Analysis on New York Times Articles on Apple Inc.

STAT 418 Final Project
Ah Sung Yang

My final project is to find a potential relationship between the sentiment of New York Times articles on Apple Inc. and its share price returns.

1. Download Data

First, I downloaded articles on Apple Inc. from New York Times Article Search API. I looked up articles on Apple Inc. issued from January 1, 2014 to April 30, 2019 by setting keyword to "Apple" and the organization field to "Apple Inc." New York Times API has rate limits of 4,000 requests per day and 10 requests per minute. According to the New York Times website, adding 6 seconds of sleep between the calls will prevent a request from hitting the per minute rate limit. Also, the API returns 10 results at a time and allows you to paginate thru up to 100 pages by using the page query parameter. Since there were more than 1,000 articles on Apple Inc. issued from Jan. 1, 2014 to Apr. 30, 2019, the function was run twice to cover the entire period.

Furthermore, I used the function called "sentiment" of the TextBlob package to compute polarity and subjectivity scores of abstract, headline, snippet, paragraph of each article. The polarity score is within the range of -1.0 and 1.0, and subjectivity is within the range of 0.0 and 1.0. In default, the function uses "patternanalyzer" in the Pattern library to evaluate sentiment of the text. As an alternative, you can change the analyzer to "NaiveBayesAnalyzer" which is the NLTK Naive Bayes Classifier trained on a movie review corpus. I evaluated sentiment of each article by applying both analyzers.

Lastly, I used the yahoo_historical package to download the daily share prices of Apple Inc. The data sets of New York Times articles with sentiment scores and share prices were exported to csv files. Please refer to the "Article Data Download" Jupyter Notebook for codes.

2. NLTK Naive Bayes Classifier

I attempted to fit NLTK Naive Bayes Classifier to the articles after labeling them with the corresponding daily share price return. In the context of text analysis, Naïve Bayes Classifier considers each word in an article as a feature that determines the classification. The "Naive" part assumes the probability of observing a certain word is independent of each other.[1]

$$p(label \mid x) = \frac{p(x \mid lable) \times p(label)}{p(x)}$$

$$p(label \mid w_1, w_2, w_3, \cdots, w_n) \propto p(lable) \prod_{i=1}^{n} p(w_i|label)$$

$$p(label \mid w_1, w_2, w_3, \cdots, w_n) \propto p(label)\, p(w_1|label)\, p(w_2|label)\, p(w_3|label) \cdots$$

The probability that a certain article has a positive share price return is in proportion of the product of individual probabilities of seeing each word in the set of articles with positive returns.[2] The reason for

[1] Gustavo Chávez, "Implementing a Naive Bayes classifier for text categorization in five steps," Medium, Feb 28. https://towardsdatascience.com/implementing-a-naive-bayes-classifier-for-text-categorization-in-five-steps-f9192cdd54c3

[2] Gustavo Chávez, "Implementing a Naive Bayes classifier for text categorization in five steps," Medium, Feb 28. https://towardsdatascience.com/implementing-a-naive-bayes-classifier-for-text-categorization-in-five-steps-f9192cdd54c3

expressing it in terms of proportion is the denominator, p(x), is ignored. According to the NLTK document, for a feature that has never seen with any label, the classifier will ignore that feature as opposed to assigning 0 to all labels.[3]

In order to apply the Naive Bayes Classifier, I grouped articles based on the date and aggregated them in one text string. For the share price data, I computed daily share price returns by taking log difference on share prices and convert them into a binary response (positive, negative) indicating the direction of the price movement. This was followed by inner-joining the two tables so that all articles have the corresponding response label. I divided the table into train and test sets. The train set contains 700 data points, which accounts 90% of the data.

I used the "NaiveBayesClassifier" function in the TextBlob package and fit the classifier to abstracts, headlines, snippets and paragraphs, respectively. The highest accuracy rate was achieved at ~60% when I used abstracts to fit and test the classifier. The 60% accuracy rate is not necessarily high; however, given that the share price movement is affected by numerous factors, it is reasonable to expect the accuracy rate around this range.

I extracted the top 20 most informative features to see which words were considered valuable by the classifier. The ratio on the right side represents the ratio of occurrence of the word in positive and negative labels. For example, "worth" appeared 7 times more in articles tagged with a positive return than the ones tagged with a negative return. From the table 1, we can see that except for "worth" and "Street", the words that have a higher ratio (i.e. a bigger difference in the number of occurrences) are mostly the ones that appear more often in the negative label. It might suggest that articles issued on the day with a negative share price return have a higher tendency to repetitively use a certain set of words, which might have led to a higher ratio. Also, it is interesting that the words that don't necessarily have association with negative meanings, such as "we", "showed", "women", "internet" and "place", appeared more often in the articles with a negative share price return. However, given that the accuracy rate of the classifier is around 60%, it is uncertain how reliable the below informative features are. In order to make the classifier available to external parties, I created a Flask API that returns either a positive or negative label for an input of texts. Further instructions can be found in the Api directory of the Github repository. Please refer to "TextBlob – Naive Bayes Classifier" Jupyter Notebook for codes.

Table 1. *Top 20 Most Informative Features Extracted from the Naive Bayes Classifier*

```
Most Informative Features
            contains(we) = True              neg : pos    =      7.6 : 1.0
         contains(worth) = True              pos : neg    =      7.1 : 1.0
           contains(yet) = True              neg : pos    =      6.9 : 1.0
       contains(showed) = True               neg : pos    =      6.1 : 1.0
        contains(women) = True               neg : pos    =      5.4 : 1.0
  contains(improvements) = True              neg : pos    =      5.4 : 1.0
       contains(appeal) = True               neg : pos    =      5.4 : 1.0
       contains(Street) = True               pos : neg    =      5.2 : 1.0
      contains(lawyers) = True               neg : pos    =      4.7 : 1.0
     contains(pressure) = True               neg : pos    =      4.7 : 1.0
         contains(sign) = True               neg : pos    =      4.7 : 1.0
      contains(improve) = True               neg : pos    =      4.7 : 1.0
     contains(internet) = True               neg : pos    =      4.7 : 1.0
        contains(place) = True               neg : pos    =      4.7 : 1.0
    contains(president) = True               pos : neg    =      4.6 : 1.0
        contains(store) = True               pos : neg    =      4.6 : 1.0
         contains(Wall) = True               pos : neg    =      4.6 : 1.0
         contains(file) = True               pos : neg    =      4.0 : 1.0
        contains(third) = True               pos : neg    =      4.0 : 1.0
       contains(turned) = True               pos : neg    =      4.0 : 1.0
```

[3] NLTK 3.4.3 documentation "Source code for nltk.classify.naivebayes" https://www.nltk.org/_modules/nltk/classify/naivebayes.html

2. Sentiment/Polarity Scores

As an alternative approach, I tried to find a relationship between sentiment scores of the articles and share price returns by using the existing sentiment analysis packages and functions. Besides the functions implemented by the TextBlob package, I also used a package called “sentimentr” in R. According to the sentimentr document, it is a “dictionary lookup approach that tries to take into account valence shifters (i.e., negators, amplifiers (intensifiers), de-amplifiers (downtoners), and adversative conjunctions) while maintaining speed.”[4] The dictionary lookup approach simply compares the words in each sentence to a dictionary of polarized words, and tags positive and negative words with a +1 and -1, respectively. The package will then consider other factors listed above to adjust the score.

I used the polarity scores computed by the TextBlob (PatternAnalzyer and NaiveBayesAnalzyer) and Sentimentr packages to find correlation with corresponding daily share price returns. However, the correlation between polarity scores and share price returns turned out to be negligible, which was also supported by the result of logistic regression (using a binary response of the share price returns as the y variable) where none of the scores was statistically significant. Also, I attempted to fit the xgboost to the sentiment scores labeled with binary share price returns. however, its accuracy rate was roughly 50%, which also indicates lack of relationship between the two variables.

Given the low correlation between sentiment scores and share price returns, I decided to create a Shiny application which focuses on illustrating the movement of historical sentiment scores. The app includes plots of historical sentiment scores of New York Times articles on Apple Inc as well as the number of articles issued during the day. The app also allows a user to choose the type of analyzer (PatternAnalyzer, NaiveBayesAnalyzer and Sentimentr), the type of text data (abstract, headline, snippet and paragraph) and frequency of data (daily, weekly and monthly). For weekly and monthly sentiment scores, it will take an average of daily sentiment scores during the specified period. Also, in case the sentimentr package is chosen, the app will also display a table of top 10 most frequently mentioned negative and positive words throughout the entire period and the number of occurrences for each word. The application is currently running at https://ahsung-yang.shinyapps.io/sentiment/.

3. Areas to Explore

One of the areas to explore is using lagged share prices instead of the spot price considering it might take some time for the stock market to reflect the sentiment of articles. Also, combining New York Times articles with the ones from other sources such Bloomberg or Financial Times might improve the accuracy rate of the classifier given that articles from the above sources are more related to financial indicators of the company and the stock market movement. Trimming the text data such as taking out irrelevant words is also likely to improve the fit of the classifier.

[4] Sentimentr documentation. https://cran.r-project.org/web/packages/sentimentr/readme/README.html