Stock Market Prediction Report

Shihan Ran - 15307130424

***Abstract*—This project is aimed at using Text Classification and Sentiment Analysis to process financial news and predict whether the price of a stock will go up or down. For reading and saving data, I use libraries like xlrd, pickle and codecs. In terms of tokenization, I choose Jieba. To achieve higher accuracy rate, I've added some financial dictionary to Jieba and removed stop-word from the already tokenized word list. As for extracting features, both positive and negative word dictionary are used and only considering the most common words in news for the purpose of reducing features dimension. Talking about training and testing models, I divided the Development Set into Training Set and Dev-Test Set, and have used cross validation to find the best classifier among Naive Bayes, Decision Tree, Maximum Entropy from nltk and Bernoulli NB, Logistic Regression, SVC, Linear SVC, NuSVC from sklearn. Finally, the best accuracy was achieved at 69.5% with SVM.**

I. Introduction

TEXT Classification and Sentiment Analysis has always been a Popular topic in NLP. With the amount of text files on Internet increasing exponentially each day, the volume of information available online continues expanding. Text Classification, as the assignment of text files to one or more predefined categories based on information contained from text files, is an important component in information management tasks.

In this project, we focus on financial news classification, to predict whether the price of a stock will go up or down. First, we discuss the principles of Bayes Theorem and the Nave Bayes algorithms. Then we discuss the principles of Nave Bayes Text Classifier by introducing the design and implementation of the stock market predictor. Finally, we analyze the testing results and discuss the future work.

II. Naive Bayes

Naive Bayes is a supervised algorithm which can be defined as Bayes Theorem with a conditional independency assumption that all variables $f_1, \cdots, f_n$ in a given category C are conditional independent with each others given C, which means we have $P(f_i, f_j|Class) = P(f_i|Class)P(f_j|Class)$.

In a probability inference task, our goal is to calculate the probability of a hypothesis C holds given conditions that data $f_1, \cdots, f_n$ have been observed. Using Bayes Theorem, we know that

$$\begin{aligned} P(C|f_1, \cdots, f_n) &= \frac{P(C)P(f_1, \cdots, f_n|C)}{P(f_1, \cdots, f_n)} \\ &= \frac{1}{Z}P(C)\prod_{i=1}^{n} P(f_i|C) \end{aligned}$$

where $Z = P(C)P(f_1, \cdots, f_n|C)$ is a constant given one probability inference task.

In stock market prediction, we use Naive Bayesian algorithm to determine which category a financial news belongs to. We choose the category Cj which yields maximum probability value of $P(Cj|f_1, \cdots, f_n)$. Then, naturally we have the formula for Naive Bayesian classification algorithm:

$$v_{NB} = \arg\max_{C_j \in C} P(C_j) \prod_i P(A_i|C_j)$$

III. Things Before Modeling

Before we get started, we should consider several questions:

1) Is an extra dictionary needed?
2) Do we need to do cross-industry analysis?
3) What algorithms should we choose? We are doing a simple classification or a much more difficult regression?
4) Should we pay attention to single word's sentiment or we can just see through the whole news?
5) How can we find the balance between using rules and improving the programs' speed?
6) How can we find the balance between accuracy and speed?

We'll answer those questions in the following sections.

IV. Design of Text Classifier

As a supervised classification application, the stock market predictor needs a training set with pairs of $(\text{news features}, \text{labels})$ indicating whether the price of a stock will go up or down.

The training stage of the stock market predictor includes following steps:

A. *Preparation of Training Set*

Reading *train.txt* and *news.txt* from disk, matching news number in *train.txt* with news in *news.txt*, title and content are stored respectively because intuitively we think title is much more important than content, hence I choose to store it separately instead of simply adding title into content or just discarding title.

Training Set is divided into positive set and negative set with pairs of ([[title1],[content1],[title2],[content2],...],labels). Then I use *pickle* package to store the already processed data into disk in order to facilitate training process.

Notice: We should pay extra attention to the encoding and decoding problems in Chinese.

B. *Generating word lists*

Jieba tokenizer creates a word list for every title and content by tokenizing it. Considering the news is from financial field, I downloaded financial and stock funds word dictionary from

Sogou[1] and loaded it in *Jieba*. And I use *cut_all=False* because the precision mode is enough for this project.

In text classification, words that appear more often than a given frequency in a set of text files are called stop words which make no contribution to effective text classification. We use a stop word list [2] to delete stops words from word lists.

C. *Extracting features*

Considering we are doing a sentiment analysis and text classification task, I have picked following features.

- The frequency of most common words
- The appearance of positive and negative words
- The existence of Not words and degree words
- The number of news related to this stock
- The number of sentiment words
- The average length of news

For the first one, I calculate the most common 1000 and 2000 words from the already processed news using *FreqDist*, which is provided by *nltk*. Then I process every title and content, only considering if it contains the most common words, returning features with pairs of ('increase', 'True').

