Feature Extraction Based Chinese Text Classification for Stock Market Prediction

Linyang He (15307130240)

November 17, 2017

Abstract

Text classification is a basic and important topic in Natural Lanuage Processing, which has lots of applications including stock market prediction. In our project, we will use some methods related to text classification to give a stock market prediction. We can utilize labeled news texts with positive or negative marks to train a classifer to estimate un-labeled news' effect on stock market changing tendency. First, we need to extract proper features. To gain proper features, we may use some methods like eliminating noise, boolean features extraction and most informational features finding to enhance the performance. Then we need to devise some classifiers such as naive bayesian, decision tree, sklearn classifiers, etc. In the end, we will compare all the features extractation methods and classifiers to get a relatively optimal model to predict the stock market.

1 Introduction

Stock market is the thermometer of a country's or a region's economic conditions, it is an essential component of market economy. Through the stock market, we might gain the information of enterprise's capital operation. If we could predict stock market in some ways, we might regulate and control the economy more effectively. A potential way is to predict stock market through a bunch of news relative to business or finance. Specifically, for a certain stock, if some recent news about it discusses the negative impact of it or states events where investors are not optimistic, this stock might go down in future days and vice versa. Therefore, we can utilize text classification technique to estimate whether a piece of news indicates a certain stock to go up or go down.

Text classification is an important subfiled in NLP research with numerous applications. Usually, the procedure of text classification includes data preparation, words frequence distribution counting, features extraction, classification and evaluation. Besides, considering the news is based on Chinese, we need to use Chinese segmentation toolkits. More details are covered in the following sections.

The rest of the report is structured as following: Section 2 introduces the background we may need. Section 3 describes the data and the experimental setup. Section 4 presents the experiment result of the F1 score and consuming time varying from different feature extraction and classify methods. Section 5 summarizes related work while Section 6 concludes our report.

2 Background

2.1 Chinese Word Segmentation

Since Chinese words are not divided by space by default, we have to separate the sentences into semantic unit. In this project, we choose "Jieba" as our segmentation toolkit. Here's how it works:

```
In [5]: import jieba

In [6]: sent = "我爱自然语言处理"
```

```
In [7]: sent_cut = jieba.lcut(sent)

In [8]: sent_cut
Out[8]: ['我', '爱', '自然语言', '处理']
```

2.2 Feature Extraction

As we know, text is not a kind of structural data type. Therefore, to structure our dataset, we should use "features" to present the text. Usually, in text classification, word frequency is the most used feature. However, sometimes we don't need to use all the features, because it's quite slow to run the program if we consider all the words. Besides, some low-frequency words might be noise instead of features, which might interference our system. In this case, we have some methods to improve the feature extraction. Such as

1. **Double the frequency of words shown in the titles**. We know that the title of a text is a brief concentration of the text, which is quite important. So a method to show the importance is to double the frequency of thoes words in title.

2. **Choose top *n* features**. This method can eliminate noise word with little frequency as well as enhance the speed of the program.

3. **Choose a certain interval of frequency features**. Considering that the most frequent words for both classes might be some words or punctuations of no great importance just like "the" in English. There's quite little semantical importance of these words, so we might strip them, too.

4. **Use boolean distribution instead of frequency distribution**. Sometimes, a word's multiple appearance cannot tell us more information. So we might just conisder if the word is shown in the text rather than counting its frequency.

5. **Choose the most informative features**. We can use NLTK' s *classifier.show_most_informative_features(n)* function to find those most informative features. But what should pay attention is that to avoid overfitting, these features' frequency shouldn't be lower than a certian number, which we can get a proper one in our experiment.

6. Other methods like considering mutual information might help, too.

2.3 Classifier

2.3.1 Naive Bayes Classifier.

This is a quite useful still quite efficient classifier used widely in text classification. For a text t and a class c, according to bayes formula, we have

$$P(c|t) = \frac{P(t|c)P(c)}{P(d)}.$$

So to get the most possible class, we have

$$\begin{aligned} c_{MAP} &= \text{argmax} P(c|t) \\ &= \text{argmax} \frac{P(t|c)P(c)}{P(t)} \end{aligned}$$

