Twitter Sentiment with Stock Price Trend

Xin Liu, Ai Jiang, Zihan Niu
CS585 Nature Language Processing Course Project

A. ABSTRACT

Social media content such as Twitter messages, can provide an accurate reflection of public sentiment on decision making when taken in aggregation. In this project, we primarily examine several machine learning techniques on providing a two-class (positive or negative) sentiment tweets which from our data collection of recently daily Twitter posts. Additionally, we apply optimal classifier to unlabeled sentiment twitter, then combine sentiment score and features from stock market data to accomplish our stock prediction task. To train all labeled sentiment tweets we collected, firstly, we created features such as POS, NER, N-gram for each tweet. We apply Principal Component Analysis(PCA) to do the feature reduction. Then, we use our selected optimal classifier to the unlabeled tweets to predict their sentiment score, combined the sentiment score and stock market information to forecast stock price. For the task of determining sentiment, we test three common effective machine learning techniques: Logistic Regression (which be used as our baseline model), Naive Bayes and Support Vector Machine. We select the optimal classifier based on the best accuracy among these models. For the predict stock price part, like the sentiment prediction part, we first created features for the tweets related to three technology companied. And we applied SVM in order to achieve a prediction about future stock movements.
In this project, the primary mission for us is to create POS, NER, N-gram and select significant features for tweet sentiment prediction. Then we try to investigate whether related public sentiment is correlated or even predictive of stock prices. After comparing different combinations of our feature sets, our results show that bigram and trigram features influence the accuracy of the classifier most. According to the result, we also believe that changes in the public sentiment can affect the stock market movement.

B. INTRODUCTION

Nowadays, large amounts of text are transmitted online through social network, which contains numerous topics. Twitter, as one of the most popular social network, has over 400 million tweets sent per day. Thus, we think to catch information from tweets become a very effective way to analysis interesting subject. Though each tweet may not be significant as a unit, a large collection of them can provide data with valuable insight about the common opinion on a particular subject. Predicting the fluctuation of stock prices depends on many factors, which included the public sentiment from social network.
In the rest of this paper, we first describe some related works about sentiment analysis technology. In the following section, we present the approaches about how we collect and preprocess tweets data from Twitter and how we created our feature sets for training our classifiers. In the section E , we explain the methods to evaluate our models and presence some experiment results of our selected models. We also describe the method we used to predict the stock market movement based on predicted sentiment scores, which explore the correlation between tweets sentiment and stock market. In the section F, we summarize our results and propose some ideas for the our future work.

C. RELATED WORK

A broad overview of some of the machine learning techniques used in sentiment classification is provided in [1]. They investigate methods in determining whether movie reviews are positive or negative. They provide an overview of Naive Bayes, Maximum Entropy, and Support Vector Machine classification techniques in the movie review domain. [4] discusses the use of Twitter specifically as a corpus for sentiment analysis. They discuss the methods of tweet gathering and processing. Their training set was split into positive and negative samples based on happy and sad emoticons, which greatly reduces manual tweet taggingAdditionally, they analyze a few accuracy improvement methods. Similarly, [2] presents a discussion of streaming Tweet mining and sentiment extraction, while furthering the discussion to include opinion mining. [6] presented one of the first indications that there may be a correlation between Twitter sentiment and the stock market. In their work, a sentiment score is correlated with the DJIA and then fed into a neural network to predict market movements. The authors use a mood tracking tool named OpinionFinder to measure mood in 6 dimensions. Then, they correlate the mood time series with DJIA closing values by using a Self Organizing Fuzzy Neural Network. [7] achieves a 75% accurate prediction mechanism using Self Organizing Fuzzy Neural Networks on Twitter and DJIA feeds. In their research, they created a custom questionnaire with words to analyze tweets for their sentiment. Their work is similar to [6], with a few minor modifications.[9] combines sentiment gathered from news with technical features of stock data. Sentiment is analyzed using SentiWordNet 3.0, a lexical resource for opinion mining. Instead of Twitter, they scrape data from Engadget. Multiple kernel learning is run to find the best coefficients for the different factors in their system for stock prediction. [8] discusses three methods of sentiment analysis on tweets: document-level, sentence- level, and aspect-based. They conclude that aspect-based analysis, which tries to recognize all sentiment within a document and the aspects which they refer to, For this point, we plan to adopt their idea but focus on finding the most impactive sentiment for predicting the stock price.

D. APPROACH

1. Data Collection

We use Yahoo finance API and Twitter API to collect the data we needed. In this project, there are three kinds of data needed.

• Data for sentiment analysis

In this project, sentiment of one sentence is either positive or negative. In order to train the sentiment classifier, we need to collect sentences that related to these two labels. There are two way to label the data.

