Stock Market Prediction with Naïve Bayes and NLTK

Ning Shangyi, *14300180071*

Abstract

In this project, we usd text classification algorithms to process financial news and predict whether the price of a stock will go up or down in a particular day. We implemented naïve bayes and decision tree by ourself and tried several other algorithms to classify the news and stocks. Finally, the best accuracy was achieved at 63.2% with decision tree which is restricted to be more universal and 63.7% with naïve bayes provided by *nltk* .

Index Terms

Language Model, Naïve Bayes Classifier, Stock Prediction

I. Introduction

IN this project, we will implement several classifiers to predict trend of stocks in a particular day by some natural language processing approaches. We implemented models including naïve bayes, k-means and decision tree method. Naïve bayes got a final accuracy of 62.1% on predicting a specific stock. Meanwhile, since the characteristic of the dataset, we proposed a new and simpler method to evaluate the models. The accuracy on predicting a specific news of different algorithm are namely 51.1%, 52.2% and 57.8% for naïve bayes, k-means and decision tree. The accuracy may be higher for this problem, but our algorithm will be more universal. In section IV, we will talk about it in detail.

II. Development Environment

- Ubuntu 16.04.1
- Python 3.5.2
- Spyder 2.3.8
- jieba 1013 for segmentation

III. Background and Dataset

The provided data contains 3 parts, *'news.txt'* contains 30339 news which are related to some stocks. The test data and training data in *'test.txt'* and *'train.txt'* contain lists of stocks and which news they are related to. Training data also provides a label reveals whether the stock went up or down in that day. '+1' means up and '-1' means down.

We want to predict the stocks in test data for the same day using the provided news.

IV. Evaluating the Accuracy

Since the provided data are from the same day, there is an important feature that most of the stock went up at that day. As a result, if we label all the stock with '+1', which means we affirm all the stock would go up, we will got a final accuracy of 63.7%, which is pretty good.

But this is all-'+1' model is not what we want, because if one day most of the stock went down, the model will be terribly bad.

Thus, we proposed a new and simpler method to evaluate the models. First, in order to adjust the proportion of positive and negative results, we predict news instead of stock. That is, we will predict whether the news is positive or negative to the stocks related to. Obviously, if we can correctly predict the news, so can we predict stocks. Further more, since there are several news related to one stock, the accuracy of predicting stock should be much higher compared to that of predicting news. As a comparison, The accuracy of hand made naïve bayes got an accuracy of 51.1% on news and 62.13% on stocks.

In addition, in order to avoid labelling all the news or stock with '+1', we restricted the proposition of positive result and negative result to 1. Thus, the final accuracy should be compared to 50%, if the accuracy is more than 50%, then the model is effective.

In this section, there is an assumption that there are exactly half of the news being positive to their related stocks. Actually, there are about 12000 pieces of news being positive and 13000 being negative, which is much better for model building compared to evaluating stocks.

V. Naïve Bayes Classifier

Naïve Bayes classifier is a classifier based on applying Bayes' Theorem with naïve independence assumptions between the features. In this problem, obviously, the feature is whether and how a word occurs in the news. Then how to extract the feature words becomes an important problem.

A. Data Preprocessing

1) Word Segmentation : The raw news are in Chinese, only split in to two parts, the title and the content of the news. In order to extract features, we should turn the title and content into word vectors, *i.e.* word segmentation. We used *cut* in *jieba* to complete this part. Further more, in order to avoid the influence of a single content, after word segmentation, the repeated word are removed. After removing the repeated words, the accuracy of our model went up 0.7% and 1.2% for news and stock in test set.

2) Dropping Useless Words: After word segmentation, there are a lot of useless words like punctuations and numbers, we implemented several lines of code in *findKeyWord* to avoid extracting them.

B. Extracting Features

In naïve bayes classifier, what we do is to compute the probability of features occur in both positive set and negative set. So we should find the most useful words. Since if there are too many features, the speed of our model will be really slow.

Useful has two means. One is that useful words can distinguish the content effectively, which means the proposition of frequency of occurrences should be higher. The other is that those words should not only work on only several pieces of news. As a result, those words must occur at least for some times, we set the number to 500 in our model, and the proposition is set to 2 times of the length of positive training data and negative training data.

