Dong Liang. Predicting Stock Price Changes with Earnings Call Transcripts. A Master's Paper for the M.S. in I.S degree. April, 2016. 43 pages. Advisor: Arcot Rajasekar

This paper adopts sentiment analysis approaches to predict stock price changes of 14 major U.S. airlines, using 325 earnings call transcripts from year 2007 to 2015. We combined machine learning classification techniques with Loughran and McDonald Sentiment Word Lists (Master Dictionary), and built the Python program from scratch. Text transcripts as well as stock prices were captured online. Transcripts were labeled according to sentiment scores defined by us. After a three-way data split, all six algorithms failed to result in an ideal accuracy. The results suggest that earnings call transcripts are not informative enough to predict stock price changes.

Headings:

Sentiment Analysis

Text Mining

Machine Learning

PREDICTING STOCK PRICE CHANGES WITH EARNINGS CALL TRANSCRIPTS

by
Dong Liang

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April 2016

Approved by

Arcot Rajasekar

Table of Contents

1. Introduction

The stock market has long been a fascinating topic for people from diverse backgrounds all over the world. The prediction of stock price is also quite hot in the past years. The accuracy of the prediction varies significantly due to the complexity and variability of the stock market. In recent years, many researchers have got good results with the help of sentiment analysis techniques and social media services like Twitter (Bollen et al., 2010; Ruiz et al., 2012; Souza et al., 2015). As there already exist quite a few excellent findings on social media sources, in this paper, we switch to another data source, the earnings call transcripts, but still use the sentiment analysis techniques.

Sentiment analysis or opinion mining refers to the application of natural language processing, computational linguistics and text analytics to identify and extract subjective information in source materials[1]. It consists of four major elements: the holder, target, polarity and types of attitude[2].

In general, there are several types of sentiment analysis. A basic one is to understand whether a text represents an objective or a subjective idea. A deeper approach is based on the analysis of the polarity of the text (i.e., positive, negative or neutral). Going further, another problem is to find out the intensity (also called strength) of an emotional state underlying a text. A more complex problem is finding the exact emotions conveyed by a textual expression, such as “happiness”, “sadness”, “anger” and so on.

Finally, the most challenging problem is represented by the extraction of users' intentions, arguments and speculations (Robaldo and Caro, 2013).

Applications of sentiment analysis are quite hot in various contexts including hotels, movies, restaurants, advertising, politics and the stock market in recent years. As we all know, financial markets are considered to be complex and to be changing rapidly. For example, financial research shows that stock prices adjust quickly on new information such as dividend announcements or other company-related news (Fama, 1970). So the sentiment expressed in related financial articles or activities is of great importance as well. Different studies provide evidence that it has an impact on the following stock price reactions. For instance, the prevalence of negative sentiment can lead to a decline in stock prices (Loughran and McDonald, 2011).

Accurately identifying sentiment is not easy. This is not only because it is not unusual for humans to disagree about the sentiment of the same text, but also because of the complex ways in which humans express sentiment, using irony, sarcasm, humor and so on. What's more, the order in which different opinions are presented can result in a completely opposite overall sentiment polarity. For example, "This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it can't hold up." Interestingly, neutral comments in feedback systems are perceived by users as negative ones rather than lying at the exact mid-point between positive and negative comments. (Pang and Lee, 2008)

Since there are many outstanding studies on social media sentiment analysis, we decided to try to use another type of data source. All public companies have to hold earnings call and the transcripts are publicly accessible. The importance, length and number of those transcripts are also quite appropriate for our analysis.

Prior to our study, I did some analysis on the movie review corpus of Python NLTK package. The corpus consist of a thousand positive reviews and a thousand negative reviews. Since the reviews are quite simple and short, classification algorithms alone are sufficient for sentiment polarity prediction, yielding an average accuracy of over 80%. Inspired by previous studies and my experiments, we treat the analysis as a classification problem in the field of machine learning and take advantage of the Loughran and McDonald Sentiment Word List created from 10-K reports. We also designed a sentiment score serving as the label or target value of the transcripts. Prediction is evaluated by accuracy, which is calculated based on how well the classification algorithms predict the sentiment scores.

One of the innovation points of our study is that our data source is earnings call transcripts instead of social media posts. Although many studies have shown that social media data is sufficient enough to yield good results, we want to explore other sources that may result in meaningful outcomes in a different way.

Another innovation point is that this paper combines classification techniques with a sentiment score metric defined by us. In some prior studies, sentiment score is calculated based on occurrences of words in texts or the intersection of some lexicon and texts. In this paper, we define our own sentiment score metric according to the trends of stock prices. However, the results we got are not ideal, suggesting that earnings call transcripts alone are not informative enough for predicting stock price changes. We will continue to explore this combination in future work.

[1] https://en.wikipedia.org/wiki/Sentiment_analysis
[2] https://www.coursera.org/course/nlp

2. Research Question

Since we’ve done some research related to airlines before, and there are few studies focusing on earnings call transcripts, we form our research questions: (1) To what extent are stock price changes of major U.S. airlines associated with earnings call transcripts? (2) Can we use transcripts of an airline to predict its stock price changes?

This kind of research may provide some insights for intelligent investment. For this purpose, we study whether sentiment analysis is useful in predicting stock price changes after the release of some articles on finance. In our paper, they are the earnings call transcripts. We adopted a novel approach: combine machine learning classification techniques with a lexicon and sentiment score. First, we created a representation for each transcript in order to get rid of meaningless words. Second, we computed a sentiment score for each representation based on the trends of stock price changes within 31 days of the release date. Third, several classification algorithms are trained according to sentiment scores. Finally, we employed a three-way data split and accuracy is calculated and used to evaluate the results.

3. Literature review

3.1 Sentiment Analysis

There exist two main approaches to the problem of extracting sentiment automatically (Taboada et al., 2011): Text classification (supervised) and lexicon-based approach (unsupervised).

Pang et al. (2002) build a sentiment lexicon for movie reviews to indicate positive and negative opinion. They also experiment with n-grams and three machine learning algorithms: Naive Bayes classification, maximum entropy classification, and support vector machines, though the most successful features seem to be basic unigrams with an accuracy of 82.9%. Given the results, we mainly focus on study of unigrams and their cleaning.

Pang & Lee (2008) provide an excellent recent survey of opinion mining or sentiment analysis problems and approaches used to tackle them. They mention that building a lexicon takes a great amount of time because researches are required in order to determine which words have strong enough sentiment consistently across domains to be included and care must be given to maintain the list so that it excludes superfluous terms. This leads to another issue of sentiment analysis – domain context, rather than just textual context. Since we are unable to afford to build our own lexicon, we take advantage of an existing lexicon generated by 10-K reports.

Turney (2002) takes a lexicon-based approach, and PMI-IR (Pointwise Mutual Information and Information Retrieval) algorithm is employed to estimate semantic orientation. Lin et al. (2014) construct a microblog-oriented sentiment lexicon using the Semantic Orientation from Pointwise Mutual Information (SO-PMI). These papers remind us of the importance of feature extraction and elimination of low information features.

The majority of the statistical text classification researches build Support Vector Machine classifiers, trained on a particular data set using features such as unigrams or bigrams, and with or without part-of-speech labels (Pang, Lee and Vaithyanathan, 2002; Salvetti, Reichenbach and Lewis, 2006).

3.2 Financial Documents and Stock Market

A supervised approach is conducted by Antweiler & Frank (2004) who collect messages posted on two finance message boards. They manually determine the sentiment of a sub-sample of 1,000 messages and use these messages to train a classifier. Then this classifier is used to assess the sentiment of the remaining messages. In this paper, we defined our own sentiment score metric derived from 8 kinds of stock price changes.

Tetlock (2007) analyzes the sentiment of a daily wall street journal column. He uses the General Inquirer’s Harvard-IV-4 classification dictionary to classify each word of the column according to its sentiment. Afterwards, he uses these classified words to calculate a pessimism factor.

