Data Science

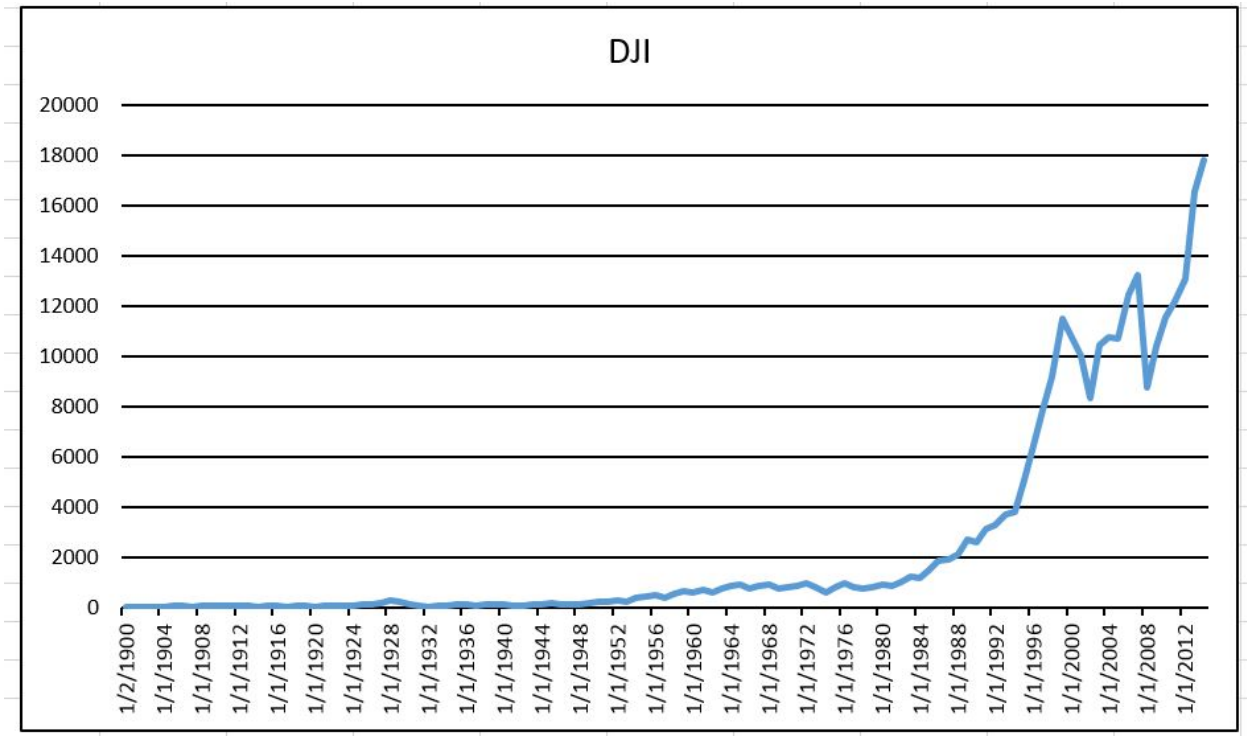
Assignment 1- Coping with a Data Science Task

