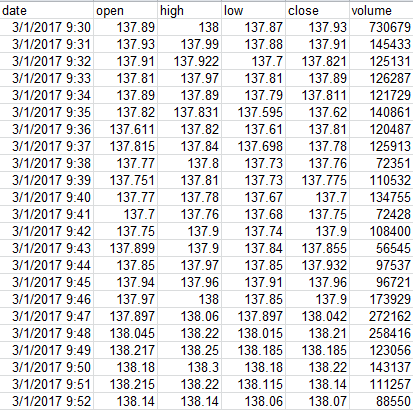
**CSCI63 Final Project Report: NLTK and Spark Near-Real Time Text Analysis**

**Intraday Data Implementation**

The ***download\_ intraday\_bloomberg.py*** python script imports a class called pybbg which is defined in the provided ***pybbg.py***. The pybbg class handles the interfacing with Bloomberg API. The script generates three csv files and saves them to the directory: H:/Course Docs/Big Data/Final Project/Results/IntradayPrices/. The following intraday information with minute-by-minute frequency on the three stocks is downloaded: open price, high price, low price, closing price, and volume. Intraday data is downloaded by running “python download\_intraday\_bloomberg.py” in command prompt. We will use this intraday data later in analyzing the relationship between StockTwit sentiment on those stocks and their respective prices.

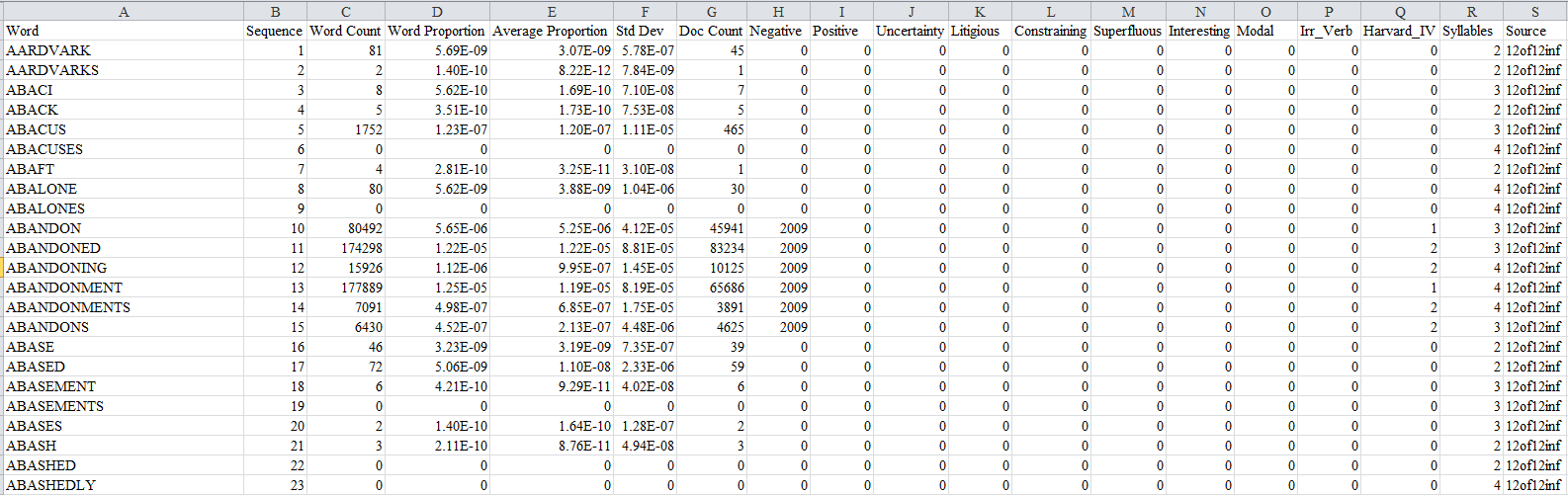
*File Sample: H:/Course Docs/Big Data/Final Project/Results/IntradayPrices/IntradayPrice.AAPL.csv*