Loughran & McDonald (2011) evaluate the sentiment of 10-K company reports. They develop different word lists containing positive and negative terms. They calculate the number of negative words per report to determine a negativity measure. This dictionary, which is also called Master Dictionary, is well-built and extremely useful to our study. We chose this lexicon for creating the representations of transcripts.

Siering (2012) adopts Support Vector Machine for the prediction of stock price reactions following a financial news message. He uses both Harvard-IV-4 lexicon and FIN provided by (Loughran & McDonald, 2011). Every document is represented by top 500 features in terms of corresponding information gain. Moreover, his document-level sentiment measure is an inspiring idea to our study. We computed a sentiment score for the representation of each transcript based on the intersection of the texts and the Master Dictionary.

3.3 Lexicons

Several lexicons are publicly available online such as Hu and Liu's Opinion Lexicon, MPQA Opinion Corpus, General Inquirer, SentiWordNet, LIWC and Loughran and McDonald Sentiment Word Lists.

The Hu and Liu’s Opinion Lexicon is compiled over many years starting from Professor Bing Liu and Dr. Minqing Hu’s first paper at University of Illinois at Chicago. It consists of about 6800 positive and negative English sentiment words[3].

The MPQA Opinion Corpus contains news articles from a wide variety of news sources manually annotated for opinions and other private states (i.e., beliefs, emotions, sentiments, speculations, etc.)[4].

The General Inquirer contains several dictionaries: (1) the Harvard IV-4 dictionary, (2) the Lasswell value dictionary, (3) several categories recently constructed, and (4) "marker" categories primarily developed as a resource for disambiguation. It also has a positive word list with 1915 words and a negative word list with 2291 words. The category and strength of words are labelled as well[5].

SentiWordNet is a lexical resource for opinion mining. It assigns to each synset of WordNet (Esuli & Sebastiani, 2006, p. 417) three sentiment scores: positivity, negativity, objectivity. A synset is a set of terms that are held together by a common definition. The current version of SentiWordNet is 3.0 (Baccianella et al., 2010)[6].

LIWC stands for Linguistic Inquiry and Word Count. It's a program that reads a given text and counts the percentage of words that reflect different emotions, thinking styles, social concerns, and even parts of speech. LIWC was developed by researchers with interests in social, clinical, health, and cognitive psychology, the language categories were created to capture people's social and psychological states. The latest LIWC2015 Dictionary is composed of almost 6,400 words, word stems, and select emoticons[7].

Loughran and McDonald Sentiment Word Lists (Master Dictionary) has about 85,131 words. It is created by Professor Tim Loughran and Bill McDonald at University of Notre Dame. The dictionary includes statistics for word frequencies in all 10-K documents from 1994-2014 (including 10-X variants). It reports counts, proportion of total, average proportion per document, standard deviation of proportion per document, document count (i.e., number of documents containing at least one occurrence of the word), nine sentiment category identifiers (e.g., negative, positive, uncertainty, litigious, modal, constraining), Harvard Word List identifier, number of syllables, and source for each word[8].

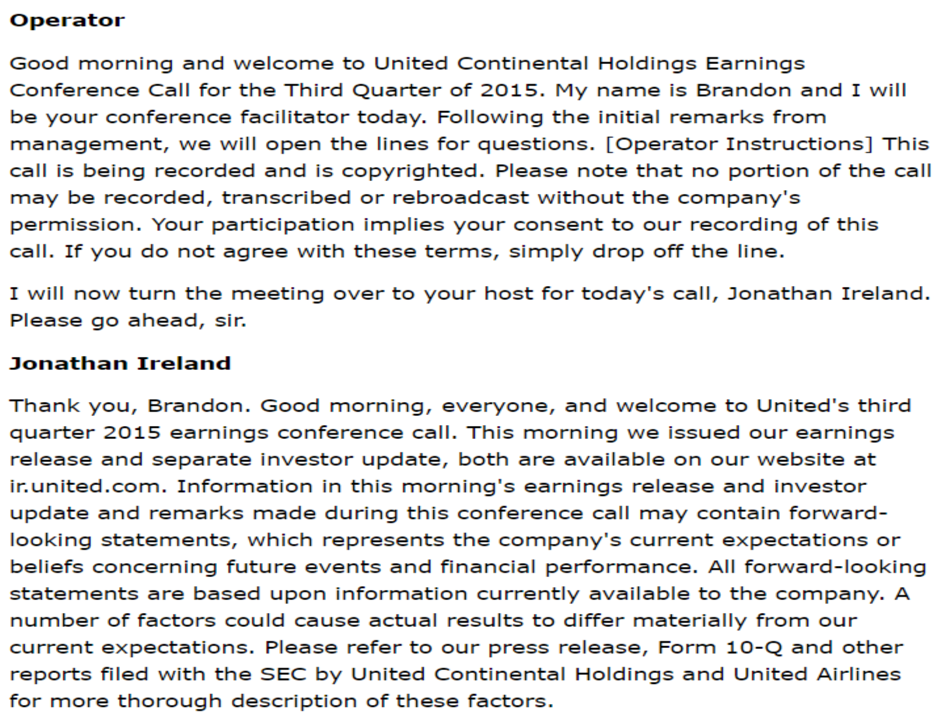
Besides lexicons of general sentiment analysis purposes, some lexicons have lists of different categories. For example, 570 words in the General Inquirer are of an economic orientation. We selected Master Dictionary as the lexicon for our analysis since it's well tailored to financial domain and has both positive and negative terms.

[3] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
[4] http://mpqa.cs.pitt.edu
[5] http://www.wjh.harvard.edu/~inquirer/homecat.htm
[6] http://sentiwordnet.isti.cnr.it
[7] http://liwc.wpengine.com
[8] http://www3.nd.edu/~mcdonald/Word_Lists.html

4. Earnings Call Transcripts

An earnings call is a teleconference, or increasingly a webcast, in which a public company discusses the financial results of a reporting period via an 800 number and on the internet. The name comes from earnings per share (EPS), the bottom line number in the income statement divided by the number of shares outstanding[9].

Holding earnings calls once a quarter is a requirement for all public companies. The transcripts will be available online shortly after the event. Attendees include executives of a company as well as some analysts. The management team would first talk about their financial performance over the last quarter, followed by a Q&A section with the analysts. Sometimes, future plans would also be revealed during an earnings call.



Operator

Good morning and welcome to United Continental Holdings Earnings Conference Call for the Third Quarter of 2015. My name is Brandon and I will be your conference facilitator today. Following the initial remarks from management, we will open the lines for questions. [Operator Instructions] This call is being recorded and is copyrighted. Please note that no portion of the call may be recorded, transcribed or rebroadcast without the company's permission. Your participation implies your consent to our recording of this call. If you do not agree with these terms, simply drop off the line.

I will now turn the meeting over to your host for today's call, Jonathan Ireland. Please go ahead, sir.

Jonathan Ireland

Thank you, Brandon. Good morning, everyone, and welcome to United's third quarter 2015 earnings conference call. This morning we issued our earnings release and separate investor update, both are available on our website at ir.united.com. Information in this morning's earnings release and investor update and remarks made during this conference call may contain forward-looking statements, which represents the company's current expectations or beliefs concerning future events and financial performance. All forward-looking statements are based upon information currently available to the company. A number of factors could cause actual results to differ materially from our current expectations. Please refer to our press release, Form 10-Q and other reports filed with the SEC by United Continental Holdings and United Airlines for more thorough description of these factors.

Figure 1. Earnings call transcript snippet

The transcripts are accessible for free on websites such as Seeking Alpha10.

In this paper, we collected transcripts of 14 major U.S. airlines from year 2007 to 2015. Among these quarters, some transcripts are missing or some stock prices are unavailable due to merger. So we got 325 transcripts in total and the average size is about 49 kilobytes.

[9] https://en.wikipedia.org/wiki/Earnings_call
[10] http://seekingalpha.com/earnings/earnings-call-transcripts

5. Methodology

The goal of our study is to find out the relationship between stock price changes and the sentiments conveyed in earnings call transcripts, and potential predictability of the changes.