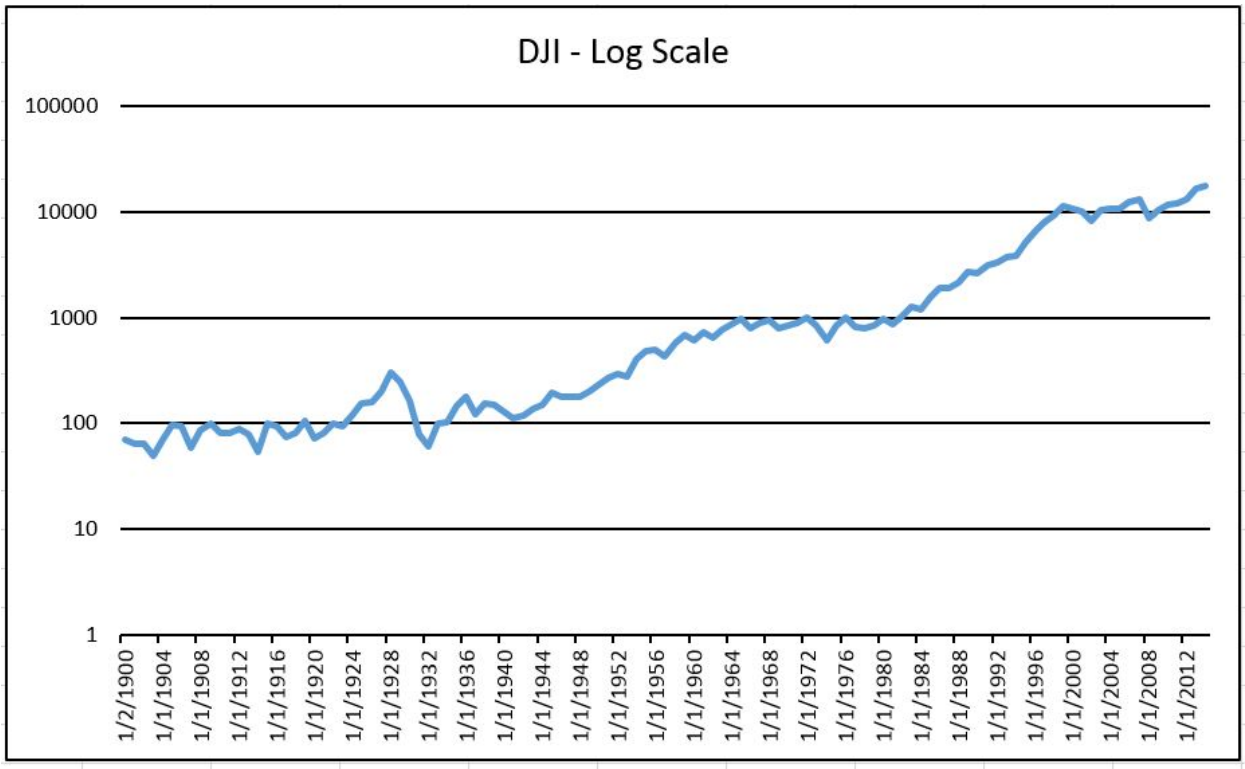
Paul Lewis & James Kizer

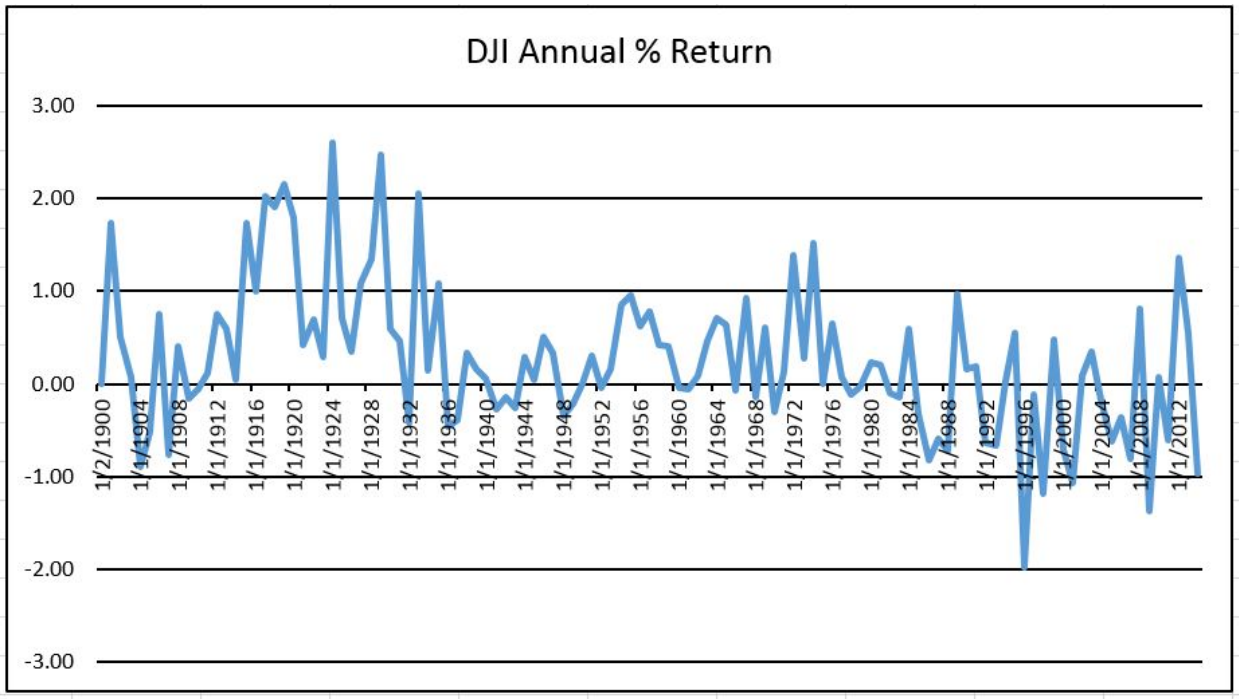
Task 1

We decided to use the Economy as are test subject. More specifically we looked at words associated with economic growth as indicators of differents stages in the economic cycle. Economies generally move through a boom-bust cycle over roughly a ten year period. We try to find words associated with these periods of high and low economic growth.
We used the Dow Jones stock index as a proxy for the state of the economy. A more accurate measure would be GDP, but statistics aren't available so far back, whereas they started calculating the Dow in the 1890's. There is high enough correlation between the stock market and the state of the actual economy to make it a suitable benchmark.
Finally we wondered if any words could actually be used to predict (as opposed to just correlate with) a downturn in the economy.

The first graph shows the DJI performance, dating back to 1900. The second shows the same data, but In particular the 'Great Depression' of the early 1930's is now easily seen. plotted