第六届“认证杯”数学中国

数学建模国际赛

承　诺　书

我们仔细阅读了第六届“认证杯”数学中国数学建模国际赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的, 如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们允许数学中国网站(www.madio.net)公布论文，以供网友之间学习交流，数学中国网站以非商业目的的论文交流不需要提前取得我们的同意。

我们的参赛队号为：1214

我们选择的题目是： B

参赛队员 (签名) ：

队员 1：Ouyang Antai

队员 2：Zhang Litian

队员 3：Ming Xuran

参赛队教练员 (签名)： 无教练员

第六届“认证杯”数学中国

数学建模国际赛

编 号 专 用 页

参赛队伍的参赛队号：（请各个参赛队提前填写好）：

1214

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

Author Identification Based on Machine Learning SVM Model

Abstract:

This article mainly solves the problem of how to judge the author's ownership by handwritten samples. E-mail is the main sample of handwriting analysis. Based on the principle of multi-class support vector machine, the author uses SVM algorithm to get the effective evaluation of the author's features through the sample training. By extracting the features of the mail samples, the e- Value, compare the author's eigenvalues of the sample training value results, and solve the problem of authorship of samples written by e-mail.

First of all, using the principle of multi-class support vector machine, some basic features of language features are separated by processing 1702 sample data which have been classified simply. On the one hand, we analyzed the author of the sample email, extracted and analyzed the characteristic attributes of each email sent by the author, comprehensively considered and evaluated, and trained each author to give certain eigenvalues. On the other hand, according to the rough eight mail categories, the eigenvalues of each e-mail are obtained by assigning values to the different e-mail categories, and the eigenvalues of the authors obtained after training are verified, so that the accuracy of the results is about 60% .

After that, in order to further verify the eigenvalues of the authors, we study the variants of the SVM model in detail and add the symbolic attributes to calculate the eigenvalues of the authors. After extracting samples' attributes again for training, an efficient and efficient support vector machine model was optimized, and the obtained eigenvalues were imported into LibSVM software for simulation experiments to determine the author's ownership accuracy increased to about 90%.

In this paper, the method of sample verification is used to verify the model and analyze the error. After analysis, the data error is small after the feature attributes are continuously improved. In addition, we also analyzed and improved the advantages and disadvantages of the model, and discussed the function of functional words and special texts.

Key words:

SVM algorithm, LibSVM, feature attributes, author attribution,

Contents

I. Introduction

1.1 The restatement of the problem

With the development of network information technology, e-mail has become an indispensable means of information exchange in people's daily life. However, while e-mail brings convenience to people, it also brings many new problems, such as spam and virus mail, causing serious harm. Because such email senders always try to hide their real identity to avoid scouting, email author identification research is imperative. The author's identification algorithm mainly includes statistical algorithms and machine learning methods. In recent years, the rapid development of machine learning technology has led to the widely used machine learning algorithms in the identification research of authors, such as neural networks, Markov chains, support vectors Machine, decision tree and genetic algorithm.

In this problem solving, based on the principle of multi-classification SVM, we try to filter out some of the more obvious characteristics of the attributes, the author's writing habits quantitative description. After some samples verify the accuracy of the model we built. Finally, the accuracy rate of more than 90% is reached, which provides a good reference for the settlement of the ownership of the email author.

1.2 Problem Description

1.2.1 Problem1: How to set the eigenvalue?
1.2.2 Problem2: How to get the eigenvalue training?
1.2.3 Problem3: How to check the correctness?

II. The Analysis of Problem

2.1 *Set the eigenvalues*

How to set the eigenvalues?

- In the initial modeling, first of all we use programming to count the number of authors, mail pieces, addresses and some other information. Select some of the language of the message as a characteristic attribute, including the length of the mail, mail frequency, vocabulary richness and so on. Then we get the author's eigenvalue matrix based on all mail training vector machine models sent by the author in the sample.
- In the process of perfecting the model, on the premise of preserving the content of the sample provided by the government, we take into account the structural features of the e-mail and make use of the eigenvalue matrix to form the eigenvalues of the e-mail and further integrate it into the eigenvalue matrix of the author.

2.2 *Training feature values*

How to get the eigenvalue training?

- The training samples were imported into LibSVM software for training. Based on the theory of multi-classification vector machine, the corresponding eigenvalue model was obtained by polynomial function, two-layer perceptron function (sigmoid function) and RBF kernel function.

