Is the Company You Are Investing Trustworthy?

Christopher Wang

***Abstract*—We studied the trustworthiness of Chinese companies by processing court files. Results of SVM and Naïve Bayes are compared. We start with the related work that has been done in the literature. After that, we describe the data collection process for this project in detail. Next we explain the methods we used to classify the instance and their results. Finally we discuss the effect of feature selection.**

I. Introduction

INVESTORS usually rely on credit rating analysis provided by Credit Rating Agencies to help facilitate investment decisions in the US. The credit rating is usually based on financial variables, which includes long-term debt, net income, total asset, net worth, etc. However, besides the financial status of the companies, investors care about the reputation and integrity of the company too. To help investors to get another view of the companies that they want to invest, this project is focused on building a credit model for Chinese construction companies based on court files. The reason behind it is that if the company is sued for any reason, and proved by the court that it is guilty, they break their code of trustworthy. The more guilty court cases are found for a particular company, the less the investors can trust it. Court files are just one example of the resource; we can also use news, magazine articles, etc.

To be more specific, given a Chinese court case file as an input, it will return if the defendant company is guilty or not. When the investors have a specific inquiry about a company, we will summarize how many guilty cases this company has went through.

II. Related Work

The classification problem of determining if the company is guilty by court file is a NLP learning problem. Similar tasks have been done for different objectives. In [1], the effectiveness of different inductive learning algorithms (Find Similar, Naïve Bayes, Bayesian Networks, Decision Trees, and Support Vector Machines (SVM)) in terms of learning speed, real-time classification speed and classification accuracy. In the mean time alternative document representations (words vs. syntactic phrases, and binary vs. non-binary features), and training set size are explored.

There are also enormous work regarding feature engineering in the literature. The type of representation that dominates the text classification literature is known as the "bag of words". Often, most frequent and infrequent words are removed, together with case and punctuation. Moreover, stop words that are functional or connective words are also filtered out. Additionally different morphological forms of the same word will be mapped to the common stem using stemming algorithm [2].

Since bag of words representation disrupted the paragraph, sentence and word order, researchers have also been seeking other means to represent the text, such as phase-based representations [3].

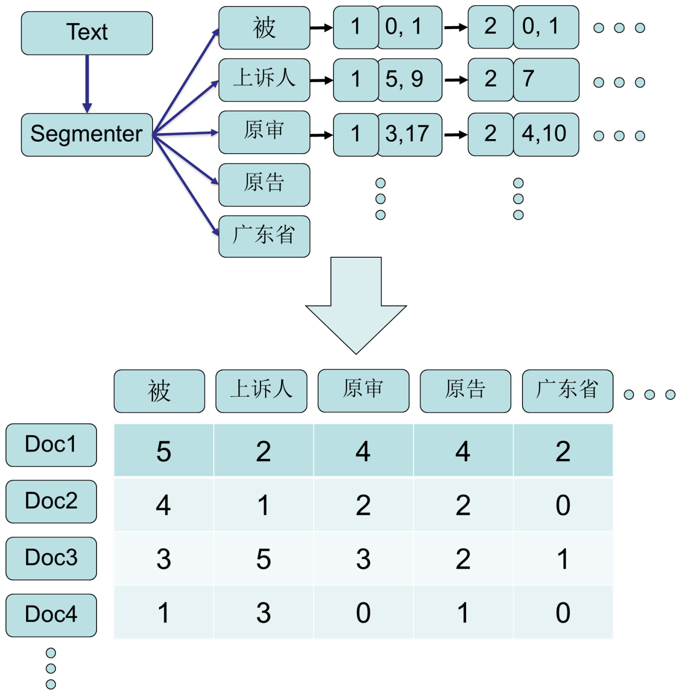
III. Data collection

The data used in this project was crawled using Scrappy from http://www.court.gov.cn/zgcpwsw/, specifically the court cases are in Guangdong Province of China. There are approximately 300,000 court cases (6 GB). The following are the steps I took to clean the data:

- Parse html files to retrieve the body documents.
- Segmented the documents using The Stanford Chinese Segmenter (http://nlp.stanford.edu/software/segmenter.shtml). Different from English, different combination or even orders of characters can mean completely different things. In addition, Chinese is standardly written without spaces between words. These facts distinguish tokenization of Chinese from that of English.
- Built inverted index with segmented documents.This is mainly for finding the exact defendant company names in a court case. To do that, we start from the inverted index, and

then find the company name by searching the position after the location of defendant (usually in Chinese 被告，被申诉人，被申请人) within bounded distance till the word 公司 (meaning "company"). Then, we can also record the number of guilty cases that a specific company we are interested in has gone through.
- Given a designated list of construction companies we are interested in, we filter the overall datasets, so that all the remaining cases have at least one of the construction companies as a defendant.
- Till now, we can build the feature set as all the tokens that appeared in the training data, and the corresponding feature value being the number of occurrence for that token.