**Bag-Of-Words Model Overview**

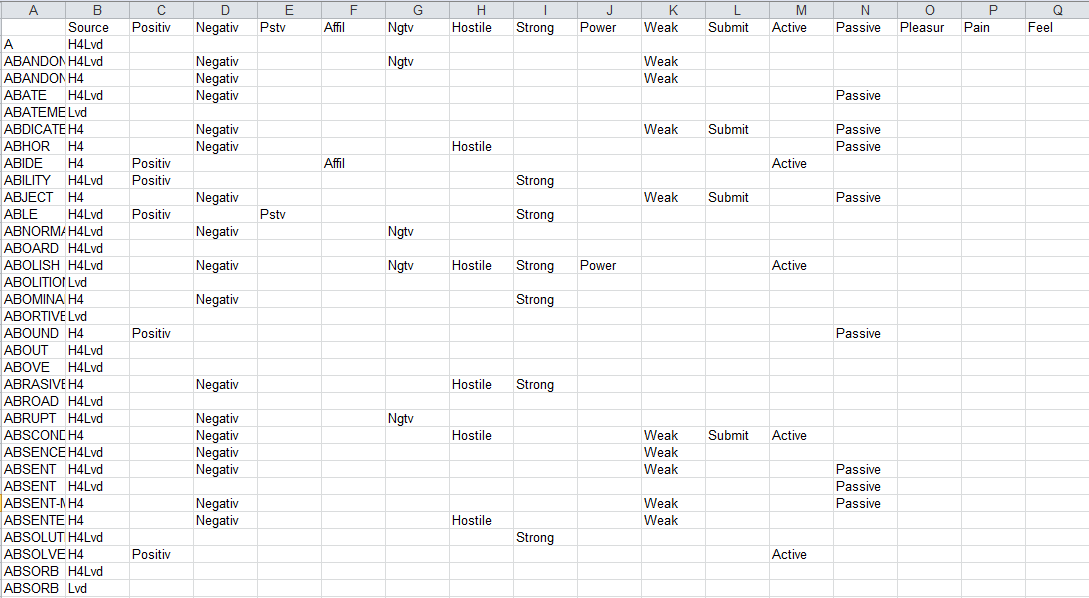
Perhaps the simplest way to extract information from text is to perform sentiment or tone analysis. We apply a bag-of-words approach: each twitter message is cleaned down to a list of words. We then apply a word classification dictionary to classify each word and either positive, negative, or neutral. The most commonly used dictionary for textual analysis is the Harvard Psychosociological Dictionary, or Harvard-IV-4 TagNeg (H4N). It is used for word classification of the English language across multiple domains. However, Loughran and McDonald [2011] found that it substantially misclassifies words when gauging tone in financial applications. The two professors from University of Notre Dame found that 73.8% of negative word counts according to the Harvard list were attributable to words that are not typically negative in a financial context (e.g. *tax, cost, capital, board, liability, foreign*, etc.). In addition, they used different variations or inflections of words (e.g. *accidental*, *accidentally*, and *accidents* from the base word *accident*). Their paper introduced a second financially-focused set of dictionaries called the LoughranMcDonald master dictionary. We noticed that LoughranMcDonald was quite imbalanced in the size of the negative and positive dictionaries: FinNeg (2,355 words) versus FinPos (354 words). Perhaps this provides some empirical evidence that the Anna Karenina principle applies in Finance: “Happy families are all alike; every unhappy family is unhappy in its own way.” General information about this dictionary is available here: <http://www3.nd.edu/~mcdonald/Word_Lists.html>. The LoughranMcDonald (Financial) word dictionary was downloaded from this website: <http://www3.nd.edu/~mcdonald/Word_Lists_files/LoughranMcDonald_MasterDictionary_2014.xlsx> and saved to the following location: H:/Course Docs/Big Data/Final Project/Docs/LoughranMcDonald\_MasterDictionary\_2014.xlsx. Columns H and I indicate negative and positive financial words, respectively.

*H:/Course Docs/Big Data/Final Project/Docs/LoughranMcDonald\_MasterDictionary\_2014.xlsx*



Not to be outdone by University of Notre Dame, Harvard updated their generalist word lists with better handling of word inflections. Whereas the original Harvard-IV-4 TagNeg list contained 1,045 positive and 1,160 negative words, the updated list contains 1,637 positive and 2,006 negative words. General information about this list is available here: <http://www.wjh.harvard.edu/~inquirer/spreadsheet_guide.htm>. The Harvard word dictionary was downloaded from this website: <http://www.wjh.harvard.edu/~inquirer/inquirerbasic.xls> and saved to the following location: H:/Course Docs/Big Data/Final Project/Docs/inquirerbasic.xls. We made a slight change to this file. Because different connotations of the same word were listed as multiple rows in the spreadsheet, we removed the ‘#X’ tag which denotes multiple entries, and the first entry of each word was used. We ran two models, one for each dictionary, in order to examine whether the domain-specific dictionary or the general dictionary would produce a more accurate prediction on our StockTwits dataset.