2.3 *Detection of eigenvalues*

How to check the correctness?

- Again using LibSVM software to detect samples from the sample inside the sample eigenvalue matrix into the software for comparison, and ultimately find the corresponding author parameters, the results returned to complete the author attribution problem.

III. Models

3.1 Basic Model

3.1.1 Terms, Definitions and Symbols

w	Normal vector of hyperplane
x_i	Training samples
A,B	Positive parameter
$n_d(w_k)$	The number of terms in the collection
$f(w_k)$	Frequency of occurrence of word w
$R(W_K)$	The fixed vector of weight of importance

3.1.2 Assumptions

- Everyone has only a mailbox.
- Hypothesis tests assume the probability distributions of known types of different authors. If different types of words in a text are counted, the natural distribution is a multinomial distribution.

3.1.3 The Foundation of Model

Support Vector Machines (SVMs) recently gained pop- clarity in the learning community[4]. In its simplest linear form, an SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum interclass distance, the *margin*. Figure 1 shows such a hyper plane with the associated margin.

The formula for the output of a linear SVM is

$$u = w * x + b \tag{1}$$

where w is the normal vector to the hyperplane, and x is the input vector. The margin is defined by the distance of the hyperplane to the nearest of the positive and negative examples. Maximizing the margin can be The constraints of the hyperplane are as follows:

$$y_r = \left[\left(w * x_r + b\right)\right] r = 1,2,3,\ldots\ldots,n$$

(2)

The constraint distance from the sample point to the hyperplane is as follows: the constraint distance from the sample point to the hyperplane is as follows:

$$d = \frac{|w * x + b|}{||w||}$$

(3)

In order to separate the two kinds of samples correctly and achieve the goal of maximizing the classification interval, we must get such a classification line, that is, the optimal classification line. If it is not classified in the plane, but in the high dimensional space, the optimal classification surface is to be obtained. As shown in the figure, star and circle represent two different types of data, w*x+b=0 is classification line, two kinds of samples in the distance classification line nearest point respectively the average line parallel to the classification line, then the distance between the two lines:

$$d_{ab} = \min_{\{x,y\}=1} \frac{|w * x + b|}{||w||} + \min_{\{x,y\}=1} \frac{|w * x + b|}{||w||}$$

(4)

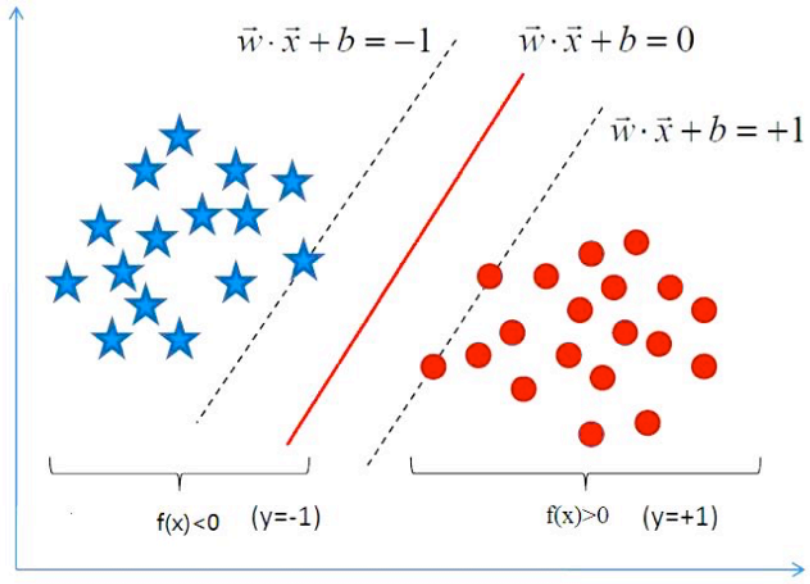

Figure 1. SVM. Hyperplane with maximal margin generated by a linear

3.1.4 Solution and Result

1.Support Vector Machines

Where x is the i-th training example and $y_i \in \{-1, 1\}$ is the correct output of the SVM for the i-th training example. Note that the hyperplane is only determined by the training instances xi on the margin, the support vectors. Of course, not all problems are linearly separable. Cortes and Vapnik [1] proposed a modification to the optimization formulation that allows, but penalizes, examples that fall on the wrong side of the decision boundary.