First one is manually label the sentence. We can collect tweets from twitter without any filter and label. Then we manually tagging the tweets as ‘positive’ or ‘negative’.

The second method is that, setting up some keywords that related to the label. Then searching tweets contains that keywords, which should belong to that label.

There are no doubt that the first method will be more precisely, since the second method may consider ‘not good’ as ‘positive’ which should belong to ‘negative’. However, the dataset needed for sentiment classifier is too large and it’s really inefficient to manually label all the data. Thus we combined these two method. Following is our approach.

Using twitter search API with ‘search/tweets’ request to query the tweets with specified keywords. Then we manually check the query result, correcting some wrong labeled data. Following is our keyword list used in the query.

pos_label = ['good', ':)', ':-)', '=)',':D', '<3', 'like', ‘love']

neg_label = [‘bad’,’:(‘,’=(‘,’hate','dislike']

• Data for stock prediction

We choose three company to see the relation of twitter sentiment and their stock. We need to collected data from twitter, analyzing the sentiment towards the target company. These three company is: apple, google, facebook. The reason that we choose these three company is that, these company is very popular which will ensure that we’ll get sufficient data.

However, the stock only open at week days, thus we only collect the twitter on week days. To collecting these part of data, we used twitter streaming api, which will collect the real time twitter. In order to collect sufficient data, we run the program each day to get that day’s twitter data. The filter track we set in the twitter streaming api is as following:

label_apple = ['apple', 'iphone', 'macbook','ipad']

label_fb = ['facebook', 'fb']

label_google = ['goole', 'gmail', ‘chrome’]

• Data of real stock

We use Yahoo finance API to get the real stock prices of the company.

However, their are rate limit in using twitter API which should be concerned in collecting data. When using twitter search api, we called sleep function to avoid the rate limitation which cost a lot of time in data collecting.

In this project, we collect 10,000 tweets for sentiment analysis. And we collect 1000 tweets/ per day of target companies for stock price prediction.

2. Tweet text pre-processing

To accurately obtain a tweet’s sentiment, we need to preprocessing the collected tweet text since the text of each tweet contains many irrelevant words that could not have a good reflection of its sentiment. We first need to filter the noise from its raw form. We incorporate a variety of methods:

- Remove noise tweets characters and words.

- Create bag of words for each text: tokenization each word in the tweet by space and punctuation marks, which will be used as features to train our sentiment classifier.

- Remove the stopwords from each text, we check each word in the bag of words against the stopwords dictionary which provided by Python's NLTK library, and simply filter it out. The list of stopwords contains articles, some prepositions, and other words that add no sentiment value.

- Remove Twitter symbols that add no value and sentiment meaning to the text.(such as '@', as well as URLs.). Also remove any non-word symbols. We keep the words following '#' since they may contain information which are useful for categorizing this tweet.

3. Feature extraction

We create the features from our collected and processed dataset. In [4], they tried to determine the best settings for the microblogging data and reported that high-order n-grams, such as trigrams, should better capture patterns of sentiments expressions though unigrams provide a good coverage of the data. Thus, we decide to choose unigram, bigrams and trigrams to build our feature set.

- Unigram: We use each unique word(token) in a tweet as a unigram feature.

- Bigrams & trigrams: We add bigrams and trigrams increase the features in our feature set by n-1 for bigrams and n-2 for trigrams, where n is the number of individual words in each tweet in our dataset. For example, for tweet 'I like nature language processing', we create 'I_like', 'like_nature', 'nature_language', 'language_processing' as our bigram features for this tweet. Similar with creating trigrams.

- External lexicon: We inclusion an external lexicon called SentiStrength Lexicon, which remedy the missing words problem in out collected Tweets dataset.

- Part-of-speech tag: We create POS tag for all words in each tweet we collected by using the pos tagger in NLTK's library , as another feature dimension. For the repeated POS tags in a tweet, we treat the appearance not the frequency as our feature. For example, if we have two 'drink' tokens in a tweet, we just count the feature 'pos=NN' (POS tag is noun) as the feature without considering the frequency of the 'NN' POS tag appears in a tweet.

- Name entity tag: We create NER tag for all words in each tweet we collected by using the Stanford NER tagger package from Stanford NLP Group, as another feature dimension. For the repeated NER tags in a tweet, we treat the appearance not the frequency as our feature. For example, if we have four 'Apple' tokens in a tweet, we just count the feature 'ner=o' (NER tag is organization) as the feature without considering the frequency of the 'o' NER tag appears in a tweet.

4. Build a sentiment classifier

After we build our feature set, we construct a classifier that can classify 'positive' and 'negative' label to new tweets with optimal accuracy. In this project, we examine two common classifiers: Logistic Regression, Naïve Bayes and Support Vector Machines. We use python 'Scikit-learn' library to do this machine learning task.