Our research questions are:

(1) To what extent are stock price changes of major U.S. airlines associated with earnings call transcripts?

(2) Can we use transcripts of an airline to predict its stock price changes?

5.1 Overview

In order to achieve our goal, we adopted a novel approach: combining machine learning techniques with a sentiment lexicon, and using sentiment score as the indicator of stock price changes. Our program is written in Python from scratch. In short, there are four steps. Firstly, since on average, each transcript has around 10,000 words, which includes a lot of meaningless words. To eliminate those low information words, in addition to stopwords removal, we compare every transcript with the Master Dictionary and take the intersection as the representation of that transcript, assuming words appear in the lexicon are more informative than those do not. In this way, a representation may only contain tens or hundreds of "good" words. Secondly, after getting stock prices, we compute 8 types of sentiment score for each representation of transcript according to different stock price changes. This price serves as the label or target value of every transcript. Thirdly, six machine learning classifiers are trained and tested via a three-way data split. The proportion of training set to test set and nested training set to validation set are both 8:2. Finally, accuracy is calculated and used to evaluate the results. Codes are in Appendix A.

5.2 Data Collection

5.2.1 Earnings Call Transcripts

The data for our study is earnings call transcripts of 14 major airlines of United States: American Airlines, Atlas Air, Allegiant Air, Alaska Airlines, Delta Air Lines, Hawaiian Airlines, JetBlue Airways, US Airways, Southwest Airlines, Republic Airlines, Spirit Airlines, SkyWest Airlines, United Airlines and UPS Airlines. US Airways merged with American Airlines but have transcripts from year 2007 to 2013. UPS is the world's largest package delivery company and it has a large fleet so it's on our list.

We can have access to the texts online on Seeking Alpha. As there are so many transcripts, getting them manually is very inefficient. With the help of Python library Requests and Beautiful Soup, and regular expressions, we captured all transcripts automatically. The release dates were also recorded in order to get stock price afterwards. Texts are saved as .doc files in UTF-8 encoding. The folder names are stock symbols. The file name is made up of quarter, stock symbol or airline name and release date. For example: "2014 Q2 American Airlines-Jul 24, 2014.doc". Transcripts are stored locally in a hard drive instead of a database since they are not complex in terms of variety.

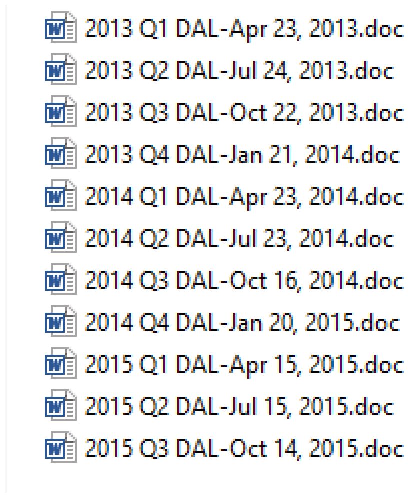
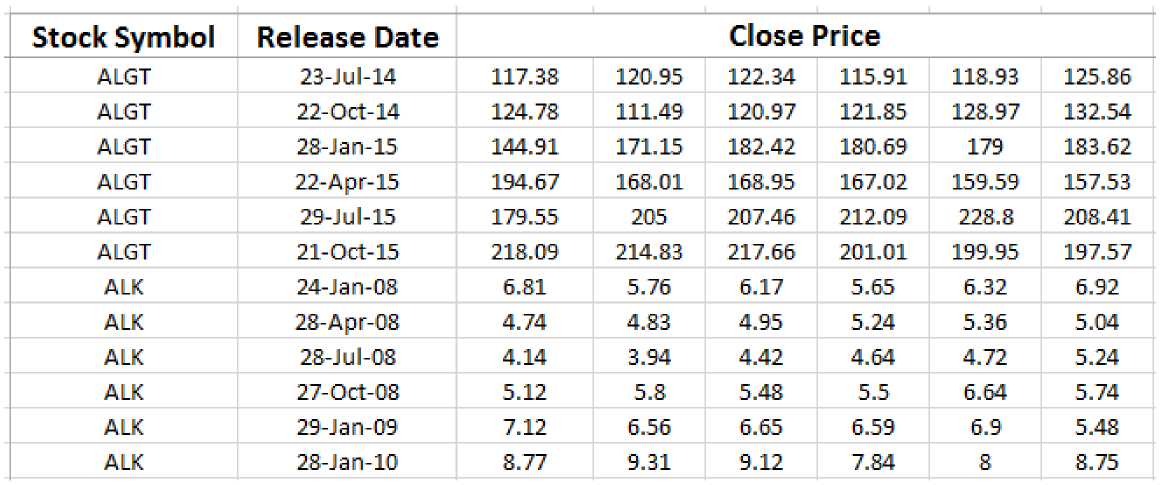

2013 Q1 DAL-Apr 23, 2013.doc
2013 Q2 DAL-Jul 24, 2013.doc
2013 Q3 DAL-Oct 22, 2013.doc
2013 Q4 DAL-Jan 21, 2014.doc
2014 Q1 DAL-Apr 23, 2014.doc
2014 Q2 DAL-Jul 23, 2014.doc
2014 Q3 DAL-Oct 16, 2014.doc
2014 Q4 DAL-Jan 20, 2015.doc
2015 Q1 DAL-Apr 15, 2015.doc
2015 Q2 DAL-Jul 15, 2015.doc
2015 Q3 DAL-Oct 14, 2015.doc

Figure 2. Examples of transcript files

In some cases, the websites would block our program because it detected unusual large amount of data flow. One workaround is to fake an operating system and browser header to make the websites believe that our program is a person surfing the Internet.

5.2.2 Stock Prices

Historical stock prices are available on websites such as Google Finance[11] and Yahoo Finance[12]. For this study, we chose Google Finance as our data source. To reflect stock price changes, we collected close prices on 6 dates related to the release date of a transcript: 31 days before and after the date, 7 days before and after the date and 1 day before and after the date. If a price is missing (i.e., that day is a holiday or weekend), the program would iteratively check for the previous or next available price, making sure that every transcript is associated with 6 close prices. Table 1 shows a part of transcripts and close prices.



Stock Symbol	Release Date	Close Price					
ALGT	23-Jul-14	117.38	120.95	122.34	115.91	118.93	125.86
ALGT	22-Oct-14	124.78	111.49	120.97	121.85	128.97	132.54
ALGT	28-Jan-15	144.91	171.15	182.42	180.69	179	183.62
ALGT	22-Apr-15	194.67	168.01	168.95	167.02	159.59	157.53
ALGT	29-Jul-15	179.55	205	207.46	212.09	228.8	208.41
ALGT	21-Oct-15	218.09	214.83	217.66	201.01	199.95	197.57
ALK	24-Jan-08	6.81	5.76	6.17	5.65	6.32	6.92
ALK	28-Apr-08	4.74	4.83	4.95	5.24	5.36	5.04
ALK	28-Jul-08	4.14	3.94	4.42	4.64	4.72	5.24
ALK	27-Oct-08	5.12	5.8	5.48	5.5	6.64	5.74
ALK	29-Jan-09	7.12	6.56	6.65	6.59	6.9	5.48
ALK	28-Jan-10	8.77	9.31	9.12	7.84	8	8.75

Table 1. Stock symbol, release date and close price

Table 2 shows the 14 airlines and their stock symbols in NASDAQ. If an airline does not have a stock symbol, we use the stock symbol of its parent organization.