Support vector machines are based on the structural risk minimization principle[34]from computational learning theory. The idea is to find a model for which we can guarantee the lowest true error. This limits the probability that the model will make an error on an unseen and randomly selected test example.

An SVM finds a model whichminimizes(approximately)a bound on the true error by controlling the model complexity (VC-Dimension). This avoids over-fitting, which is the main problem for other semi-parametric models.

The SVM can be extended to a non-linear model by mapping the input space to a very high-dimensional feature space of the first experience choice. The optimal separation hyperplane in this space is constructed. The constraint of the hyperplane and the constraint distance of the hyperplane of the sample point are given by(), and mapping is given:

$$x \to z = \phi(x)$$

So for a given linear class

$$u = s * z + b$$

can be learned by the parameter s. But note that this algorithm uses dot product. Therefore, there is no need to calculate the high-dimensional values z and s, just the product

$$k(x, w) := \phi(x) * \phi(w) = s * z \tag{5}$$

This can be re-entered in the space identified, generic kernel example:

Polynomial

$$h(x) = \operatorname{sgn}(\sum_{i=1}^{n} \alpha_i y_i k(x_i + x) + b) \tag{6}$$

Sigmoid $$k(x,y)+\tanh(\kappa(xy')+\Theta)$$ (7)

RBF $$k(x,y)=e^{-\gamma\|x-y\|^2},\gamma>0$$ (8)

If it is judged that the following function value is greater than 0, the new vector x is classified into category 1

$$h(x)=\mathrm{sgn}(\sum_{i=1}^{n}\alpha_i y_i k(x_i+x)+b)$$ (9)

Only the α_i corresponding to the support vectors Wi in the training set, i.e. the examples on the margin, are different from 0. If we rewrite $h(x)=\mathrm{sgn}(w*x)$ and select some $\gamma>0,\sigma\in(0,1)$ then for all distributions generating the data the following inequality holds with probability $\geq 1-\sigma$ over the training patterns:

$$TestError\leq v+\sqrt{\frac{c}{\ell}(\frac{R^2\Lambda^2}{\gamma^2}\log^2\ell+\log\frac{1}{\delta})}$$ (10)

Where ||x||≤R, ||w||≤Λ, the term υ is the fraction of training samples with margin $y_i(wx_i)$ and c is a large constant. Therefore the inputs x should all have a comparable length ||x||.

Training SVMs in general requires solutions to quadratic programming (QP) problems. Any QP optimization method can be used to learn and b training example. However, many QP methods can be very slow for big problems. We use the method implemented by Joachims [1], which is especially useful for text categorization because it uses the sparse representation of input. Once weights have been learned, the new item x is classified by calculating h (x) as defined in (6). The unique advantage of support vector machines in text typing is the ability to handle thousands of different inputs. This provides an opportunity to use all the words in the text directly as features. The number of occurrences for each word record. Usually a corpus contains more than a few different words, each covering only a small portion. We categorized documents into

different thematic categories, and as a function, he used stemming to create statistically significant features.

3.2 Transformations of Frequency Vectors

3.2.1 Normalization of Length

Because of the rather uneven frequency distribution of text in people, in Zip's law where f (r) is the frequency of the level r term in the text, A and B are positive parameters[3].

$$f(r) = (\frac{\mathrm{A}}{\mathrm{B}+r})^{\frac{1}{\gamma-1}} \tag{11}$$

Frequency vectors of different lengths must be normalized to the standard length. From the SVM point of view, the best specifications are:

$$d_i^* = \frac{d_i}{\left\| d_i \right\|_t} \tag{12}$$

This is the common word frequency conversion SVM applies to text.

3.2.2 Importance Weights

The importance weight is often used to reduce the learning task dimension. Feature extraction in text retrieval often takes into accounting the reduction of the dimensionality of the input space. But SVM can manage a large number of dimensions. Therefore, a reduction in dimensions is not necessary, and weight of importance can be used to quantify the importance of a given type in documents collected in the text. Types that are evenly distributed across a collection of documents will be given a low weight of importance because they are judged less specific for the documents that appear in them than just the types used in a few documents.