on a logarithmic scale so that the earlier parts of the data are more visible.The third graph plots the annual percentage returns, showing the peaks and troughs of the index.
Even though more granular data was available, we bucketed the returns into annual blocks to correspond to the Google ngram bucketing.
We are hoping to find a correlation between ngram frequency of our chosen words and the peaks and troughs of the third graph.


DJI
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
1/2/1900
1/1/1904
1/1/1908
1/1/1912
1/1/1916
1/1/1920
1/1/1924
1/1/1928
1/1/1932
1/1/1936
1/1/1940
1/1/1944
1/1/1948
1/1/1952
1/1/1956
1/1/1960
1/1/1964
1/1/1968
1/1/1972
1/1/1976
1/1/1980
1/1/1984
1/1/1988
1/1/1992
1/1/1996
1/1/2000
1/1/2004
1/1/2008
1/1/2012

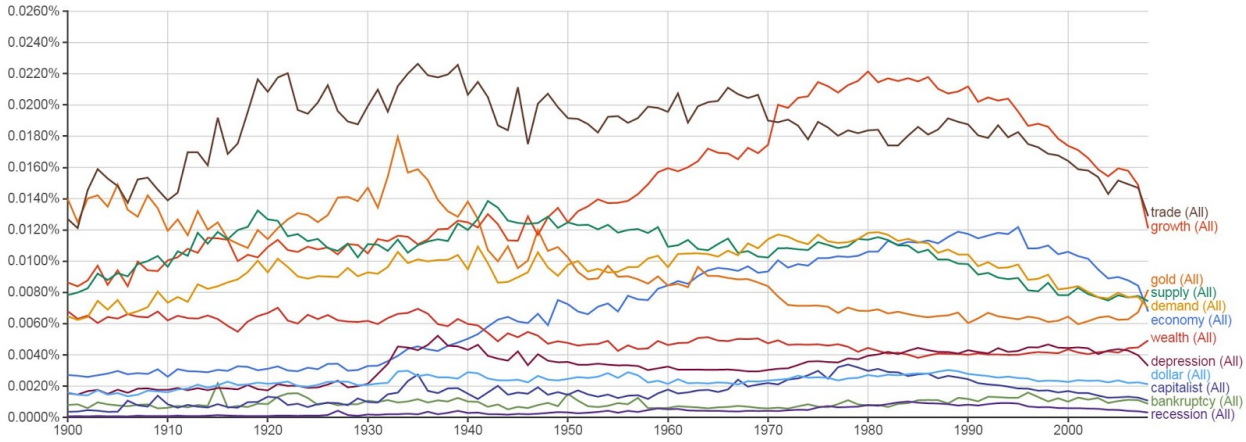

DJI - Log Scale
100000
10000
1000
100
10
1
1/2/1900
1/1/1904
1/1/1908
1/1/1912
1/1/1916
1/1/1920
1/1/1924
1/1/1928
1/1/1932
1/1/1936
1/1/1940
1/1/1944
1/1/1948
1/1/1952
1/1/1956
1/1/1960
1/1/1964
1/1/1968
1/1/1972
1/1/1976
1/1/1980
1/1/1984
1/1/1988
1/1/1992
1/1/1996
1/1/2000
1/1/2004
1/1/2008
1/1/2012