The process is also depicted in Fig. 1.



	被	上诉人	原审	原告	广东省
Doc1	5	2	4	4	2
Doc2	4	1	2	2	0
Doc3	3	5	3	2	1
Doc4	1	3	0	1	0

Fig. 1. Example of data collection and pre-processing procedure.

IV. Methods

In this project, we used two algorithms to classify: SVM and Naïve Bayes. In the following, we briefly introduce these two methods.

A. SVM

SVM is to find the hyperplane (i.e. decision boundary) linearly separating our classes. Imagine we have a binary classification problem with labels $y \in \{-1, 1\}$ and features x. In general, the SVM is a classifier that can be represented as

$$h_{w,b} = g(w^T x + b), \tag{1}$$

where $g(z) = 1$ if $z \geq 0$, and $g(z) = -1$ otherwise. To reflect a very confident set of predictions, SVM tries to maximize the margin, which is the minimum distance between the decision boundary to the training data points. Equivalently, that is to find the w, b to solve the following optimization problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$s.t. y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \cdots, m$$

There are also cases that data is not perfectly linearly separable. Under this situation, one wants to allow some data points of one class to appear on the other side of the boundary. In this case, we can introduce slack variables $\epsilon_i \geq 0$ for each x_i. And the quadratic programming problem becomes:

$$\min_{w,b,\epsilon} \frac{1}{2}\|w\|^2 + C\sum_i \epsilon_i$$
$$s.t. y^{(i)}(w^T x^{(i)} + b) \geq 1 - \epsilon, i = 1, 2, \cdots, m$$
$$\epsilon_i \geq 0.$$

B. Naïve Bayes

In the Na ive Bayes model, we assume that the features x_i's are conditionally independent given the label y. In the multinomial setting based on the training data, we will built the conditional probabilities for each labeling using Maximum Likelihood Estimator:

$$\phi_{jk|y=1} = \frac{\sum_{i=1}^m 1\{x_j^{(i)} = k \& y^{(i)} = 1\}}{\sum_{i=1}^m 1\{y^i = 1\}}, \tag{2}$$

$$\phi_{jk|y=0} = \frac{\sum_{i=1}^m 1\{x_j^{(i)} = k \& y^{(i)} = 0\}}{\sum_{i=1}^m 1\{y^i = 0\}}, \tag{3}$$

$$\phi_y = \frac{\sum_{i=1}^m 1\{y^{(i)} = 1\}}{m}. \tag{4}$$

Then, for given feature values in the test data, we then simply calculate

$$p(y = 1|x) = \frac{p(x|y = 1)p(y = 1)}{p(x)}$$
$$= \frac{(\prod_{i=1}^n p(x_i|y = 1))p(y = 1)}{(\prod_{i=1}^n p(x_i|y = 1))p(y = 1) + (\prod_{i=1}^n p(x_i|y = 0))p(y = 0)},$$

and pick whichever class has the higher posterior probability.

Due to the fact that in the NLP problem, the feature values are normally very sparse, i.e., a lot of them will be zeros. To prevent the scenarios of $0/0$, we use Laplace smoothing, which replace Eq. (2)-(4) with

$$\phi_{jk|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = k \& y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^i = 1\} + 2}, \quad (5)$$

$$\phi_{jk|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = k \& y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^i = 0\} + 2}, \quad (6)$$

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 1}{m + k}. \quad (7)$$

C. Results

In this project, we trains the data manually, there are 120 files with labels in total. We split them into two different ratios for the sake of comparison: one is 70% training vs. 30% testing; the other is 50% training vs. 50% testing. In all the cases, we use the features that appears in less than 118 documents. In other words, we stripped out the most frequent features that occur almost in every documents.