- Logistic Regression

In logistic regression, there a list of input data, and parameter, using a function to predict the to predict the probability that a given example belongs to the "1" class versus the probability that it belongs to the "0" class. The function we used to learn the data is as following:

$$p(y_i|\vec{x}_i) = \frac{1}{1 + e^{-y_i \vec{x}_i \cdot \vec{\theta}}}$$

In this project, we use Logistic Regression as our baseline model. the input x is the vector of each tweets, and y is the class this tweet belong to. In order to prevent overfitting, there are two penalty function can be applied to logistic regression. During implementation, we use logistic regression classifier as our baseline. However, there are kinds of ways to set up the parameters for training the classifier. We compared the evaluation result and choose the best settings.

- Naïve Bayes

Naïve Bayes classifier is probabilistic classifier based on Bayes rule and conditional independence assumption. With the Naive Bayes model, we do not take only a small set of positive and negative words into account, but all words the NB Classifier was trained with, i.e. all words present in the training set. If a word has not appeared in the training set, we have no data available and apply Laplacian smoothing.

$$p = p(c)\prod_i p(d_i\,|c) = p(c)\prod_i^n \frac{count(d_i, c)}{\sum_i count(d_i, c)} = p(c)\prod_i^n \frac{count(d_i, c)}{V_c}$$

Vc does not change, so it can be placed outside of the product:

$$p = \frac{p(c)}{V_C^n}\prod_i^n count(d_i, c)$$

Then we can classify new tweets by calculating the class probability for each class and select the class with the highest probability.

- Support Vector Machine

The second classifier we used in our analysis is Support Vector Machine. Unlike the Naïve Bayes classier, the SVM is large margin classifier, rather than probabilistic. SVM attempt to find a separation between a linearly separable set of data, with as wide of a gap as possible between them, called a margin. With our input training dataset, we can use SVM to find the hyperplane so that each point can correctly classified and the hyperplane is maximally far from the closest points. ‘support vector’ means the points on the margin which between the hyperplane and the nearest data points. SVM try to find a parameter vector ‘a’ that maximizes the distance between the hyperplane and every training point. In essence, it is an optimization problem:

$$\begin{gathered} Minimize\ \frac{1}{2}a * a \\ s.t.\ \ y(a * x_i + b) \geq 1 \end{gathered}$$

where y is the class label (-1, 1) for negative and positive. Once we build our SVM classifier, classification of new tweets simply determining which side of the hyperplane that they fall on. In our sentiment analysis, there are only two classes, positive and negative, we did a research to compare different kind of kernel function. Linear kernel is much faster and the performance is not bad. Thus we use linear kernel function in this project.

E. EXPERIMENT

1. Datasets

For the sentiment prediction part, we use tweets with the programming label as the training data. For the stock prediction part, we use the unlabeled tweets data from the most popular three tech. companies plus the stock market information as the training data.

2. Baseline model

We choose the Logistic Regression model as our baseline model, we compare the accuracy, precision and F-score on different models with baseline model, in order to decide which classifier will be better applied in our tweet sentiment prediction task.

3. Evaluation

For the purpose of our analysis, we used k-fold cross validation and found the average accuracy. K-fold cross validation splits the training dataset into k smaller datasets. Every time left one out as testing dataset, and the other k-1 subsets are used to train. This procedure is repeated k times. Firstly, we measure accuracy for each classifier. Secondly, we measured precision and F1 scores for each of the labels. Precision is measured as the number of true positive tweets over all positive tweets:

$$Precision = \frac{tp}{tp + fp}$$

Recall is the true positive rate of the classifier and it’s calculated as follows:

$$Recall = \frac{tp}{tp + fn}$$

In this project, one sentence can have lots of features, such as token(unigram), POS, NER, bigram, trigram.
However, there are more than 10,000 feature when using unigram features. It will result overfitting if we do not cut off the feature dimension. There are also experiments for feature dimension selection. In this project, we've tried several numbers (i.e.1000,2000,3000). We finally choose 1000 most important features . We list the performance result of different classifier using different feature combination in Chart_3.

- Choosing the parameter setting per classifier.