*H:/Course Docs/Big Data/Final Project/Docs/inquirerbasic.xls*



Once word classification is complete for each twitter message, we calculate a sentiment score based on the number of positive words minus the number of negative words divided by the sum of positive and negative words. The following is a mathematical representation of our calculations. Each tweet is denoted and each word in the tweet denoted . Sentiment score for a single tweet is denoted . A categorical “Sentiment” is defined in accuracy testing and denoted .

The sentiment score can range between -1 and +1. If a tweet contained three positive financial words and two negative words, then the sentiment score would be +0.2.

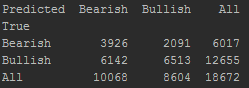
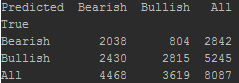
When calculating sentiment scores for a single day, we defined a simple measure to be the sum total of sentiment scores that day divided by the number of tweets. The shortcoming of the simple daily sentiment score is that the magnitude of sentiment on days where there is an above average number of tweets is not taken into account. Our final metric is a weighted sentiment score, which takes the simple sentiment score and scales it by the number of tweets that day divided by the average number of tweets per day over the entire period.

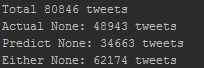
In mathematical terms, let the total number of days be denoted and the number of tweets on day be:

StockTwits provides us with messages and corresponding tags (“Bullish”, “Bearish”, or “None”), which allows us to test model accuracy very easily. We use the dictionary to count the number of positive and negative words, as defined by whichever word dictionary we use. If the number of positive words is greater, then the tweet is predicted to be “Bullish”. If the number of negative words is greater, then the tweet is predicted to be “Bearish”. For the purpose of testing accuracy, we ignore all tweets that are tagged with “None”. This allows for some tweets to be bullish or bearish in sentiment, but for whatever reason the user decided not to tag them.

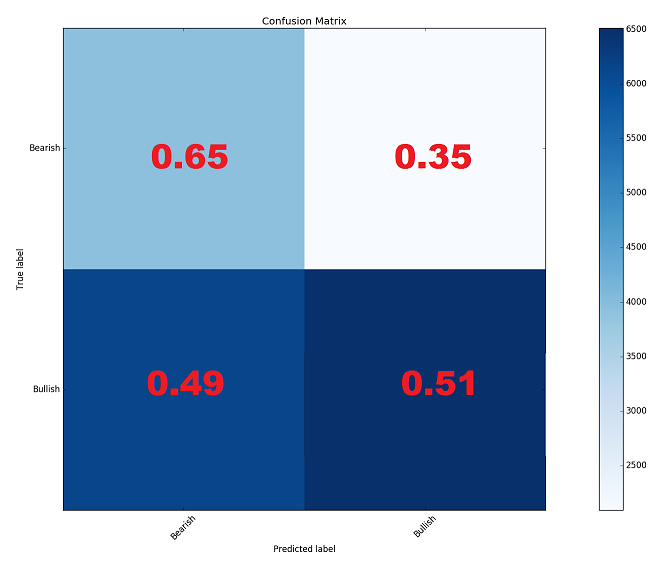
Because this particular model does not need to be trained, we used every tweet we had (AAPL, FB, and TSLA from 3/1 to 4/30) to test accuracy. We can generate the summary information, confusion matrices, and output file test\_dict\_output.csv below by running in command prompt: *python dictTesting.py*. The variable in Line 9 can be set to “Harvard” or “Financial”, for specifying the Harvard and Financial dictionaries respectively.