Since when we are classifying the same text, the *P(t)* doesn't matter. So we have:

$$C_{MAP} = \text{argmax} P(t|c)P(c)$$

P(c) is the prior probability which we can gain from

$$P(c = neg) = \frac{N_{neg}}{N}$$

and

$$P(c = pos) = \frac{N_{pos}}{N}.$$

Here N_{neg} denotes the number of "-1" stock in our train dataset whileN_{pos} denotes that of "+1" in the training data. N denotes the number of the whole stocks.
After feature extraction, text t can be represented as features $v_1, v_2, ...v_n$. So we can consider$P(t|c)$ as $P(v_1, v_2, ..., v_n|c)$.
In this project, we have two assumptions, wrong might they be in fact, but they are quite useful to simpify our model. That is:

- Bag of Words Assumpotion. We believe that the position doesn't matter.
- Conditional Independence. We assume the feature probablities $P(v_i|c_j)$ are independent from each other given class c.

So we have
$P(v_1, \ldots, v_n|c) = P(v_1|c)P(v_2|c)...P(v_n|c)$.
We have

$$C = \text{argmax}P(c_i)\prod P(v_j|c)$$

As for $P(v_j|c)$, using maximum likelihood estimates we have

$$P(v_i|c_j) = \frac{count(v_i, c_j)}{count(c_j)}.$$

To smooth the formula using *Laplace* method, we have

$$P(v_i, c_j) = \frac{count(v_i, c_j) + 1}{count(c_j) + |V|}$$

2.3.2 Decision Tree Classifier

2.3.3 Maxent Classifier

We can use these classifiers with NLTK toolkit straightly.

2.4 Evaluation

Unlike regression problem, all we need is just the accuracy, in classification problem, we should consider both recall and precision. Now we introduce **noise matrix** like this

	Model("+1")	Model("−1")
Truth("+1")	TP	FN
Truth("−1")	FP	TN

And let

$$Precision = \frac{TP}{TP + FP}$$

and

$$Recall = \frac{TP}{TP + FN}$$

We have

$$F1 = \frac{2PR}{P + R}.$$

Usually, we use F1 score to evaluate our model.

3 Project

3.1 Dataset Preparation

All our dataset is based on Chinese news. So we segment these text first. We choose *Jieba* for our project. To enhance the efficiency of the program, we could save the dataset after word segmentation into a binary file called as *"news_dataset_cut.txt"*. After we save the dataset to file, every time we run our program, we don't need to do segmentation over and over agian, we just need to load this file.

3.2 Feature Extraction

We will use different methods of feature extraction to see the differnce of F1 score.

3.3 Classification

We write our bootstrap version of Naive Bayes classifier. As for the others, we just use NLTK.

4 Results

	NB	NB_NLTK	DTree	Maxent
Time	80s	150 s	> 1 h	ME
Recall	0.27	0.83	0.56	/
Precision	0.69	0.76	0.65	/
Accuracy	0.46	0.72	0.526	/
F1 Score	0.39	**0.79**	0.6	/

Table 1: All Features with Double Title

The "ME" denotes memory error, which means the data of all features is too large for this classifier.
Now for the boostrap version naive bayes model, we choose different features to see the f1 score.

Next we will discuss some NLTK classifiers. We select 20 dimension of each training set's feature, and we have the result.

For Maxent model, we can have the iteration information:

NB	Top1000	Top100	[20,100]	[20,1000]
Time	120s	150 s		
Recall	0.27	0.83		
Precision	0.69	0.76		
Accuracy	0.46	0.72		
F1 Score	0.39	**0.79**		

Table 2: Different Features with Double Title for Bootstrap NB

	NB_NLTK	DTree	Maxent
Time		> 1 h	680s
Recall			0.87
Precision			0.66
Accuracy			0.63
F1 Score			**0.75**

Table 3: Features with Double Title for NLTK Models

Iteration	Log Likelihood	Accuracy
1	-0.69315	0.366
10	-0.23632	0.973
20	-0.14762	0.990
30	-0.10858	0.994
33	-0.10072	**0.995**
40	-0.08630	0.995

Table 4: Maxent Iteration Information with 20 Features

We notice that when the iteration num comes to 33, the accuracy converges. So the real time of it to classify may just one third of the whole time we see in Tible3.

5 Summary

We can find that the NLTK naive bayes model is better than the boostrap version. We believe this is because that NLTK might use better smoothing methods.

6 Conclusions

Using all features with NLTK naive bayes is the optimal classifier.

7 References

[1] *Natural Language Processing With Python.* Steven Bird, Ewan Klein & Edward Loper