The weight of one type of importance is multiplied by the frequency of occurrence of its conversion. Therefore, each different important weight

can be combined with each frequency transform described above. We examined the document's redundancy, redundancy to quantify the skewness of probability distributions. We consider the empirical distribution of each type on different documents and define a weight of importance[2]:

$$R(w_k) = M_{\max} - H = \log n + \sum_{i=1}^{n_d} \frac{f(w_k, d_i)}{f(w_k)} \log \frac{f(w_k, d_i)}{f(w_k)} \tag{13}$$

Where is $f(w_k, d)$ the frequency of appearance of the item, and n is the number of documents in the training set. A fixed vector containing all types of importance weights appearing in the training set is formed.

3.2.3 Data

For our experiments, we used Enron email analysis at the University of California, Berkeley, and these options were chosen on a semi-motivated basis (focusing on business-related emails and the California energy crisis and later emails that occurred during the collection. By counting the 1702 emails sent from 175 addresses, we counted all email addresses and the email closures they originated from that address (all have Java implementations, the code is in the appendix) and have a LibSVM implementation.

Based on the SVM's ability to manage a large number of dimensions, we analyzed the emails provided by the website using Enron email, categorized the email content and extracted the features, which were then transferred to a feature matrix.

Using java we can count how many emails are sent to each email and give each number, as shown below. And I do you statistics corresponding to the corresponding number of the e-mail address of the word frequency, as the file too much, put in the attachment

NO	Number of messages	mailbox
0	1000	steven.kean@enron.com
1	4	drew.fossum@enron.com
2	14	legalonline-compliance@enron.com
3	5	james.steffes@enron.com
4	2	gramlr@pjm.com
5	16	jeff.dasovich@enron.com
6	10	mary.hain@enron.com
7	4	terri_l_ponce_de_leon@calpx.com
8	3	mona.petrochko@enron.com
9	1	steven@iepa.com
10	31	miyung.buster@enron.com
11	12	susan.mara@enron.com
12	5	rcarroll@bracepatt.com
13	1	douglass@arterhadden.com
14	2	gfergus@brobeck.com
15	1	dwatkiss@bracepatt.com
16	20	alan.comnes@enron.com
17	1	foothill@lmi.net
18	3	karen.denne@enron.com
19	2	paul.kaufman@enron.com
20	1	dan.leff@enron.com
21	11	jmunoz@mcnallytemple.com
22	1	jmball@ns.net
23	1	henry.means@enron.com
24	1	sandi_j._thompson@calpx.com
25	2	ann.schmidt@enron.com
26	1	hap.boyd@enron.com
27	1	jennifer.thome@enron.com
28	1	davidtaylorsf@aol.com
29	89	john.shelk@enron.com
30	1	mkahl@ka-pow.com
31	1	angela.wilson@enron.com
32	4	foothil9@idt.net
33	2	joseph.alamo@enron.com
34	2	m..schmidt@enron.com
35	1	robert.frank@enron.com
36	6	d..steffes@enron.com
37	4	mike.mcconnell@enron.com
38	1	john.nowlan@enron.com