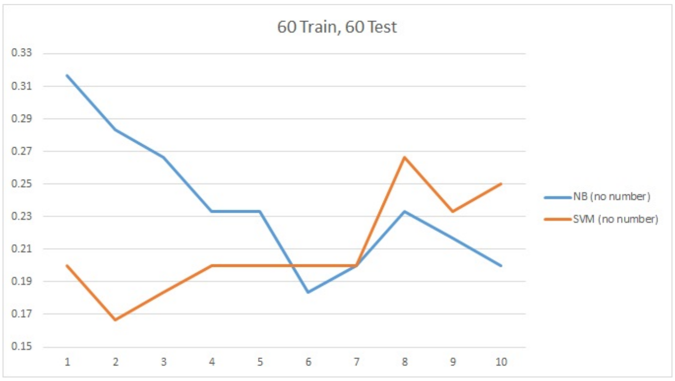


Fig. 2. Comparison of Naïve Bayes and SVM using features without including numbers (50% training, 50% testing).

Fig. 2 and 3 depict the testing errors vs. feature sets included in more than x documents. The x-axis indicates the number of infrequent features stripped. In general, Naïve Bayes performs better when more infrequent words are removed, while the SVM gets worse.

Fig. 4 shows the testing errors impacted by different feature selection: all the features, features without numbers, and feature without numbers and single characters. In Chinese, most single words are

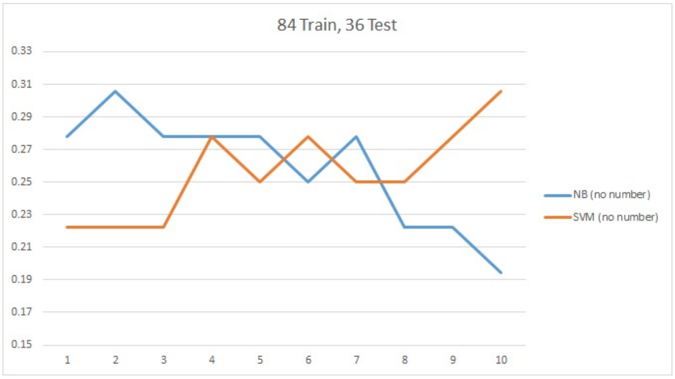


Fig. 3. Comparison of Naïve Bayes and SVM using features without including numbers (70% training, 30% testing).

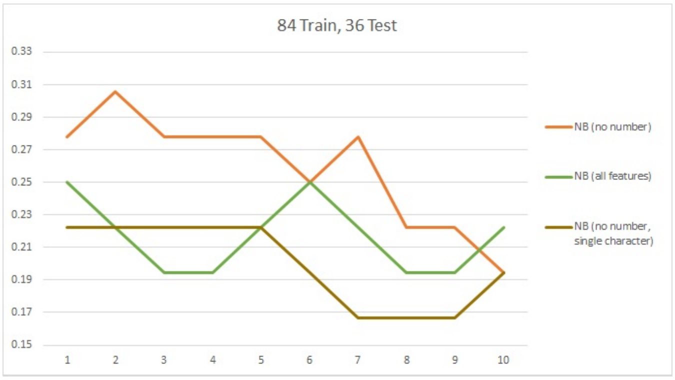


Fig. 4. Comparison of different feature selections on Naïve Bayes Method (70% training, 30% testing).

either people's last name or some connective characters. In general, by more selective and informative features, Naïve Bayes performs better and better. Fig. 5 shows the impacts of feature selections on SVM, where the patterns are more random. With more features stripped out, SVM performs worse. It performs better when it includes all the features. If you cross compare the results on SVM and that on the Naïve Bayes, Naïve Bayes performs more stable than that of SVM.

V. More Discussions on Feature Selections

It is found from this study that as we stripped more infrequent features out, the most indicative words in the Naïve Bayes make more sense in Chinese. Fig. 6 shows the actual top 20 indicative words. There are still some name, location related words other than number. However, it is very challenge to remove those words out in Chinese, as there

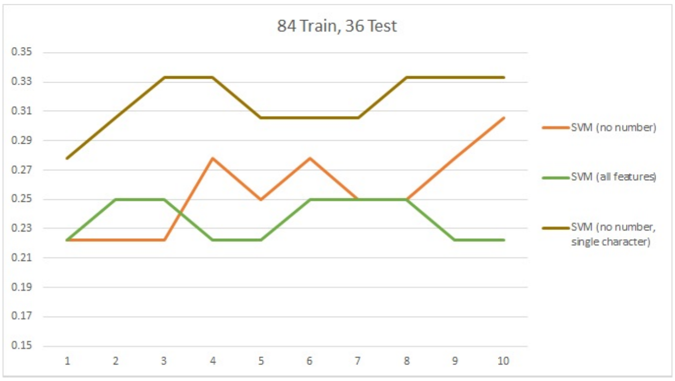


Fig. 5. Comparison of different feature selections on SVM (70% training, 30% testing).