**Harvard Financial**

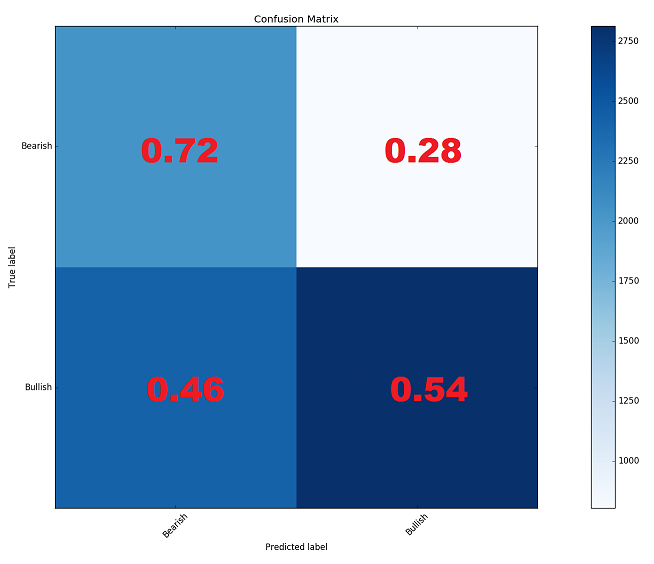
** **

** **

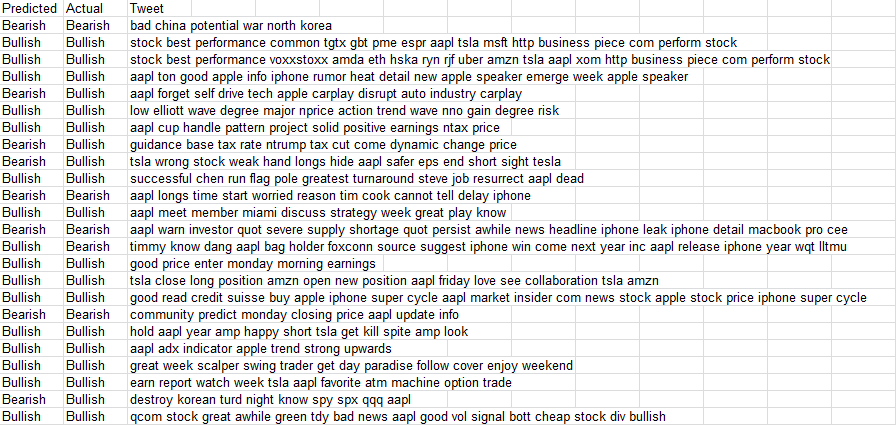
*Harvard Dictionary Confusion Matrix*

****

*Financial Dictionary Confusion Matrix*

****

*H:/Course Docs/Big Data/Final Project/Results/Sentiment Analysis-1/test\_dict\_output.csv*

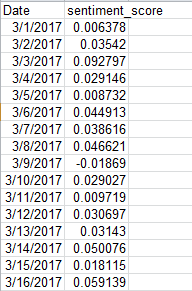
****

On Bullish tweets, the Harvard dictionary accuracy of 51% was not much different from the Financial dictionary’s accuracy of 54%. Neither dictionary does much better than a coin flip. On Bearish tweets, however, the Harvard dictionary is able to correctly predict 65% of tweets. The Financial dictionary again does better at 72%.

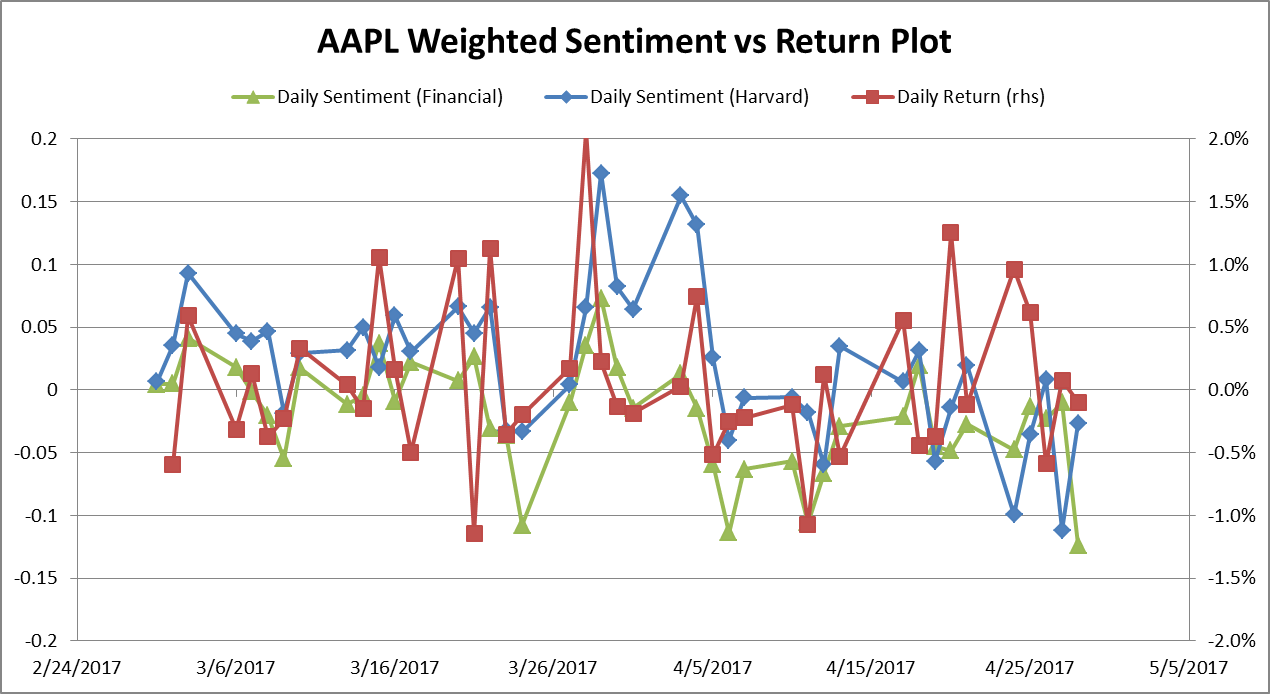
The number of (non-neutral) forecasts made is also an important metric to consider. Single tweets tend to be a very condensed set of words and many of the tweets do not have a single word which classifies as either bullish or bearish. Out of all 80,846 tweets considered, only 48,943 of them had user-provided StockTwit tags on them which we could use to test accuracy. Of those, the model using a Financial dictionary made non-neutral forecasts on 8,087 tweets. The model using the general Harvard dictionary made non-neutral forecasts on 18,672 tweets. We are able to produce sentiment scores on twice as many tweets using the general Harvard dictionary (38% vs 17% of all possible).