This is baseline of our project. Only unigram feature is used in this model. From the Chart_2, we can clearly see that Logistic Regression and SVM classifier have a better performance. In SVM classifier, although rbf kernel has a better performance, it costs too much time to train the classifier. So we'll use linear kernel in SVM classifier.

penalty	solver	accuracy	precision	f1
L1		0.917	0.917	0.915
L2	newton-cg	0.924	0.924	0.923
	lbfgs	0.919	0.919	0.918
	liblinear	0.922	0.922	0.921
	sag	0.924	0.924	0.923

Chart_1. Logistic Regression Parameter Comparison

classifier		accuracy	precision	f1
naive bayes		0.633	0.633	0.622
svm	linear	0.929	0.929	0.926
	poly	0.633	0.633	0.580
	rbf	0.933	0.933	0.930
	sigmoid	0.633	0.633	0.580

Chart_2. Naive Bayes and SVM Parameter Comparison

- Choosing feature combinations per classifier

Since our training data is balanced, we only consider accuracy as our evaluation parameter.

	logistic	svm	naive bayes
unigram	0.924	0.929	0.633
bigram	0.960	0.936	0.895
trigram	0.925	0.937	0.793
unigram + ner	0.938	0.930	0.630
unigram + pos	0.969	0.959	0.643
bigram + trigram	0.954	0.963	0.926
bigram + ner	0.931	0.972	0.853
bigram + pos	0.927	0.947	0.783
trigram+ner	0.923	0.977	0.736
trigram +pos	0.908	0.977	0.776

Chart_3. Comparison Between Different Combination (Accuracy)

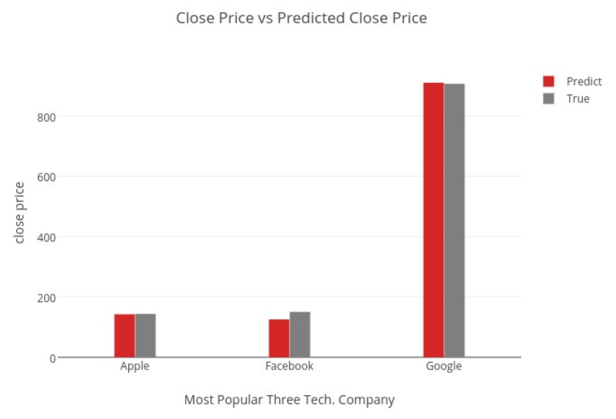
4. Stock market prediction

After model selection and feature selection, we decided to use bigram and trigram as our tweet features, SVM as our classifier. To predict stock price, we focused on three top technology companies' stock information according to Yahoo! Finance, which is Apple, Google and Facebook. We created bigram and trigram features for these unlabeled tweets related to these three companies and applied SVM to predict their sentiment. Then we compute percentage of positive sentiment score and percentage of negative score for each company. Besides, we created two additional fundamental metrics:

$$HLPCT = \frac{High - Low}{Low}$$

$$PCTchange = \frac{Close - Open}{Open}$$

HLPCT stands for 'High-Low Percentage', PCTchange means 'percentage of change'.



Fgure_1

So our feature matrix include following features:

- percentage positive sentiment score
- percentage negative score
- close price
- HLPCT
- PCT change
- volume

We applied linear regression classifier to predict these three companies' stock price. Finally, we compared out predicted result with the actual value in Figure_1.
From this plot, we can observe that over predict price and actual price are pretty close. Conclusively, we believe that public sentiment can effective reflect stock market.

F. CONCLUSION

In this project, from the beginning part, we learned how to use Twitter's Search API to collect our sentiment tweet dataset and Yahoo! Finance to collect stocks. Then we explored how to create efficient features such POS, NER, N-gram for tweet test information. In the model selection section, we practiced the process of training classifier and how to evaluate our results. Furthermore, in the prediction stock section, we experimented with how to combine sentiment score and stock information to forecast stock movement.

In the future work, we plan to collect more data for more precisely prediction. We also could add another label for our sentiment classifier class, neutral label, and extend more feature dimensions. Besides, we will try to implement more classifier to explore the sentiment analysis on Twitter data, and hope it will give us more interesting results.

G. REFERENCES

[1] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. pages 79–86, 2002.
[2] A. E. Stefano Baccianella and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In LREC. LREC
[3] Albert Bifet and Eibe Frank. Sentiment knowledge discovery in twitter streaming data. 2011.
[4] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. Proceedings of LREC, 2010.
[5] A. Bromberg. Sentiment analysis in python. 2013
[6] J. Bollen and H. Mao. Twitter mood as a stock market predictor. IEEE Computer, 44(10):91–94, 2010.
[7] Anshul Mittal and Arpit Goel. Stock prediction using twitter sentiment analysis. 2012.
[8] Ronen Feldman. Techniques and applications for sentiment analysis. Communications of the ACM, 56(4):82–89, 2013.
[9] Shangkun Deng, Takashi Mitsubuchi, Kei Shioda, Tatsuro Shimada, and Akito Sakurai. Combining technical analysis with sentiment analysis for stock price prediction. In Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, pages 800–807, EEE Computer Society, Washington, DC, USA., 2011. DASC '11 IEEE Computer Society.
[10] Alec Go, Lei Huang, and Richa Bhayani. 2009. Twitter sentiment analysis. Final Projects from CS224N for Spring 2008/2009 at The Stanford Natural Language Processing Group.