For the second and third one, after surfing the internet, I downloaded sentiment words dictionary from Dalian University of Technology[3], Not and degress words dictionary via HowNet knowledge Database[4]. Intuitively, we both think the frequency is more informative than appearance(although some papers have proved verses is true), and it's wildly agreed that the appearance of Not word is vital because it can change the sentiment of the news directly, furthermore, the degree word sometimes can be helpful to determine the confidence of your classification. Finally, I decided to generate two features:

1) Simply considering the appearance of positive and negative words
2) Calculating sentiment score of the news using positive, negative, not, degree words

Again, we can process every title and content, only considering if it contains positive and negative words, returning features with pairs of ('dislike', 'True').

In order to calculate sentiment score, thanks to Dalian University of Technology, they have labeled every sentiment word with a number between 1 and 10 as an extent, so that I can use these numbers directly. Degree words in HowNet knowledge Database were divided into *extreme/most, very, more, some, insufficiently, over*, so I manually labeled the corresponding words as {10, 7, 6, 2, 1, 5}. At last, the special word dict is like {'good':5, 'bad':-5, 'extreme':10, 'very':7}.

The idea is pretty much naive, first I tried to use *LocateSpecialWord* to find the location of Sentiment words, Not words, and Degree words, additionally, storing the index into corresponding arrays *SentiLoc, NotLoc, DegreeLoc*. Then I iterate each word, the Score can be computed as $Score = Score + W * (SentiDict[word])$, where W is the weight depends on Not words and Degree words, $SentiDict[word]$ is the labeled extent number of sentiment word. When we are doing our iteration, W should change when there exist Not words or Degree words between one sentiment word and next sentiment word.

The algorithm can be described as following codes:

```
1: function SentimentScore($SpecialDict, sentence$)
2:     $SpecialWordLoc \leftarrow$ LocateSpecialWord()
3:     $Weight \leftarrow 1, Loc \leftarrow 0$
4:     for word in sentence do
5:         if $index \in SentiLoc$ then
6:             $Loc = Loc + 1$
7:             $Score = Score + W * (SentiDict[word])$
8:             if $Loc < len(SentiLoc) - 1$ then
9:                 for $no. \in [SLoc[Loc], SLoc[Loc + 1]]$ do
10:                    if $no, \in NotLoc$ then
11:                        $W = W * (-1)$
12:                    end if
13:                    if $no. \in DegreeLoc$ then
14:                        $W = DegreeLoc[word]$
15:                    end if
16:                end for
17:            end if
18:        end if
19:    end for
20: end function
```

Algorithm 1: SentimentScore

V. Training and Testing

The training of the stock market predictor is the process of computing probabilities require by:

$$v_{NB} = \arg\max_{C_j \in C} P(C_j) \prod_i P(A_i|C_j)$$

Prior probabilities $P(Cj)$, $Cj\{positive, negative\}$ are computed as: $P(Cj) = \frac{news_j}{total_{news}}$, where $news_j$ is the number of news in the given class, and $total_{news}$ is the total number of news in the training set. For a given class $news_j$, to each word w_i in the word lists, we use $P(Wk|Cj) = \frac{n_k}{n}$ to compute $P(Wk|Cj)$, n_k is the occurrences of Wk in $news_j$, and n is number of news in $news_j$.

A testing set is a set of samples with pre-classified label values as the training set and is used to determine the classification accuracy of the text classifier. By comparing the label value assigned by the classifier with the pre-classified label value, we can compute the classification accuracy of the text classifier.

A. *Use Dev-Test set*

The corpus is divided into Development set and Test set, go a step further, we divided the Development set into Training set and Dev-Test set.

Hence, we do training on Training set, testing, adjusting parameters and choosing the best classifier base on the results from Dev-Test set. At last, we use our trained best classifier to do test on testing set.

B. *Find best Classifier and informative features*

For classifiers, I've written my own NaiveBayes Classifier and I also considered several available classifiers in *nltk* and *sklearn* like {'Maximum Entropy', 'DecisionTree', 'BernoulliNB', 'LogisticRegression', 'SVC', 'LinearSVC', 'NuSVC'}.

As I said, intuitively we think title is much more important than content, hence I give title a weight, and during iteration, I'll find the most suitable weight.

C. *Single fold and Cross Validation*

To conduct cross validation, first is to find the training set and testing set every round. It can be realized by:

$subset\_size \leftarrow int(math.floor(len(train\_group)/n))$

$testing\_this\_round \leftarrow train_group[i * subset\_size :][: subset\_size]$

$training\_this\_round \leftarrow train\_group[: i * subset\_size] + train\_group[(i+1) * subset\_size :]$

Iterate classifier in candidate list and run cross validation, store the classifiers into disk using *pickle*, and compute the average accuracy, precision, recall, F1 score.