Fig 2

And can be programmed to obtain eigenvalues.

```
------------------------------------------------------------------------------------------------------------------------
mailbox:drew.fossum@enron.com
no:1
Total number of emails sent: 4
1 -10:220 -9:1 -8:0 -7:5 -6:0 -5:9 -4:25 -3:5 -2:21 -1:122 1:2 26:2 43:2
1 -10:342 -9:7 -8:26 -7:9 -6:0 -5:14 -4:36 -3:29 -2:38 -1:193 1:1 6:1 8:1 9:2 26:2 27:1 43:2
1 -10:326 -9:7 -8:26 -7:9 -6:0 -5:14 -4:36 -3:29 -2:20 -1:193 1:2 8:2 9:2 26:2 30:2
1 -10:209 -9:1 -8:0 -7:5 -6:0 -5:9 -4:25 -3:5 -2:9 -1:122 1:1 26:1 30:1 43:1

------------------------------------------------------------------------------------------------------------------------
mailbox:legalonline-compliance@enron.com
no:2
Total number of emails sent: 14
2 -10:290 -9:6 -8:0 -7:4 -6:0 -5:13 -4:34 -3:41 -2:39 -1:152 1:2 16:2 27:2
2 -10:290 -9:6 -8:0 -7:4 -6:0 -5:13 -4:34 -3:41 -2:39 -1:152 1:2 16:2 27:2 30:1
2 -10:290 -9:6 -8:0 -7:4 -6:0 -5:13 -4:35 -3:41 -2:39 -1:152 1:2 27:2 30:2 43:2
2 -10:291 -9:6 -8:0 -7:4 -6:0 -5:14 -4:34 -3:41 -2:39 -1:153 1:1 9:1 16:1 27:1
2 -10:290 -9:6 -8:0 -7:4 -6:0 -5:13 -4:35 -3:41 -2:39 -1:152 1:1 8:1 9:1 27:1
2 -10:334 -9:7 -8:0 -7:8 -6:0 -5:14 -4:40 -3:30 -2:15 -1:170 1:1 16:1 27:1
2 -10:334 -9:7 -8:0 -7:8 -6:0 -5:14 -4:40 -3:30 -2:15 -1:170 1:2 27:2 43:1
2 -10:333 -9:7 -8:0 -7:8 -6:0 -5:13 -4:40 -3:31 -2:15 -1:169 1:1 16:1 27:1
2 -10:334 -9:7 -8:0 -7:8 -6:0 -5:14 -4:40 -3:30 -2:15 -1:170 1:1 16:1 27:1
2 -10:290 -9:6 -8:0 -7:4 -6:0 -5:13 -4:34 -3:41 -2:39 -1:152 1:2 27:2
2 -10:285 -9:6 -8:0 -7:4 -6:0 -5:12 -4:34 -3:41 -2:35 -1:151 1:1 16:1 27:1
2 -10:337 -9:7 -8:0 -7:8 -6:0 -5:14 -4:40 -3:31 -2:15 -1:173 1:1 16:1 27:1
2 -10:297 -9:6 -8:0 -7:4 -6:0 -5:14 -4:37 -3:41 -2:19 -1:152 1:1 8:1 9:1 16:1 27:1
2 -10:332 -9:7 -8:0 -7:8 -6:0 -5:14 -4:40 -3:30 -2:20 -1:170 1:2 9:1 21:1 27:1

------------------------------------------------------------------------------------------------------------------------
mailbox:james.steffes@enron.com
no:3
Total number of emails sent: 5
3 -10:2591 -9:41 -8:24 -7:59 -6:1 -5:122 -4:245 -3:221 -2:371 -1:796 1:2 9:2
3 -10:115 -9:0 -8:0 -7:1 -6:0 -5:5 -4:10 -3:7 -2:16 -1:76 1:2 26:2 43:1
3 -10:782 -9:14 -8:4 -7:13 -6:0 -5:25 -4:86 -3:54 -2:112 -1:309 1:2 22:1 24:1 26:1 28:1 43:2 45:1
3 -10:154 -9:6 -8:24 -7:0 -6:0 -5:6 -4:20 -3:37 -2:24 -1:94 1:2 8:2 10:2 20:2 26:1
3 -10:381 -9:2 -8:0 -7:6 -6:3 -5:18 -4:43 -3:2 -2:43 -1:178 1:1 22:1 24:1 27:1

------------------------------------------------------------------------------------------------------------------------
mailbox:gramlr@pjm.com
no:4
Total number of emails sent: 2
4 -10:249 -9:8 -8:4 -7:6 -6:0 -5:18 -4:25 -3:11 -2:31 -1:155 1:2 8:2 9:2 29:2
4 -10:639 -9:7 -8:4 -7:8 -6:0 -5:35 -4:86 -3:10 -2:65 -1:318 4:1 8:1 9:1
```