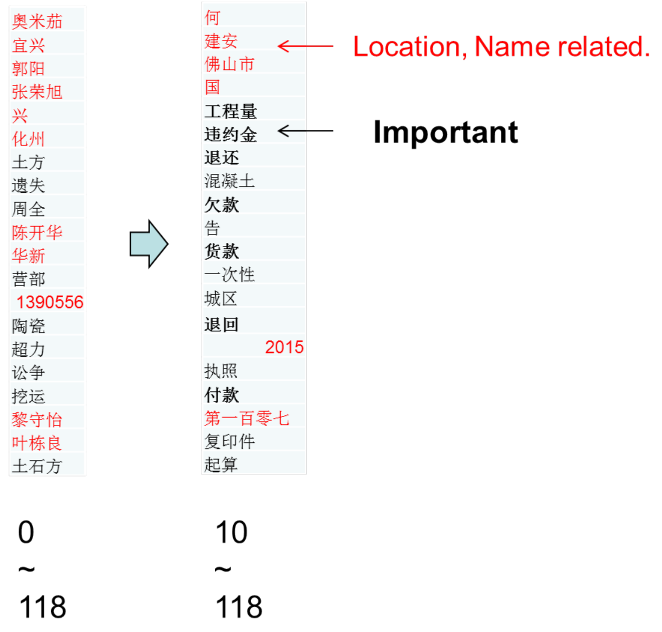
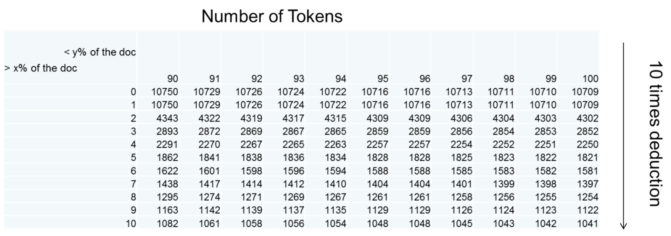


Fig. 6. Comparison of different sets of most indicative words by stripping more infrequent features.

are a lot of location and names, which is very hard to exclude them all out.


Number of Tokens

< y% of the doc / > x% of the doc	90	91	92	93	94	95	96	97	98	99	100
0	10750	10729	10726	10724	10722	10716	10716	10713	10711	10710	10709
1	10750	10729	10726	10724	10722	10716	10716	10713	10711	10710	10709
2	4343	4322	4319	4317	4315	4309	4309	4306	4304	4303	4302
3	2893	2872	2869	2867	2865	2859	2859	2856	2854	2853	2852
4	2291	2270	2267	2265	2263	2257	2257	2254	2252	2251	2250
5	1862	1841	1838	1836	1834	1828	1828	1825	1823	1822	1821
6	1622	1601	1598	1596	1594	1588	1588	1585	1583	1582	1581
7	1438	1417	1414	1412	1410	1404	1404	1401	1399	1398	1397
8	1295	1274	1271	1269	1267	1261	1261	1258	1256	1255	1254
9	1163	1142	1139	1137	1135	1129	1129	1126	1124	1123	1122
10	1082	1061	1058	1056	1054	1048	1048	1045	1043	1042	1041

Fig. 7. Feature Size Deduction Trend.

In Fig. 7, we can see that the deduction speed of infrequent words are very high. One smarter way would be to gather some nature of Chinese language with infrequency of some of the words. For example, we can strip infrequent single characters. As a result, we will end up with even less features, that may enable some other ways that we would not do with enormous features, such as decision trees.

Another complication of the court cases are that some of the cases may involve more than one defendants. For now, I use one class to label them all. But in reality, there are scenarios that one company is guilty, while the other is not. Further thinking needs to be done for those more complicated cases.

VI. Conclusion

In this project, we have classified the company involved in court cases to be trustworthy or not by machine learning algorithms. In particular, Naïve Bayes and SVM are used for learning. We conclude that, Naïve Bayes performs slightly better than SVM. Feature selection is a very critical task in NLP. Besides common ways of selecting features, such as stripping infrequent and frequent words out, we should combine the nature of the language to do it more smartly.

References

[1] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155. ACM, 1998.

[2] Julie B Lovins. *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory, 1968.

[3] Sam Scott and Stan Matwin. Feature engineering for text classification. In *ICML*, volume 99, pages 379–388. Citeseer, 1999.