Since using test accuracy to determine the better dictionary model was inconclusive, we looked at how the weighted daily sentiment scores of the two dictionaries forecasts compared to actual daily returns of the underlying stock. Whereas our accuracy test was restricted to only tweets which had user-provided Bullish or Bearish tags, for this part of the analysis we were able to calculate a sentiment score on every tweet. Note also that we are calculating sentiment score with range -1 to +1 on every tweet rather than just a categorical “Bullish” or “Bearish” for the earlier analysis. As an input to this test, we again use all tweet data on all three stocks. To generate the output data files, we can run in command prompt: *python* *dictScoreAll.py*. This produces 6 files (one simple daily sentiment score and one weighted daily sentiment score for each of the three stocks) for the particular dictionary selected on Line 9 of the code.

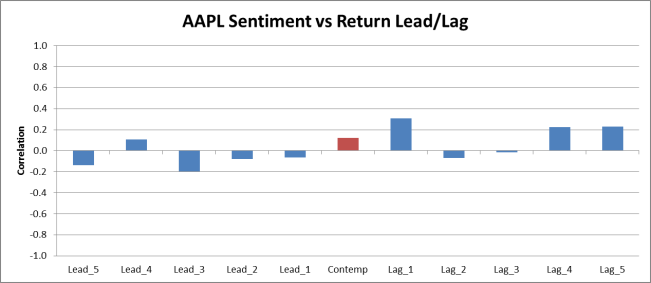
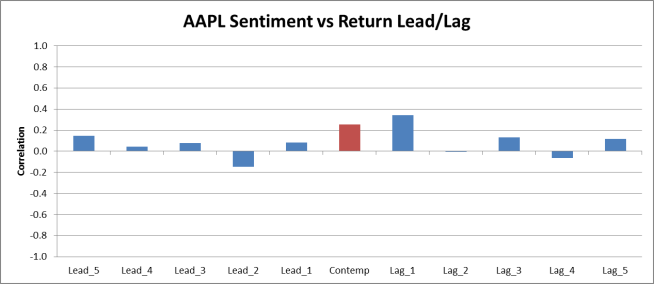
*H:/Course Docs/Big Data/Final Project/Results/Sentiment Analysis-1/dict\_output\_weighted.AAPL.Harvard.csv*

**

Using this data, we picked AAPL stock and plotted the weighted daily sentiment scores for both dictionary methods against the daily percentage returns on that same day. From reading this chart, there does not appear to be much of a relationship. We also created a correlation lead/lag charts which look at both contemporaneous correlations as well as the correlations by shifting the sentiment time series ahead or behind by a day or multiple days. The lead/lag chart below shows the edge goes to the Financial dictionary over Harvard dictionary due to a higher contemporaneous correlation. There does not seem to be a strong leading or lagging relationship between AAPL sentiment and the stock price returns, at least in our sample of two months data.