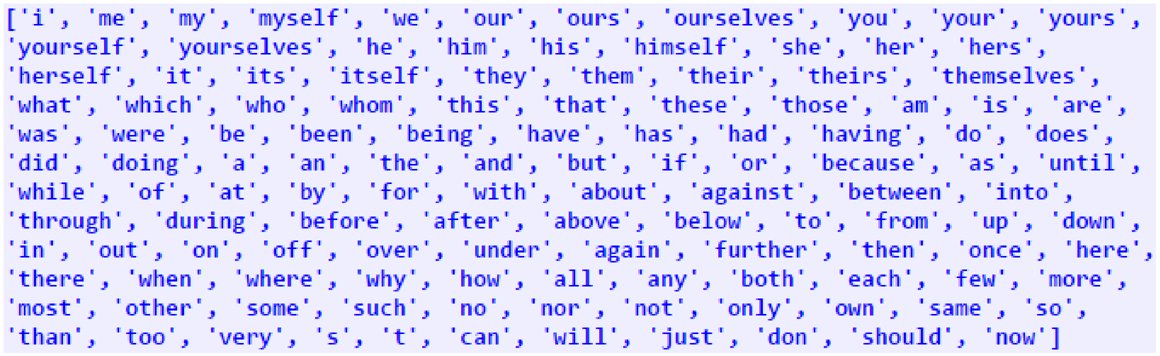
Airline Name	Stock Symbol
American Airlines	AAR / AAL
Atlas Air	AAWW
Allegiant Air	ALGT
Alaska Airlines	ALK
Delta Air Lines	DAL
Hawaiian Airlines	HA
JetBlue Airways	JBLU
US Airways	LCC
Southwest Airlines	LUV
Republic Airlines	RJET
Spirit Airlines	SAVE
SkyWest Airlines	SKYW
United Airlines	UAL
UPS Airlines	UPS

Table 2. Airline names and stock symbols

5.3 Data Preprocessing

5.3.1 Data Cleaning

The earnings call transcripts we captured from Seeking Alpha are quite noisy. Each transcript contains about 10,000 words, among which there are a large number of meaningless words. So the first step of data preprocessing is to remove the stopwords. Stopwords are words that are generally considered useless since they are so common in the corpus that including them would greatly increase computational complexity without improving accuracy. The Python NLTK package has a stopword list that includes a list of 127 English stopwords, which is the basis of our customized list. Whereas, these words are far from enough, we manually added more words to the list according to the characteristics of our corpus. For example, airline names, weekday names, English numbers and so on.



```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']
```

Figure 3. NLTK stopword list

As there are a lot of proper nouns in the transcripts such as human names, city names and organization names, which are also considered as stopwords, we created a list for the removal of these words. For example, “Jerry”, “Alexandre”, “Vancouver”, “Goldman Sachs”, etc.

In addition, regarding our transcripts, airport names, country names and U.S. city names are also meaningless. So we constructed 3 lists called “airport_names.txt”, “country_names.txt” and “US_city_names.txt” for the elimination of these words.

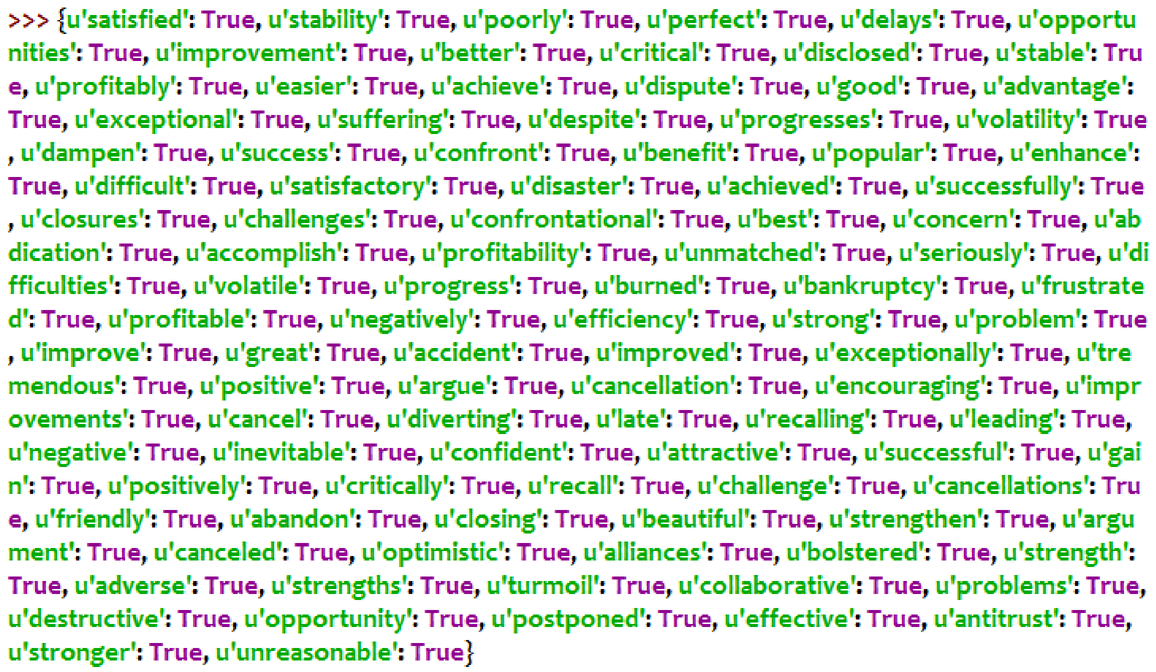
After removing those stopwords, regular expressions are employed to remove characters that are not English words, and abbreviations, symbols and digits. If the length of a word is less than or equal to 3, we removed it as well because this kind of word tends to contain less information.

The “clean” texts are saved as a .pickle file. Pickling is used for serializing and de-serializing a Python object structure so that an object can be saved and loaded for future usage.

5.3.2 Transcript Representations

The texts are relatively clean after stopword removal, but they are not clean enough and not that informative. In order to clean the texts more thoroughly, reduce computational complexity and improve the accuracy of prediction, we created a representation for each transcript, that is, took the intersection of a transcript and the Master Dictionary. If a word appears both in the transcript and the lexicon, it’s added to the representation. In this way, we significantly reduced the number of words in each transcript and at the same time, kept the informative ones in the representation. We only need to handle tens or hundreds of words per representation rather than 10,000 words per transcript. These words are considered to be rich in information in determining the sentiment of a transcript as well as stock price changes.

Every representation is a Python dictionary and each element in the dictionary is formatted as “word: True”, meaning it contains this word. Figure 4 shows an example of the representation. Letter ‘u’ indicates Unicode encoding.



```
>>> {u'satisfied': True, u'stability': True, u'poorly': True, u'perfect': True, u'delays': True, u'opportu
nities': True, u'improvement': True, u'better': True, u'critical': True, u'disclosed': True, u'stable': Tru
e, u'profitably': True, u'easier': True, u'achieve': True, u'dispute': True, u'good': True, u'advantage':
True, u'exceptional': True, u'suffering': True, u'despite': True, u'progresses': True, u'volatility': True
, u'dampen': True, u'success': True, u'confront': True, u'benefit': True, u'popular': True, u'enhance':
True, u'difficult': True, u'satisfactory': True, u'disaster': True, u'achieved': True, u'successfully': True
, u'closures': True, u'challenges': True, u'confrontational': True, u'best': True, u'concern': True, u'ab
dication': True, u'accomplish': True, u'profitability': True, u'unmatched': True, u'seriously': True, u'di
fficulties': True, u'volatile': True, u'progress': True, u'burned': True, u'bankruptcy': True, u'frustrate
d': True, u'profitable': True, u'negatively': True, u'efficiency': True, u'strong': True, u'problem': True
, u'improve': True, u'great': True, u'accident': True, u'improved': True, u'exceptionally': True, u'tre
mendous': True, u'positive': True, u'argue': True, u'cancellation': True, u'encouraging': True, u'impr
ovements': True, u'cancel': True, u'diverting': True, u'late': True, u'recalling': True, u'leading': True,
u'negative': True, u'inevitable': True, u'confident': True, u'attractive': True, u'successful': True, u'gai
n': True, u'positively': True, u'critically': True, u'recall': True, u'challenge': True, u'cancellations': Tru
e, u'friendly': True, u'abandon': True, u'closing': True, u'beautiful': True, u'strengthen': True, u'argu
ment': True, u'canceled': True, u'optimistic': True, u'alliances': True, u'bolstered': True, u'strength':
True, u'adverse': True, u'strengths': True, u'turmoil': True, u'collaborative': True, u'problems': True,
u'destructive': True, u'opportunity': True, u'postponed': True, u'effective': True, u'antitrust': True,
u'stronger': True, u'unreasonable': True}
```

Figure 4. An example of the representation of a transcript

5.4 Data Analysis

5.4.1 Sentiment Score

Inspired by Siering (2012), we developed our own measurement called sentiment score, which is the label for every representation to reflect the fluctuation of stock prices. As each representation is associated with 6 close prices, we define them as P_{-31}, P_{-7}, P_{-1}, P_{+1}, P_{+7} and P_{+31}.

Stock Price					
-31 days	-7 days	-1 day	+1 day	+7 day	+31 day
P_{-31}	P_{-7}	P_{-1}	P_{+1}	P_{+7}	P_{+31}

Table 3. Six stock prices

Accordingly, there are 3 changes:

$$\mathbf{V_1 = P_{+31} - P_{-31}}$$

$$\mathbf{V_2 = P_{+7} - P_{-7}}$$

$$\mathbf{V_3 = P_{+1} - P_{-1}}$$

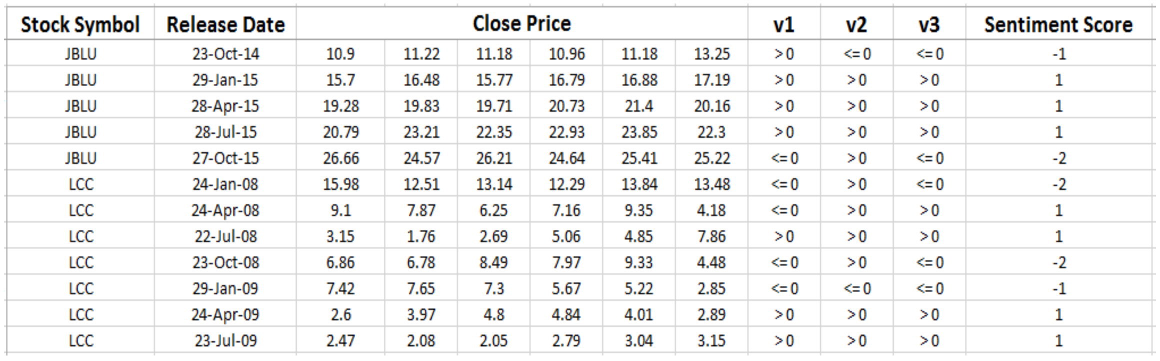
The idea behind using 6 prices and 3 changes is that we want to have more comprehensive measurement of the stock price changes and monitor whether the changes are caused mainly by the transcripts or other factors.

Sentiment score is defined in Table 4.

V_1	V_2	V_3	Sentiment Score
> 0	> 0	> 0	1
> 0	> 0	≤ 0	-3
> 0	≤ 0	> 0	2
> 0	≤ 0	≤ 0	-1
≤ 0	> 0	> 0	1
≤ 0	> 0	≤ 0	-2
≤ 0	≤ 0	> 0	3
≤ 0	≤ 0	≤ 0	-1

Table 4. Calculation of sentiment score

A sentiment score is assigned to every representation and is set as the label. It's also added to the Python dictionary of representations. The structure is: ({dictionary}, sentiment score).



Stock Symbol	Release Date	Close Price						v1	v2	v3	Sentiment Score
JBLU	23-Oct-14	10.9	11.22	11.18	10.96	11.18	13.25	>0	<=0	<=0	-1
JBLU	29-Jan-15	15.7	16.48	15.77	16.79	16.88	17.19	>0	>0	>0	1
JBLU	28-Apr-15	19.28	19.83	19.71	20.73	21.4	20.16	>0	>0	>0	1
JBLU	28-Jul-15	20.79	23.21	22.35	22.93	23.85	22.3	>0	>0	>0	1
JBLU	27-Oct-15	26.66	24.57	26.21	24.64	25.41	25.22	<=0	>0	<=0	-2
LCC	24-Jan-08	15.98	12.51	13.14	12.29	13.84	13.48	<=0	>0	<=0	-2
LCC	24-Apr-08	9.1	7.87	6.25	7.16	9.35	4.18	<=0	>0	>0	1
LCC	22-Jul-08	3.15	1.76	2.69	5.06	4.85	7.86	>0	>0	>0	1
LCC	23-Oct-08	6.86	6.78	8.49	7.97	9.33	4.48	<=0	>0	<=0	-2
LCC	29-Jan-09	7.42	7.65	7.3	5.67	5.22	2.85	<=0	<=0	<=0	-1
LCC	24-Apr-09	2.6	3.97	4.8	4.84	4.01	2.89	>0	>0	>0	1
LCC	23-Jul-09	2.47	2.08	2.05	2.79	3.04	3.15	>0	>0	>0	1

Table 5. Sentiment score and representation of transcript

5.4.2 Three-Way Data Split

In order to get more accurate, not biased and better results, we performed a three-way data split on the corpus. We divided all the representations into 3 sets: training set, validation set and test set. The training set is used for fitting a classifier, the validation set is used to tune the classifier, and the test set is used only for assessing the performance of a trained classifier.

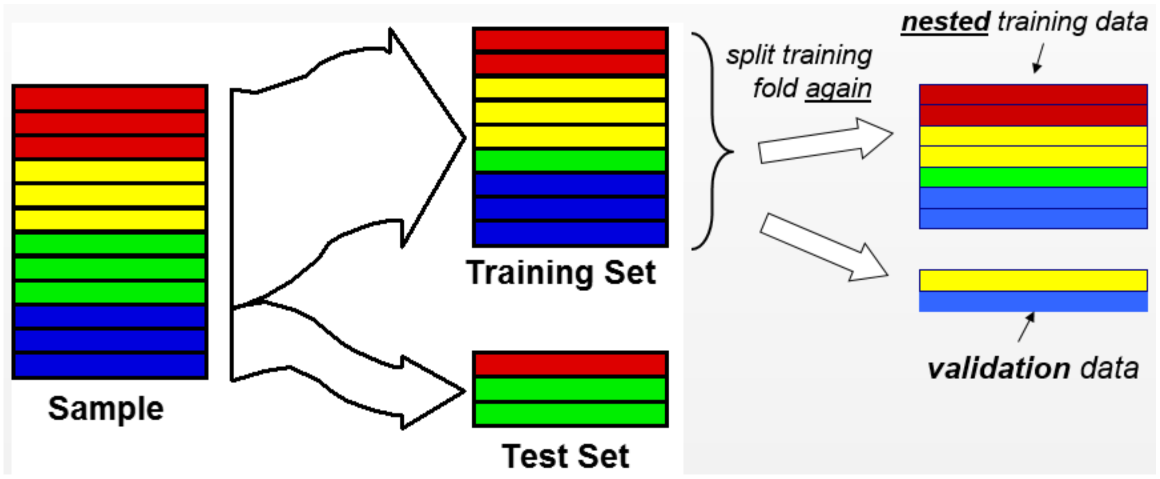


Figure 5. Three-way data split

We held out 20% of our data as the test set and in the remaining 80% of data, 20% is used as the validation set. So the size of training set to validation set to test set equals 64: 16: 20.

5.4.3 Classifiers

We treat our analysis as a classification problem so appropriate classifiers are needed. Naïve Bayes, linear models, SVM are commonly used in classification.

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naïve" assumption of independence between every pair of features. The different naive Bayes classifiers differ mainly by the assumptions they make. They are simple to implement and efficient.

Logistic Regression is generally used for predicting a categorical outcome. SGD estimator implements regularized linear models with stochastic gradient descent learning.

Given our problem, we picked six classifiers for our study: Naïve Bayes, Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Logistic Regression, Stochastic Gradient Descent (SGD) and Linear Support Vector. These classifiers are available in Python scikit-learn package[13].

5.4.4 Cross Validation

We followed the idea of k-fold cross validation. Since we took 20% of the data as the test set, it's a 5-fold cross validation. For each experiment:

(1) Shuffle the whole data set (representations and sentiment scores)

(2) Divide the whole set into training set, validation set and test set

(3) Train the six classifiers using the training set

(4) Evaluate the classifiers using the validation set

(5) Perform 5-fold cross validation on the training set and validation set

(6) Select the model that yields highest accuracy

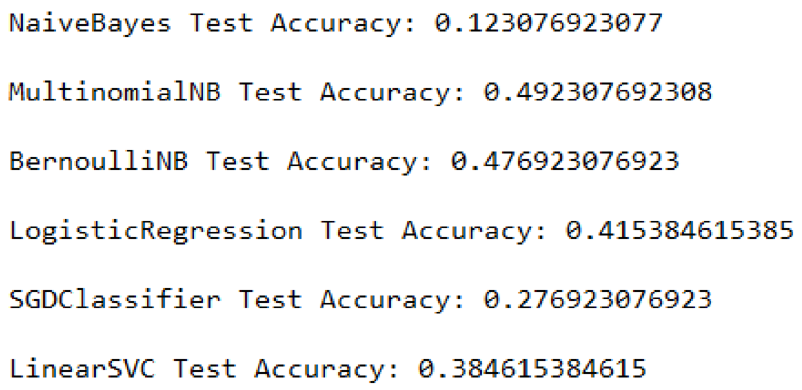
(7) Assess this model using the test set

The definition of accuracy is the percentage of correct prediction of sentiment score made by a classifier. As different sentiment scores indicate different types of stock price changes, accuracy is the key metric in determining to what extent are stock price changes associated with earnings call transcripts and whether can we use them to predict stock price changes.

[11] https://www.google.com/finance
[12] http://finance.yahoo.com
[13] http://scikit-learn.org

6. Results and Discussion

Results of our experiments were written to a text file called “Accuracy.txt”. All six classifiers failed to get an accuracy over 50%, which was not what we expected. After running experiments for more times, the best accuracy we could get was 49% achieved by the Multinomial Naïve Bayes classifier. The Bernoulli Naïve Bayes classifier got a similar accuracy and the Naïve Bayes did the worst. The output file is in Appendix B.



```
NaiveBayes Test Accuracy: 0.123076923077

MultinomialNB Test Accuracy: 0.492307692308

BernoulliNB Test Accuracy: 0.476923076923

LogisticRegression Test Accuracy: 0.415384615385

SGDClassifier Test Accuracy: 0.276923076923

LinearSVC Test Accuracy: 0.384615384615
```

Figure 6. Best accuracy

The failure may be caused by multiple factors since prediction of stock price changes is very complicated. The low accuracy may due to inappropriate data source, representation of texts or inaccurate models, etc.

Earnings call transcripts are relatively less used in predicting stock price changes, though it’s a “new” perspective of solving the problem of prediction, transcripts alone do not suffice to affect the stock prices.

In addition, problems may lie in data processing. Due to limitations, it's likely that we got rid of words rich in sentiment but kept some useless words. Moreover, building representation of transcripts is also very important yet difficult. A lexicon that is manually created and customized is key to the success of filtering useful words. Polarity as well as sentiment strength in different context must be taken into account. Text mining techniques such as stemming and bi-grams may also be beneficial in some circumstances.

Last but not least, classifiers also have much room for improvement. The training and test strategy, selection of algorithms and tuning of parameters all contribute to an ideal accuracy.

Nevertheless, the results suggest that earnings call transcripts alone are not informative enough to predict stock price changes. There may be little causation between the two. Stock market is way too complex and numerous factors need to be considered when predicting stock prices. Admittedly, earnings call transcripts can only account for little part of it. We can't solely rely on these transcripts when making predictions.

7. Conclusion

We explored whether we can predict stock price changes of 14 major U.S. airlines using their earnings call transcripts. We combined text mining techniques and a sentiment lexicon (Master Dictionary) with machine learning classification, and defined a sentiment score as the label of this supervised learning approach. A three-way data split was also introduced. Then we took advantage of six classifiers and run 5-fold cross validation. We wrote our program in Python from scratch instead of using other software.

The accuracy of our prediction was not ideal at all. None of the classifiers reached more than 50%, which is like flipping a coin. The results suggest that earning call transcripts have little influence on the stock price changes and they are not suitable for predicting the stock market.

Future improvement includes:

- Variety of data sources. Stock prices are too complicated to be analyzed based on few kinds of data sources.
- Text mining techniques. Text preprocessing and analysis skills need further exploration.
- A tailored Lexicon. A more domain-specific and comprehensive lexicon that is created manually is ideal for sentiment analysis.
- Choice of label. Sentiment score is a good concept but the definition and formula may need modification.

- Model selection and tuning. More classifiers and parameters should be considered.

We still have a long way to go in predicting stock price changes as the stock market is changing every millisecond and we don't have access to all the information, and even we do, we can't afford to process it. Although our study shows negative results, we should keep exploring new sources and methods towards understanding or even solving this mystery problem.

With the rising of machine learning techniques, sentiment analysis will be hotter and hotter in the following years. It will definitely play a vital role in every part of business and making the world a better place.

References

Antweiler, Werner, and Murray Z. Frank. "Is all that talk just noise? The information content of internet stock message boards." *The Journal of Finance* 59.3 (2004): 1259-1294.

Baccianella, Stefano, Andrea Esuli, and Fabrizio Sebastiani. "SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining." *LREC.* Vol. 10. 2010.

Bollen, Johan, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market." *Journal of Computational Science* 2.1 (2011): 1-8.

Dasgupta, Sajib, and Vincent Ng. "Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification." *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2.* Association for Computational Linguistics, 2009.

Esuli, Andrea, and Fabrizio Sebastiani. "Sentiwordnet: A publicly available lexical resource for opinion mining." *Proceedings of LREC.* Vol. 6. 2006.

Fama, Eugene F. "Efficient capital markets: A review of theory and empirical work*." *The journal of Finance* 25.2 (1970): 383-417.

Kaur, Amandeep, and Vishal Gupta. "A survey on sentiment analysis and opinion mining techniques." *Journal of Emerging Technologies in Web Intelligence* 5.4 (2013): 367-371.

Lin, Lu, et al. "Opinion Mining and Sentiment Analysis in Social Networks: A Retweeting Structure-Aware Approach." *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on.* IEEE, 2014.

Loughran, Tim, and Bill McDonald. "When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks." *The Journal of Finance* 66.1 (2011): 35-65.

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques." *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10.* Association for Computational Linguistics, 2002.

Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." *Foundations and trends in information retrieval* 2.1-2 (2008): 1-135.

Robaldo, Livio, and Luigi Di Caro. "Opinionmining-ml." *Computer Standards & Interfaces* 35.5 (2013): 454-469.

Ruiz, Eduardo J., et al. "Correlating financial time series with micro-blogging activity." *Proceedings of the fifth ACM international conference on Web search and data mining.* ACM, 2012.

Siering, Michael. "" Boom" or" Ruin"--Does It Make a Difference? Using Text Mining and Sentiment Analysis to Support Intraday Investment Decisions." *System Science (HICSS), 2012 45th Hawaii International Conference on.* IEEE, 2012.

Souza, Thársis Tuani Pinto, et al. "Twitter sentiment analysis applied to finance: A case study in the retail industry." *arXiv preprint arXiv:1507.00784* (2015).

Taboada, Maite, et al. "Lexicon-based methods for sentiment analysis." *Computational linguistics* 37.2 (2011): 267-307.

Tetlock, Paul C. "Giving content to investor sentiment: The role of media in the stock market." *The Journal of Finance* 62.3 (2007): 1139-1168.

Turney, Peter D. "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews." *Proceedings of the 40th annual meeting on association for computational linguistics.* Association for Computational Linguistics, 2002.

Appendix A: Python Codes

Get_Stock_Price.py

```
import requests, glob, time, pickle
from bs4 import BeautifulSoup
from datetime import datetime, timedelta
import requests.packages.urllib3
requests.packages.urllib3.disable_warnings()
import csv

# get html files
def get_soup(params):
    headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95
Safari/537.36'}
    r = requests.get("https://www.google.com/finance/historical", params=params,
headers=headers)
    soup = BeautifulSoup(r.text, 'html.parser')
    return soup

# get file paths
def file_paths():
    return glob.glob('Domestic/*/*')

# get release dates of transcripts
def get_dates(filepaths):
    dates = []
    for path in filepaths:
        date = path.split('-')[1]
        date = date.replace('.doc', '')
        dates.append(date)
    return dates

# get stock symbols
def get_symbols(filepaths):
    symbols = []
    for i in range(27):
        symbols.append('AAR')
    for path in filepaths[27:]:
        symbol = path.split('\\')[1]
        symbols.append(symbol)
    return symbols

# convert weekends to Fridays
def tweak_date(date):
    interval = timedelta(days=1)
    if date.weekday() == 5:
        date -= interval
    elif date.weekday() == 6:
        date -= 2 * interval
    return date

# get the date after release date
def get_six_dates(symbol, date):
    intervals = []
    intervals.append(timedelta(days=-31))
    intervals.append(timedelta(days=-7))
    intervals.append(timedelta(days=-1))
    intervals.append(timedelta(days=1))
    intervals.append(timedelta(days=7))
    intervals.append(timedelta(days=31))

    date = datetime.strptime(date, '%b %d, %Y')
    six_dates = []
    for interval in intervals:
        new_date = date + interval
        new_date = tweak_date(new_date)
        new_date = new_date.strftime('%Y %b %d')
        six_dates.append(new_date)

    # deal with weekends
    #if date_after.weekday() == 5:
    #   date_after += 2 * interval
    #date_after = date_after.strftime('%Y %b %d')
    return six_dates

# get price data
def get_data(symbol, date):
    data = []
    # parameters of URL
    params = {'q': symbol, 'startdate': date, 'enddate': date}
    soup = get_soup(params)
    # get a line of prices
    for s in soup('td'):
        try:
            if s.get('class')[0] == 'rgt':
                s = str(s.text)
                data = s.split('\n')
                break
        except: continue
    if data == []:
        data = ['-'] * 4
        return data
    for d in data:
        if d == '':
            data.remove(d)
    # Open, High, Low, Close, Volume
    if len(data) == 5:
        data.pop()
    return data

def get_stock_price(symbols, dates):
    if not len(dates) == len(symbols):
        print 'ERROR'
    prices = []
    for i in range(len(symbols)):
        six_dates = get_six_dates(symbols[i], dates[i])
        close_prices = []
        # get the close prices
        for date in six_dates:
            close_prices.append(get_data(symbols[i], date)[3])
        # symbol + prices
        prices.append((symbols[i], dates[i], close_prices))
    return prices
    #prices = collections.OrderedDict(sorted(prices.items()))

def write_csv(prices):
    global symbols, dates
    # if 'w' mode, every line follows by an extra new line
    with open('Stock_Prices.csv', 'wb') as f:
        w = csv.writer(f)
        w.writerow(['Stock Symbol', 'Release Date', 'Close Price'])
        for p in prices:
            w.writerow([p[0], p[1], p[2][0], p[2][1], p[2][2], p[2][3], p[2][4], p[2][5]])

print '--starting...--'
start = time.time()
dates = get_dates(file_paths())
symbols = get_symbols(file_paths())
prices = get_stock_price(symbols, dates)
```

```
write_csv(prices)

save = open("Pickled/stock_price.pickle","wb")
pickle.dump(prices, save)
save.close()
print '--stock price saved--'
print "run time:", time.time() - start
```

Get_Missing_Stock_Price.py

```
import requests, time, pickle
from bs4 import BeautifulSoup
from datetime import datetime, timedelta
import requests.packages.urllib3
requests.packages.urllib3.disable_warnings()
import csv

def get_soup(params):
    headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95
Safari/537.36'}
    r = requests.get("https://www.google.com/finance/historical", params=params,
headers=headers)
    soup = BeautifulSoup(r.text, 'html.parser')
    return soup

def get_price(symbol, date, index):
    price = get_data(symbol, date)[3]
    if index < 3:
            interval = timedelta(days=-1)
    else: interval = timedelta(days=1)
    counter = 1
    # keep checking the price and date for at most 10 times
    while price == '-' and counter <= 10:
        date = date + interval
        price = get_data(symbol, date)[3]
        counter += 1
    return price

def get_data(symbol, date):
    data = []
    date = date.strftime('%Y %b %d')
    # parameters of URL
    params = {'q': symbol, 'startdate': date, 'enddate': date}
    soup = get_soup(params)
    # get a line of prices
    for s in soup('td'):
        try:
            if s.get('class')[0] == 'rgt':
                s = str(s.text)
                data = s.split('\n')
                break
        except: continue
    if data == []:
        data = ['-'] * 4
        return data
    for d in data:
        if d == '':
            data.remove(d)
    # Open, High, Low, Close, Volume
    if len(data) == 5:
        data.pop()
    return data

def write_csv(full_stock_prices):
    with open('Full_Stock_Prices.csv', 'wb') as f:
        w = csv.writer(f)
        w.writerow(['Stock Symbol', 'Release Date', 'Close Price'])
        for p in full_stock_prices:
            w.writerow([p[0], p[1], p[2][0], p[2][1], p[2][2], p[2][3], p[2][4], p[2][5]])


start = time.time()
with open("Pickled/stock_price.pickle") as prices:
        stock_prices = pickle.load(prices)
# The price of the date is already missing, so check the day before or after
table = [-32, -8, -2, 2, 8, 32]

full_stock_prices = []
for element in stock_prices:
    index = 0
    prices = []
    for p in element[2]:
        if p == '-':
            date = element[1]
            date = datetime.strptime(date, '%b %d, %Y')
            date = date + timedelta(days=table[index])
            price = get_price(element[0], date, index)
            prices.append(price)
            #print price
        else: prices.append(p)
        index += 1
    full_stock_prices.append((element[0], element[1], prices))

write_csv(full_stock_prices)
save = open("Pickled/full_stock_price.pickle","wb")
pickle.dump(full_stock_prices, save)
save.close()
print "run time:", time.time() - start
```

Save_All_Words.py

```
import glob, re, pickle, random, time
from nltk.tokenize import word_tokenize

# get file paths
def file_paths():
    return glob.glob('Domestic/*/*')

# get all the "clean" words in corpus
def save_words(filepaths):
    stop_words = []
    with open('TXT/stop_words.txt', 'r') as stop:
        for s in stop:
            stop_words.append(s.decode('utf-8').strip())
    extra_words = []
    with open('TXT/extra_words.txt', 'r') as extra:
        for e in extra:
            extra_words.append(e.decode('utf-8').strip())
    country_names = []
    with open('TXT/country_names.txt', 'r') as country:
        for c in country:
            country_names.append(c.decode('utf-8').strip())
    city_names = []
    with open('TXT/US_city_names.txt', 'r') as city:
        for c in city:
            names = c.decode('utf-8').split(' ')
            for name in names:
                city_names.append(name)
    airport_names = []
    with open('TXT/airport_names.txt', 'r') as airport:
        for a in airport:
            names = a.decode('utf-8').split(' ')
            for name in names:
                airport_names.append(name)
    remove_list = set(extra_words + country_names + city_names + airport_names)

    all_words = []
    for path in filepaths:
        with open(path, 'r') as f:
            sub_words = []
            for lines in f:
                words = word_tokenize(lines.decode('utf-8'))
                for w in words:
                    # remove abbreviation like 's / 've / 'm
                    w = re.sub(r'\W\w*', '', w)
                    # remove abbreviation like J. / D. / L.
                    w = re.sub(r'\w\W', '', w)
                    # remove phrases like 1q08 2Q15
                    w = re.sub(r'\dQ\d\d|\dq\d\d', '', w)
                    w = re.sub(r'FY\d\d|fy\d\d', '', w)
                    w = re.sub(r'\W+', '', w)
                    w = re.sub(r'\d+', '', w)
                    # remove punctuation
                    if len(w) > 3 and w not in remove_list:
                        w = w.lower()
                        if w not in stop_words:
                            sub_words.append(w)
            all_words.append(sub_words)
    save = open("Pickled/all_words.pickle","wb")
    pickle.dump(all_words, save)
    save.close()
    print '--all_words saved--'

start = time.time()
save_words(file_paths())
print "run time:", time.time() - start
```

Save_Rep_of_Transcripts.py

```
import pickle, time

# get intersection of transcripts and lexicon
def save_rep():
    with open("Pickled/all_words.pickle", "rb") as word:
        all_words = pickle.load(word)
    with open("Pickled/stock_price_senti_score.pickle", "rb") as score:
        scores = pickle.load(score)
    with open("Pickled/L_McDonald_Dict_POS.pickle","rb") as pos:
        pos_lexicon  = pickle.load(pos)
    with open("Pickled/L_McDonald_Dict_NEG.pickle","rb") as neg:
        neg_lexicon = pickle.load(neg)
    rep_of_trans = []
    for i, item in enumerate(all_words):
        word_rep = []
        for word in item:
            if word in (pos_lexicon + neg_lexicon):
                word_rep.append(word)

        word_score_dict = (dict([(word, True) for word in word_rep]), scores[i][2])
        rep_of_trans.append(word_score_dict)

    save = open("Pickled/rep_of_trans.pickle","wb")
    pickle.dump(rep_of_trans, save)
    save.close()
    print '--rep_of_trans saved--'

start = time.time()
save_rep()
print "run time:", time.time() - start
```

Calculate_Sentiment_Score.py

```
import pickle, csv

with open("Pickled/full_stock_price.pickle") as prices:
        stock_prices = pickle.load(prices)

stock_price_senti_score = []
with open('Stock_Prices_Senti_Score.csv', 'wb') as f:
    w = csv.writer(f)
    w.writerow(['Stock Symbol', 'Release Date'])
    for e in stock_prices:
        v1 = float(e[2][5]) - float(e[2][0])
        v2 = float(e[2][4]) - float(e[2][1])
        v3 = float(e[2][3]) - float(e[2][2])
        if v1 > 0:
            s1 = "> 0"
            if v2 > 0:
                s2 = "> 0"
                if v3 > 0:
                    s3 = "> 0"
                    senti_score = 1
                else:
                    s3 = "<= 0"
                    senti_score = -3
            else:
                s2 = "<= 0"
                if v3 > 0:
                    s3 = "> 0"
                    senti_score = 2
                else:
                    s3 = "<= 0"
                    senti_score = -1
        else:
            s1 = "<= 0"
            if v2 > 0:
                s2 = "> 0"
                if v3 > 0:
                    s3 = "> 0"
                    senti_score = 1
                else:
                    s3 = "<= 0"
                    senti_score = -2
            else:
                s2 = "<= 0"
                if v3 > 0:
                    s3 = "> 0"
                    senti_score = 3
                else:
                    s3 = "<= 0"
                    senti_score = -1
        w.writerow([e[0], e[1], s1, s2, s3, senti_score])
        stock_price_senti_score.append([e[0], e[1], senti_score])

save = open("Pickled/stock_price_senti_score.pickle","wb")
pickle.dump(stock_price_senti_score, save)
save.close()
```

Three_Way_Split.py

```
import pickle, random, time, nltk
from nltk.classify.scikitlearn import SklearnClassifier
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.svm import LinearSVC

def train_test(train_set, validation_set):
    NB_cl = nltk.NaiveBayesClassifier.train(train_set)
    MNB_cl = SklearnClassifier(MultinomialNB()).train(train_set)
    BNB_cl = SklearnClassifier(BernoulliNB()).train(train_set)
    LR_cl = SklearnClassifier(LogisticRegression()).train(train_set)
    SGD_cl = SklearnClassifier(SGDClassifier()).train(train_set)
    LSVC_cl = SklearnClassifier(LinearSVC()).train(train_set)

    nb_test = nltk.classify.accuracy(NB_cl, test_set)
    mnb_test = nltk.classify.accuracy(MNB_cl, test_set)
    bnb_test = nltk.classify.accuracy(BNB_cl, test_set)
    lr_test = nltk.classify.accuracy(LR_cl, test_set)
    sgd_test = nltk.classify.accuracy(SGD_cl, test_set)
    lsvc_test = nltk.classify.accuracy(LSVC_cl, test_set)
    f.write("NaiveBayes Test Accuracy: " + str(nb_test) + "\n")
    f.write("MultinomialNB Test Accuracy: " + str(mnb_test) + "\n")
    f.write("BernoulliNB Test Accuracy: " + str(bnb_test) + "\n")
    f.write("LogisticRegression Test Accuracy: " + str(lr_test) + "\n")
    f.write("SGDClassifier Test Accuracy: " + str(sgd_test) + "\n")
    f.write("LinearSVC Test Accuracy: " + str(lsvc_test) + "\n\n")

with open("Pickled/rep_of_trans.pickle", "rb") as rep:
        rep_of_trans = pickle.load(rep)
random.seed(time.time())
random.shuffle(rep_of_trans)
rep_of_trans_size = len(rep_of_trans)    # 325
cut_off = 0.2
test_cut_off = int(rep_of_trans_size * cut_off)
test_set = rep_of_trans[:test_cut_off]
rep_of_trans = rep_of_trans[test_cut_off:]
training_size = int(rep_of_trans_size * (1 - cut_off))
validation_size = int(rep_of_trans_size * (1 - cut_off) * cut_off)

counter = 1 / cut_off
f = open('Accuracy.txt', 'w')
for i in range(0, training_size, validation_size):
    validation_set = rep_of_trans[i:i+validation_size]
    train_set = [item for item in rep_of_trans if item not in validation_set]
    train_test(train_set, validation_set)
```

Appendix B: Best Accuracy

```
NaiveBayes Test Accuracy: 0.123076923077
MultinomialNB Test Accuracy: 0.492307692308
BernoulliNB Test Accuracy: 0.476923076923
LogisticRegression Test Accuracy: 0.415384615385
SGDClassifier Test Accuracy: 0.276923076923
LinearSVC Test Accuracy: 0.384615384615

NaiveBayes Test Accuracy: 0.107692307692
MultinomialNB Test Accuracy: 0.461538461538
BernoulliNB Test Accuracy: 0.430769230769
LogisticRegression Test Accuracy: 0.430769230769
SGDClassifier Test Accuracy: 0.338461538462
LinearSVC Test Accuracy: 0.4

NaiveBayes Test Accuracy: 0.107692307692
MultinomialNB Test Accuracy: 0.415384615385
BernoulliNB Test Accuracy: 0.415384615385
LogisticRegression Test Accuracy: 0.476923076923
SGDClassifier Test Accuracy: 0.384615384615
LinearSVC Test Accuracy: 0.446153846154

NaiveBayes Test Accuracy: 0.107692307692
MultinomialNB Test Accuracy: 0.446153846154
BernoulliNB Test Accuracy: 0.446153846154
LogisticRegression Test Accuracy: 0.384615384615
SGDClassifier Test Accuracy: 0.415384615385
LinearSVC Test Accuracy: 0.430769230769

NaiveBayes Test Accuracy: 0.0769230769231
MultinomialNB Test Accuracy: 0.461538461538
BernoulliNB Test Accuracy: 0.476923076923
LogisticRegression Test Accuracy: 0.4
SGDClassifier Test Accuracy: 0.338461538462
LinearSVC Test Accuracy: 0.384615384615
```