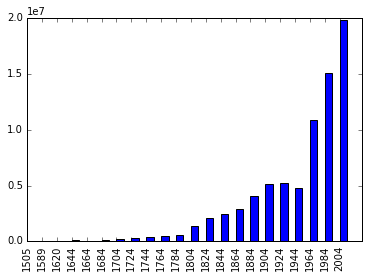
2006: 19952152 ngrams

2007: 19804152 ngrams

2008: 20355300 ngrams

**Full ngrams vs year plot:**

****

**Condensed ngrams vs year plot.**

Cloud Computing Project 4 Twitter Problem 7

# Problem Statement

For those tweets with 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?

# Approach

The centroid, total, and percentage of all tweeter’s locations was found by using various Spark MapReduce actions. For the centroid search, a 65 mile threshold was adapted (~1 degree of latitude = 65 miles laterally). Firstly, the file was invoked by .textFile('hdfs://path/to/file'), a RDD was created, and a count of total tweets was measured. The RDD was then invoked by map(trim\_locations), where trim\_locations was the initial stage that truncated locations to a < ~65 mile padding and then count the locations that appear multiple times. The next step was to .filter(filter\_locations) null locations and then invoke .count() to show the total number of tweets with locations. A simple 1.count()/2.count() was invoked and the total percentage of tweets with locations versus total was found. The next step, .reduceByKey(reduce\_locations), was invoked to sum the values of the keys that were the same. The final step was to determine which key had the maximum value by using .max(max\_locations) and how many it had. The results are displayed in the Results section of this paper.

# Results

**Centroid**: ('39,-84', 320007)

**Total Tweets with Location**: 1868302

**Percentage**: 30.7271%