VI. Result analysis

Firstly, let's pay attention to how important title should be. I only use sentiment word as features, by multiplying a weight to title's score, and plot the accuracy of different models under different weights of title between 1 to 10, I got:

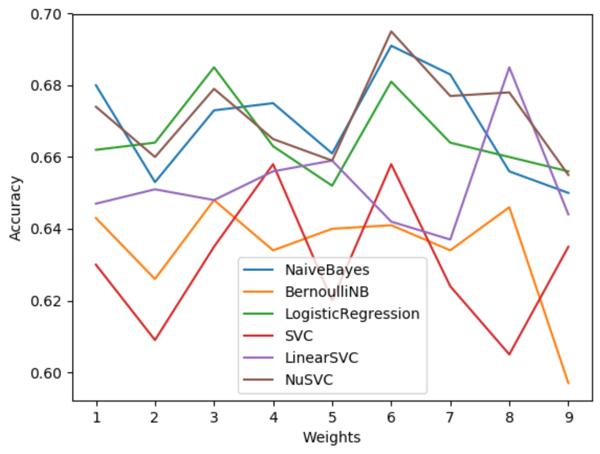

Fig. 1. The accuracy of different models under different weights

And the best accuracy is achieved as the following Table I.

TABLE I
ACCURACY UNDER DIFFERENT WEIGHTS

Classifier	Weight	Best Accuracy
NaiveBayes	6	0.691
BernoulliNB	3	0.648
LogisticRegression	8	0.687
SVC	6	0.658
LinearSVC	8	0.685
NuSVC	6	**0.695**

We can naturally draw a conclusion that the best accuracy is reached when weight of title $\in \{3, 6, 8\}$. And among these classifiers, most accurate ones $\in$ {'NaiveBayes', 'LogisticRegression', 'LinearSVC', 'NuSVC'}.

Then set $Weight = 6$ and I ran Cross Validation models, the result I got on Dev-Test set through different models is as the following Table II.

TABLE II
ONLY USE SENTIMENT WORDS

Classifier	1-fold	5-fold	10-fold
NaiveBayes	0.658	**0.668**	0.666
BernoulliNB	0.629	0.633	0.630
LogisticRegression	0.659	0.668	**0.672**
SVC	0.646	0.652	0.658
LinearSVC	0.647	0.653	0.658
NuSVC	0.653	0.666	**0.673**

Next I fixed Cross Validation at 5-fold, and tried many more possible combinations of features, the result I got on Dev-Test set through different models is as the following Table III.

In the Table, *Frequency* means we **only** use the most common words as features, and *Freq_1000* means we use the most 1000 common words, similarly, *Freq_2000* means we use the most 2000 common words. *Sentiment* means we **only** use the appearance of sentiment words and sentiment score as features, *All* means we use the whole features set as mentioned above:

- The frequency of most common words
- The appearance of positive and negative words
- The existence of Not words and degree words
- The number of news related to this stock
- The number of sentiment words
- The average length of news

TABLE III
DIFFERENT COMBINATIONS OF FEATURES

Classifier	Freq_1000	Freq_2000	Sentiment	All
NaiveBayes	0.635	0.640	**0.668**	0.661
BernoulliNB	0.568	0.581	0.633	0.632
LogisticRegression	0.656	0.662	**0.668**	0.664
SVC	0.655	0.647	0.652	0.538
LinearSVC	0.655	0.655	0.653	0.575
NuSVC	**0.677**	0.677	0.666	0.618

VII. Conclusion

The main objective of this project is to apply Naive Bayesian algorithm on financial news classification. We explored Bayes Theorem and Naive Bayesian algorithm for text classification. Also, we explored methods used for text preprocessing, probability computation, sentiment analysis in text classification.

In this project, we have learned the advantage and easy-to-use feature of Naive Bayes classifier. However, there does remain scope for improving the performance of the stock market predictor.

The stock market predictor uses a static model with probabilities of words unchanged during the classification step. Such a model is restricted by the fact that flaw of prior knowledge may cause noticeable inaccuracy in its classification. Statistically speaking, the bigger the training set is and the more randomly that news are chosen, the better model can be obtained. However, the size of the training set is always limited and the randomly picked news only reflect the trend of news in a given period. Thereby, a more comprehensive and robust model can be constructed by tuning probabilities of words in the vocabulary table dynamically during the classification step. The advantage of such a dynamic tuning mechanism is that it enables the model to reflect the latest trend of news, which eliminates the cost of manually adjust the model or even rebuild the model.

References

[1] Financial word dictionary in SoGou cell dictionary, (2017) [Online] Available: http://pinyin.sogou.com/dict/detail/index/15127?rf=dictindex (2017/11/13)
[2] Most 1208 Common Chinese Stop-word, (Nov, 2012) [Online] Available: http://www.datatang.com/data/43894 (2017/11/15)
[3] Positive and Negative word dictionary, (2008) [Online] Available: http://ir.dlut.edu.cn/EmotionOntologyDownload (2017/11/13)
[4] Not and Degree word dictionary, (Sep, 2007) [Online] Available: http://www.keenage.com/html/c_index.html (2017/11/13)