Fig 3

And LibSVM gets the training value

```
0 -10:208 -9:16 -8:72 -7:2 -6:1 -5:8 -4:26 -3:60 -2:56 -1:91 1:1 13:1 20:1 23:1
0 -10:369 -9:10 -8:48 -7:2 -6:0 -5:12 -4:45 -3:43 -2:56 -1:196 1:1 8:1 9:1 20:1 23:1 24:1 43:1
0 -10:140 -9:0 -8:1 -7:6 -6:0 -5:3 -4:15 -3:2 -2:11 -1:87 1:2 21:2 26:2
0 -10:72 -9:0 -8:0 -7:3 -6:0 -5:3 -4:5 -3:0 -2:6 -1:55 1:2 4:2 24:2 26:2
0 -10:849 -9:6 -8:27 -7:9 -6:0 -5:32 -4:76 -3:82 -2:102 -1:320 1:2 8:2 11:2 21:2 26:1 27:2 44:1
0 -10:122 -9:4 -8:0 -7:4 -6:0 -5:8 -4:16 -3:12 -2:22 -1:74 1:1 25:1 26:1
0 -10:1692 -9:8 -8:5 -7:60 -6:1 -5:86 -4:205 -3:81 -2:207 -1:702 1:1 4:1 8:2 9:2 11:2 21:2 22:1
0 -10:225 -9:7 -8:64 -7:1 -6:0 -5:13 -4:21 -3:40 -2:29 -1:108 1:1 3:1 9:2 20:2 26:2
0 -10:343 -9:4 -8:0 -7:7 -6:0 -5:11 -4:34 -3:43 -2:36 -1:194 1:2 8:2 9:2 20:1 26:2
0 -10:354 -9:5 -8:0 -7:5 -6:2 -5:9 -4:35 -3:24 -2:35 -1:187 1:2 20:1 22:2 24:2 26:2
0 -10:204 -9:5 -8:1 -7:6 -6:0 -5:8 -4:25 -3:12 -2:26 -1:113 1:2 8:2 9:2 26:1 29:2
0 -10:1360 -9:21 -8:18 -7:48 -6:1 -5:54 -4:137 -3:52 -2:148 -1:503 1:2 8:2 9:2 21:2 38:1
0 -10:55 -9:5 -8:4 -7:0 -6:0 -5:1 -4:5 -3:24 -2:12 -1:32 1:2 9:1 20:2 21:1
0 -10:65 -9:4 -8:0 -7:0 -6:0 -5:1 -4:6 -3:12 -2:18 -1:35 1:2 20:2 29:2
0 -10:167 -9:6 -8:0 -7:0 -6:1 -5:5 -4:19 -3:12 -2:25 -1:98 1:1 9:1 22:1
0 -10:736 -9:19 -8:52 -7:4 -6:0 -5:33 -4:79 -3:103 -2:116 -1:323 1:1 8:1 9:1 21:1 24:1 26:1
0 -10:1791 -9:9 -8:3 -7:102 -6:0 -5:86 -4:177 -3:75 -2:165 -1:702 1:2 9:2 26:2
0 -10:1296 -9:16 -8:48 -7:0 -6:3 -5:50 -4:140 -3:274 -2:205 -1:550 1:2 9:1 12:1 26:2
0 -10:476 -9:10 -8:4 -7:15 -6:0 -5:30 -4:56 -3:37 -2:54 -1:247 1:2 8:2 9:2 25:1 30:2 43:2
0 -10:246 -9:3 -8:0 -7:5 -6:2 -5:20 -4:30 -3:13 -2:22 -1:141 1:2 32:2
0 -10:943 -9:50 -8:148 -7:18 -6:0 -5:66 -4:78 -3:208 -2:168 -1:350 1:2 9:2 13:1 26:2
0 -10:103 -9:6 -8:24 -7:0 -6:0 -5:2 -4:8 -3:24 -2:20 -1:66 1:2 20:2 23:2 26:2
0 -10:301 -9:9 -8:0 -7:3 -6:0 -5:10 -4:33 -3:24 -2:50 -1:138 1:1 9:1 26:1 44:1
0 -10:241 -9:4 -8:0 -7:2 -6:0 -5:8 -4:26 -3:18 -2:42 -1:130 1:1 9:1 26:1
0 -10:532 -9:11 -8:3 -7:21 -6:5 -5:24 -4:67 -3:24 -2:68 -1:275 1:2 8:2 9:2 26:2
0 -10:524 -9:15 -8:48 -7:3 -6:0 -5:17 -4:48 -3:76 -2:61 -1:222 1:2 8:2 9:2 26:2
0 -10:509 -9:11 -8:3 -7:21 -6:5 -5:24 -4:67 -3:24 -2:43 -1:275 1:2 9:2 25:1 26:1
0 -10:268 -9:5 -8:0 -7:8 -6:0 -5:12 -4:31 -3:7 -2:22 -1:138 1:2 30:1 32:2
0 -10:205 -9:6 -8:24 -7:5 -6:0 -5:11 -4:17 -3:40 -2:28 -1:116 1:2 20:2 26:2 43:2
0 -10:69 -9:2 -8:0 -7:1 -6:0 -5:3 -4:9 -3:16 -2:7 -1:51 1:2 4:2 29:2 31:2
0 -10:4 -9:0 -8:0 -7:0 -6:0 -5:0 -4:2 -3:0 -2:3 -1:4 1:2
0 -10:46 -9:0 -8:0 -7:0 -6:0 -5:1 -4:6 -3:0 -2:5 -1:38 1:2 25:2
0 -10:136 -9:8 -8:40 -7:2 -6:0 -5:8 -4:14 -3:20 -2:25 -1:86 1:2 8:2 9:2 20:2 21:2 22:1
0 -10:491 -9:17 -8:48 -7:2 -6:0 -5:14 -4:62 -3:64 -2:83 -1:224 1:2 8:2 9:2 22:1 28:1
0 -10:532 -9:6 -8:24 -7:12 -6:0 -5:18 -4:52 -3:34 -2:58 -1:225 1:2 9:2 28:2
0 -10:1189 -9:6 -8:24 -7:26 -6:1 -5:47 -4:102 -3:24 -2:101 -1:461 1:2 9:2 29:2 41:2
0 -10:3795 -9:29 -8:100 -7:24 -6:0 -5:75 -4:138 -3:325 -2:229 -1:603 1:1 8:1 9:1 22:1 29:1
0 -10:82 -9:0 -8:0 -7:2 -6:0 -5:5 -4:8 -3:0 -2:7 -1:51 1:2 22:2 29:1
0 -10:397 -9:27 -8:64 -7:5 -6:3 -5:17 -4:34 -3:78 -2:61 -1:187 1:2 9:2 22:2
0 -10:891 -9:7 -8:24 -7:4 -6:1 -5:65 -4:107 -3:100 -2:93 -1:387 1:2 9:2 21:2 22:2 27:2 29:2
0 -10:255 -9:5 -8:26 -7:3 -6:0 -5:12 -4:22 -3:67 -2:32 -1:143 1:2 8:2 25:2
0 -10:1189 -9:18 -8:48 -7:24 -6:3 -5:69 -4:105 -3:133 -2:121 -1:451 1:1 8:1 9:1 22:1
0 -10:60 -9:0 -8:1 -7:1 -6:0 -5:4 -4:5 -3:0 -2:5 -1:41 1:2 23:2
0 -10:2780 -9:41 -8:102 -7:36 -6:7 -5:131 -4:284 -3:153 -2:316 -1:1032 1:2 8:2 9:2 22:2 24:2 45:2 4
0 -10:702 -9:20 -8:73 -7:12 -6:0 -5:20 -4:100 -3:116 -2:99 -1:324 1:2 5:2 9:2 28:2
0 -10:235 -9:6 -8:0 -7:3 -6:0 -5:10 -4:28 -3:12 -2:35 -1:127 1:2 4:2 8:2 9:2 22:2
```

Fig 4

We get a higher accuracy rate based on the secondary test data set.

IV. Conclusions

4.1 Conclusions of the problem

- Article discusses some work on the identity of the author, and introduces the support vector machine into a new method. Especially suitable for this task, because the feature space has a very high dimension, most of the features carry important information, and the data of the specific instance is sparse. The experimental results show that the support vector machine (SVM) has always achieved good performance in the recognition task.
- There is no need to select specific features, and we can simply take all the word frequency. In addition, the preprocessing and weighting of the features are not important, and many methods lead to almost the same results. Support vector machine based on author attribution and text mining can process large amounts of length documents and large amounts of text data, so support vector machine is the preferred method of current author attribution.
- The functional words studied in this paper are enough to be used to identify. The form of functional words is not good enough than the form of words. We need to further explore if labels or other structural features can be used as "no content" features for authorship identification.

4.2 Roles of our model

- Better to estimate a person's writing characteristics
- Find the author through notes when conditions permit
- It provides a practical solution to many practical problems (such as criminal analysis)

V. Future Work

5.1 Model improvement

Because people's text frequency distribution is quite unbalanced, A and B are positive parameters in Zipf's law, but the word frequency difference is large in the text, and few units are highly specific. Therefore, the generalized form of Zipf's law is as follows:

$$f(r)=(\frac{\mathrm{A}}{\mathrm{B}+r})^{\frac{1}{\gamma-1}}$$

It will be more realistic; the optimization result is better.

5.2 Model Application

The complexity of the support vector machine has nothing to do with the dimension of the sample, and the learning efficiency and accuracy are higher. It is suitable for the dataset of high-dimensional stylistic features and is therefore one of the most commonly used algorithms for author identification.