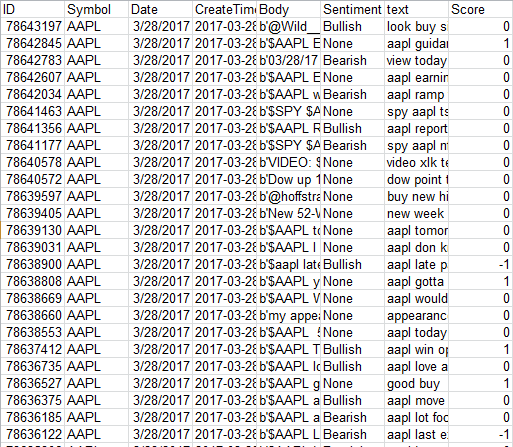


**Harvard Financial**

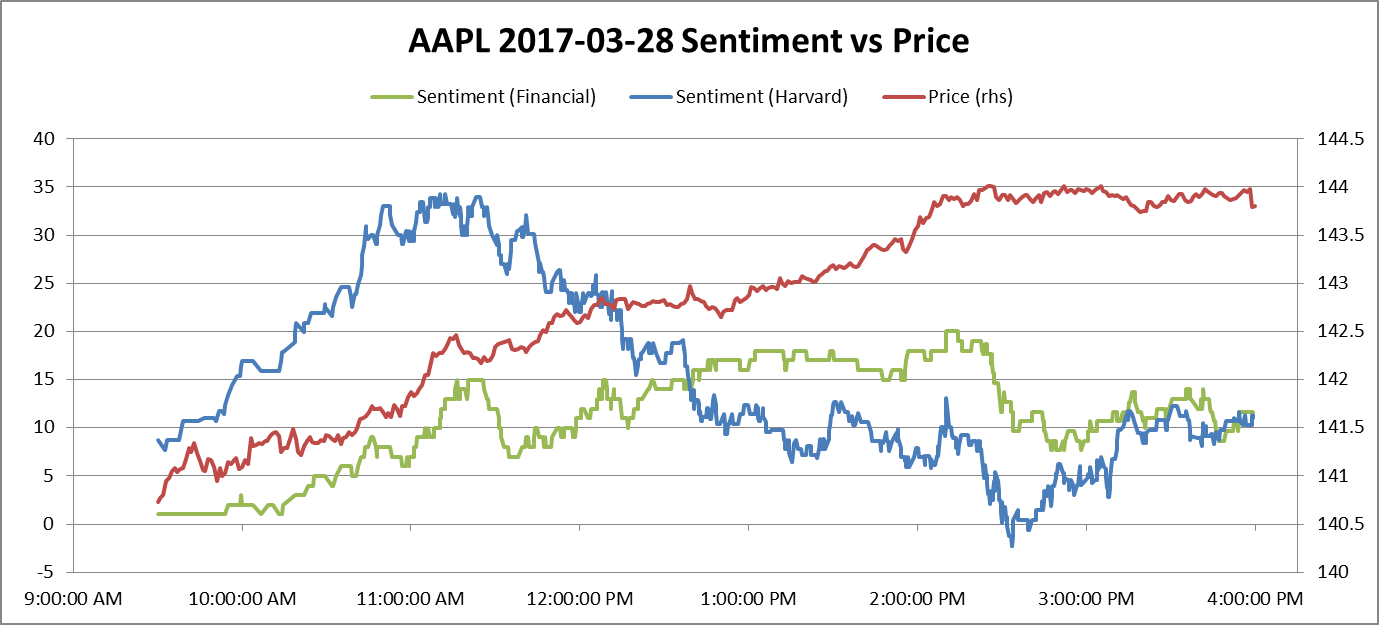
 

Our final test on the Bag-Of-Words model was using the intraday data we downloaded earlier. We calculated a cumulative sum of sentiment scores starting from when the stock market opened at 9:30AM on March 28th, 2017 until market closed at 4PM. The output file test\_dict\_one\_output.csv is generated by running in command prompt: *python dictScoreOne.py*. The only input file for this was the AAPL file of tweets, and the example result below was using the Financial dictionary:

*H:/Course Docs/Big Data/Final Project/Results/Sentiment Analysis-1/* *test\_dict\_one\_output.csv*



Below, we plotted in Excel our data series and noticed that that while there definitely wasn’t a strong relationship between sentiment and price, it appears that at least in this instance, the Financial dictionary does better than the Sentiment dictionary in Bag-Of-Words analysis. Overall, we found that the Financial dictionary was slightly better than the general Harvard dictionary.



**Reference**

Loughran, T., and McDonald, B. [2011]. “When is a Liability not a Liability? Textual Analysis, Dictionaries, and 10-Ks”, SSRN, <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1331573>