DJI Annual % Return
3.00
2.00
1.00
0.00
-1.00
-2.00
-3.00
1/2/1900
1/1/1904
1/1/1908
1/1/1912
1/1/1916
1/1/1920
1/1/1924
1/1/1928
1/1/1932
1/1/1936
1/1/1940
1/1/1944
1/1/1948
1/1/1952
1/1/1956
1/1/1960
1/1/1964
1/1/1968
1/1/1972
1/1/1976
1/1/1980
1/1/1984
1/1/1988
1/1/1992
1/1/1996
1/1/2000
1/1/2004
1/1/2008
1/1/2012

We used an online lookup for words associated with the word ‘economy’ and then filtered this list by using the ngram viewer to filter for words with a higher occurrence. This method produced the following list of words:

economy, wealth, bankruptcy, gold, capitalist, depression, dollar,growth, supply, demand, recession, trade

An overview of the results for the 20th century are as follows:



One other word of interest we found was “**stockmarket**” which gave the following results:

This graph is interesting as it has two distinct spikes which appear to correspond to the bid stock market crashes of 1929 and 1987.

Task 2:

The data science task we chose was develop a tool to understand the correlations between 1-grams and how those correlations may change over time.

We configured a cluster of 5 m3.large EC2 nodes, 1 master and 4 workers. We used the spark-ec2 script. The process took ~45 minutes to complete, but it was relatively straightforward.

Our data source is the Google ngram database that is publicly available in an AWS S3 bucket. Although a seemingly straightforward task, this took much longer than expected.

Spark has APIs for converting files of various types and locations to RDDs. It took me roughly 30 minutes to obtain access to the file due to a lack of understanding of how permissions work with public S3 buckets. After accessing the data, it was apparent that it was compressed. To make a long story short, it took several more hours to get the correct combination of settings and packages installed in order to access the data.

The data are defined in rows containing:

- ngram
- year
- # of distinct volumes
- total # of mentions

The data wrangling is as follows:

1. Rows are aggregated by ngram
2. 1grams are filtered if they are not strictly alpha characters and then are converted to lowercase
3. A vector is created for total # of mentions for years 1908-2008
4. 1grams are filtered if they do not contain at least one mention for every year in 1908-2008
5. Vectors are normalized (l2)

We are left with an RDD containing (1gram, vec) pairs. We can now (or at least we thought) create the cartesian product of the of the previous RDD and perform some correlation analysis between 1grams.

We ran into issues serializing the Apache Commons PearsonsCorrelation object, so we ended up hand rolling a correlation implementation.

We initially tested with 1000 entries taken from the dataset after data wrangling, resulting in < 5E5 distinct cartesian pairs of words to analyze. This ended up taking roughly 7-8 mins to run. We then tried running on the entire dataset. After data wrangling, we were left with ~100K

(1gram, vec) pairs, and thus ~5B distinct cartesian pairs of 1grams. Obviously this is too many values to evaluate, so we chose to take a random sample of 1% of the entries. We were left with 1717 (1gram, vec) pairs and 1473186 distinct cartesian pairs.

We wanted to understand how correlated words were over the entire test period. For each pair, we computed the correlation of all years of the vectors. We got the following results:

The 10 most positively correlated are:

1. ((first,time),0.9974267761655861)
2. ((brightly,look),0.9965929044742745)
3. ((about,post),0.9964981014550388)
4. ((look,overheard),0.9963899506501526)
5. ((about,asking),0.9955064490719524)
6. ((boost,messing),0.9954599802113887)
7. ((look,shimmering),0.9953915964410327)
8. ((cracked,look),0.9952865037239669)
9. ((about,time),0.9952565586311044)
10. ((asking,enthralled),0.9950235137502121)

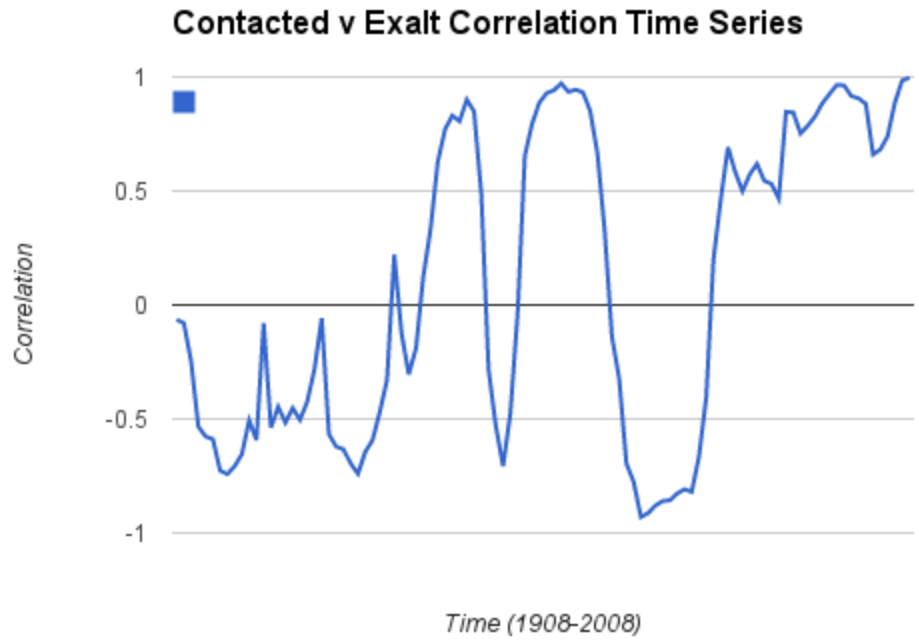
The 10 most negatively correlated words are:

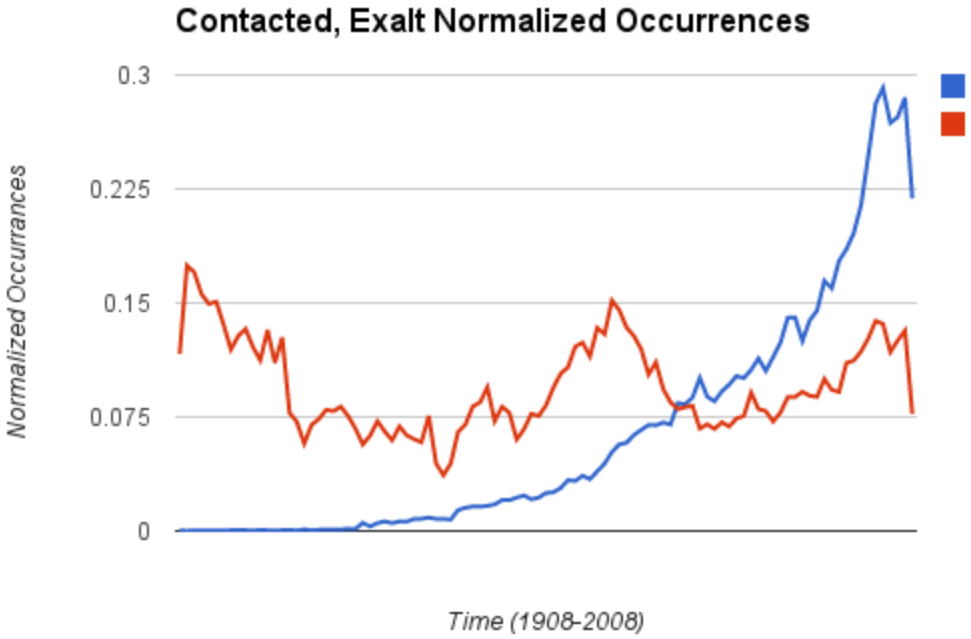
1. ((gayety,overemphasized),-0.7937690788670532)
2. ((chlorobenzene,gayety),-0.7767228496602299)
3. ((benzine,overemphasized),-0.7685527626371156)
4. ((financed,gayety),-0.7420196579258719)
5. ((gayety,unhurriedly),-0.7413280001255854)
6. ((benzine,slogans),-0.7401479513539186)
7. ((benzine,financed),-0.7358610141162433)
8. ((benzine,inescapably),-0.7285090768207056)
9. ((gayety,largescale),-0.7279940499667986)
10. ((gayety,slogans),-0.7260442811984937)

The 10 least correlated words are:

1. ((breviaries,tokaido),1.4139074040218732E-7)
2. ((ofte,oilcan),1.693814509139229E-7)
3. ((equipments,nypl),5.231286384861425E-7)
4. ((cranch,hayden's),7.473636890477439E-7)
5. ((leyendas,wek),8.278263167107728E-7)
6. ((cobh,kinsman's),8.991900490989341E-7)
7. ((paspalum,tanze),1.0297711540016348E-6)
8. ((floatable,potero),1.498552573617868E-6)
9. ((reconstructions,tentacular),2.3222843277418605E-6)
10. ((charlotta,mahomedans),2.781438125026983E-6)

We wanted to understand how word correlations changed over the test period. For each pair, we computed a 10 year sliding window of the correlation between words. The pair with the highest standard deviation was (contacted, exalt). See graphs below:





The pair with the lowest standard deviation was (contacted, exalt). See graphs below: