Stock Sentiment Using Text Based Analysis

Marius Boda and Subham Das

September 20th, 2023

1 Introduction

The analysis and prediction of the stock market is a tremendously big market. Many banks and financial institutions are continuously looking for ways to help in prediction. Machine learning has been tried time and time again to 'beat' the market, but the staggering truth is that the market cannot be predicted. At its core, no individual nor institution nor machine learning module knows which way the market or a stock will trend. There are ways, however, to aid in the analysis of stocks or the grander market. These methods can provide insight into a stock and can provide investors with more information; thus allowing for many to make a more educated guess. One of these methods is analyzing the public sentiment of certain stocks based on tweets. For example, if the public sentiment of a stock is looking positive, we may see a lot of tweets talking about how individuals are optimistic for the future of stock 'ABC' or 'XYZ'. On the other hand, if sentiment is negative it is expected that certain tickers are associated with words that display a negative sentiment.

1.1 Report Outline

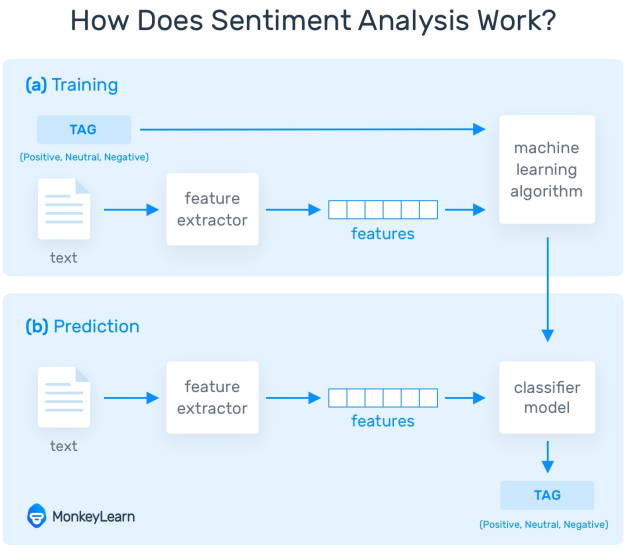
The first section, Section 1, is an introduction to the report. Section 2 goes into the details of the problem formulation. Section 3, discusses the methods used to achieve results. Results, then, are discussed in the following section, Section 4. Finally, Section 5 summarizes the report and provides a conclusion. Section 6 lists sources and Section 7 is the code appendix.

2 Problem Formulation

This machine learning task is a classification problem. Our classifier is inputted an excerpt of text, or tweets, and then outputs a result of either negative, positive, or neutral. We must be able to extract features from our tweets, we will do this by using a method called 'bag-of-words'.

Our data set is a set of 3887 tweets containing @apple or the hashtag for Apple. This ensures that all of our tweets pertain to Apple. Each data point has text, date, judged sentiment, and other miscellaneous info. The labels for our data set will be 1, 3, and 5. 1 is a negative sentiment, 3 is neutral, 5 is for positive sentiment.

The publicly available data set was found on www.crowdflower.com, it is a data set meant for sentiment analysis of Apple tweets. The data set contains prejudged sentiment for each tweet which will make it easy to compare our ml results with what is expected.



Figur 1: Problem Formulation. Source: MonkeyLearn

3 Methods

3.1 Pre-Processing

We have a collection of 3886 tweets relating to Apple. Data pre-processing included creating headers for each column and then extracting only the columns relevant.

```
data = pd.read_csv("Apple-Twitter-Sentiment-DFE.csv", sep=';', encoding='utf-8', header = 0)
headers = data.iloc[0]
data.columns = headers
data = data.iloc[1:]
filtered_data = pd.DataFrame()
filtered_data['sentiment'] = data['sentiment']
filtered_data['sentiment confidence'] = data['sentiment:confidence']
filtered_data['text'] = data['text']
text_data = pd.DataFrame({'text': filtered_data['text']})
```

3.2 Feature Selection

Feature selection for the classification of text-based sentiment is done with a 'Bag-of-Words' model. In essence, we create vectors for each tweet that contain a binary reading for each column. Each column is a single word in our vocabulary. First, to create our vocabulary, we define a function.

```
def create_vocabulary_list(data):
    vocabulary = set()
    for text in data['text']:
```

```
        words = str(text).lower().split()
        vocabulary.update(words)

    vocabulary = {word for word in vocabulary if "http://" not in word}
    vocabulary = {word for word in vocabulary if "https" not in word}
    vocabulary = {word for word in vocabulary if "@" not in word}
    return vocabulary
```

This function takes in our filtered data set and returns the set of all words contained in all 3886 tweets. This function additionally filters out words that start with 'http://' or 'https' or '@'. If need be, we can adjust and add more filters to make our 'Bag-of-Words' model better.

The next step in creating our feature vectors is making the document term matrix. To do this we define a new function which goes through each single tweet and converts it into a vector form. This vector form, for each individual tweet, contains '1' in every column where the column's header word appears in the tweet. Similarly, the vector contains '0' in every column where the column's header word does not appear in the tweet. An example is provided after the code below.

```
def create_document_term_matrix(data, vocabulary_list):

    document_word_counts = []

    for _, row in data.iterrows():
        document = str(row['text']).lower()
        word_counts = {word: document.count(word) for word in vocabulary_list}
        document_word_counts.append(word_counts)

    # Create a DataFrame from the list of dictionaries
    document_term_matrix = pd.DataFrame(document_word_counts)
    document_term_matrix.index = range(1, len(document_term_matrix) + 1)
    document_term_matrix = document_term_matrix.fillna(0)  # Fill NaN values with 0
    return document_term_matrix
```

As an example, imagine that one of our tweets in our data set is: 'Hello, this is my first tweet'. The vocabulary for this space is Hello, this, is, my, first, tweet. Now, the document term matrix for this tweet, 'Hello, this is my first tweet', is [1, 1, 1, 1, 1, 1] because every word in the vocabulary appears in the tweet. Now if we add more data to this example we will see how the document term matrices change. For example, our new data set is: 'Hello, this is my first tweet', 'Hello, I hate this'. Our new vocabulary is now, Hello, this, is, my, first, tweet, I, hate. The document term matrix for the first data point is [1, 1, 1, 1, 1, 1]. For the second data point it is [1, 1, 0, 0, 0, 0, 1, 1].

3.3 ML Model

The chosen machine learning model for this use case is Naive Bayes Classifier. Using the sklearn package we use MultinomialNB to complete our task. Naive Bayes Classifier uses probability theory and more specifically Bayes' Theorem.

3.4 Loss Function

For the chosen Naive Bayes classifier, a maximum likelihood function is the most optimal since the method of estimating the labels based on the classifier is very probabilistic and such a loss function works best with probabilistic predictions.

3.5 Model Validation

The data set has been split into 60% training set, 20 % test set, and 20% validation set. We have chosen this split because of the vast amount of data that we have. We want to have a lot of data in our training set, and this is why we have it at a staggering 60 %.

4 Results

5 Conclusion

6 Sources

Referenser

[1] sentiment analysis `https://monkeylearn.com/sentiment-analysis/`

[2] crowdflower `https://data.world/crowdflower/apple-twitter-sentiment`

7 Code Appendix