This section is what we did in function *findKeyWord*.

C. Probability of a Word

The core part of bayes classifier is to compute the probability of the stock or the news to be positive. First of all, we should compute the probability of the news if there is a feature word in its content.

If the feature word occurs a times in positive training news and b times in negative training news, the bayes' theorem tells us, the probability of the news being positive is

$$p(+1|C) = \frac{P(+1)P(C|+1)}{P(C)} = \frac{1}{1+b/a}$$

where C denotes the feature word occurs.

If there are more feature words, say n words, the probability becomes

$$\begin{aligned} p(+1|C_1,\cdots,C_n) &= \frac{P(+1)P(C_1,\cdots,C_n)}{P(C_1,\cdots,C_n)} \\ &\approx \frac{P(+1)\prod P(C_i|+1)}{\prod P(C_i)} \\ &= \frac{1}{k}\frac{\prod kP(+1)P(C_i|+1)}{\prod P(C_i)} \\ &= \frac{1}{k}\prod \frac{k}{1+b_i/a_i} \end{aligned}$$

where $k = \dfrac{1}{P(+1)}$ is a constant. The step of approx used independent assumption in naïve bayes classifier.

We computed $\dfrac{k}{1+b/a}$ in *findKeyWord*and store it in dictionaries *bayesPTitle* and *bayesPContent* namely for the probability of title and content.

D. Probability of News

After computing the probability of a word, compute the production of feature words occurred in test content, we can easily calculate the probability of a piece of news being positive. What need to be pointed out is that, the coefficient $\dfrac{1}{k}$ is no longer needed because we will drop it in the evaluate step, which is talked in section IV and V-E. Remember what we compare is the probability of positive compared to other news.

E. Probability of a Stock

Lastly, we should calculate the probability of a stock being positive. Since we only want to know whether the probability is greater than 0.5, we can calculate in this way,

$$\begin{aligned} p(+1) > 0.5 &= \prod P_i(+1) > \prod P_i(-1) \\ &= \frac{\prod P_i(+1)}{\prod P_i(+1) + \prod P_i(-1)} > 0.5 \end{aligned}$$

where

$$\begin{aligned} P_i(+1) &= \frac{1}{k} \prod \frac{k}{1 + b_{ij}/a_{ij}} \\ P_i(-1) &= 1 - P_i(+1) \end{aligned}$$

and $P_i(+1)$ denotes the i-th news to be positive.

Section V-D and V-E are what we did in function *naivebayes* and *writeAnswer* .

VI. Classifiers with NLTK

A. Naïve Bayes

We also implemented the naïve bayes classifier provided by *nltk* . Since we can not restrict it to predict half of the news positive, the accuracy becomes 62.0%, and there are only several hundreds negative predictions, which is definitely not what we want.

We did not implement any changes to the *nltk* function, so the features are chosen by itself from words that occurs in most pieces of news.

B. K-means

K-means is an unsupervised learning method. We chose this method mainly for contracting.

We divide the news content into two groups by Hamming distance of the word vector generated in section V-A1. After classifying, name the group with more positive training news with positive group, and use it to classify the test data.

Since it is unsupervised, the result is not so good compared to others'.

C. Decision Tree

In order to restrict the algorithm to predict half of the news positive, after trying the function provided by *nltk* , we implemented a hand made script of decision tree.

The feature is chosen from the dictionary of probability of words generated in section V-C. If the most frequently occurred word in positive training set is in the content, we affirm the content to be positive. And if there are more positive pieces of news than negative ones, we affirm the stock will go up.

The decision tree method got a final accuracy of 63.2% on predicting stocks and 57.8% on predicting news.

VII. Conclusion

TABLE I
ACCURACY OF DIFFERENT MODELS

Models	Accuracy on News	Aacuracy on Stocks
Naïve Bayes by hand	51.1%	62.0%
Naïve Bayes by *nltk*	62.0%	63.7%
K-means	52.2%	62.4%
Decision Tree	57.8%	63.2%

The accuracy of different models are shown in Table I. The accuracy is hard to compare because Naïve Bayes and K-means by *nltk* can not be restricted. However, the restricted models has a higher precision because they predicted more negative results.