VI. References

[1] T. Joachims, “Making large-scale SVM learning practical,” Technical Report, Uni Dortmund, 1998.

[2] G. Salton and C. Buckley, “Term weighting approaches in auto-matic text retrieval,” Information Processing and Management, vol. 24, pp. 513–523, 1988

[3] R.J.Chitashvili and R.H.Baayen, “Word frequency distributions,” in Quantitative Text Analysis, edited by G. Altmann and L.Hřebíček, wvt,Trier,1993, pp. 46–135.

[4] V.N.Vapnik, Statistical Learning Theory, Wiley: New York,1998.

Appendix

Show.java

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.util.ArrayList;
import java.util.List;

public class Show {
	public static void main(String[] args) {
		List<People> listOfPeople=new ArrayList<People>();

		listOfPeople=getList();
		//Show show=new Show();

		Analyze a=new Analyze();
		//a.showList(listOfPeople);
		//a.showEmailMata(listOfPeople);
		//a.showData(listOfPeople);
		a.showTrainData(listOfPeople);
	}

	public static  List<People> getList() {
		File file = new File("listOfPeople.dat");
		FileInputStream in;
		List<People> list = null;
		try {
			in = new FileInputStream(file);
			ObjectInputStream objIn = new ObjectInputStream(in);
			list = (List<People>) objIn.readObject();

			objIn.close();
		} catch (IOException e) {
			e.printStackTrace();
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}

		return list;
	}

}
```

People.java

```
import java.io.IOException;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class People implements Serializable {
        String name;
        Integer lableNumber;
        List<Email> emailList =new ArrayList<Email>();
        Integer numberOfEmail=0;

        public People(String name){
                this.name=name;
        }

        /**
         *
         *
         * @param email
         */
        public void addEmail ( String email,String path)throws IOException {
                Email peopleEmail;
                peopleEmail=new Email(email,name,lableNumber,path);
                peopleEmail.createMata(path);
                peopleEmail.createEmailText();
                peopleEmail.createOtherFeature();
                //peopleEmail.showEmailValue();
                emailList.add(peopleEmail);
        }

        public void setLableNumber(int lable){
                this.lableNumber=lable;
        }
}
```

Analyze.java

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class Analyze {
        public static void main(String[] args)throws
FileNotFoundException, IOException {
                List<People> listOfPeople=new ArrayList<People>();
                String[][] textsLists=new String[8][];
                String[] textList;
                Integer[] numberOfPeople=new Integer[8];

                for(int i=0;i<8;i++){

                        int k=i+1;
                        textsLists[i]=fileList("data/"+k+"/txt");
                }

                for(int m=1;m<=8;m++){

                        textList=textsLists[m-1];
                        for (int i = 0; i < textList.length; i++)
{
                                Boolean ifAdd = true;
                                String peoplename = null;
                                peoplename = openFile("data/"+m+"/
txt/"+textList[i]);

                                for (int j = 0; j <
listOfPeople.size(); j++) {
                                        //判断是否重复
                                        People p =
listOfPeople.get(j);

                                        if
(peoplename.equals(p.name)) {

                                                ifAdd = false;

listOfPeople.get(j).numberOfEmail++;

listOfPeople.get(j).addEmail(textList[i],"data/"+m+"/cats/");

                                                break;
                                        }
                                }

                                if (ifAdd) {

                                        People person = new
People(peoplename);

person.addEmail(textList[i],"data/"+m+"/cats/");
```

```
person.setLableNumber(listOfPeople.size());
                                        person.numberOfEmail++;
                                        listOfPeople.add(person);

                                }
                        }

                        if(m==1)

numberOfPeople[m-1]=listOfPeople.size();
                        else {

numberOfPeople[m-1]=listOfPeople.size()-numberOfPeople[m-2];
                        }
                }

                //showEmailMata(listOfPeople);
                //showList(listOfPeople);
                //showData(listOfPeople);
                createPeoleListText(listOfPeople);
                savePeopleList(listOfPeople);
                //createPeoleListText(listOfPeople);

        }

        /**
         *
         *
         * @param peopleList
         */
        public static void showEmailMata(List<People> peopleList){
                for(int i=0;i<peopleList.size();i++){

System.out.println(peopleList.get(i).name);
                        for(int j = 0;
j<peopleList.get(i).emailList.size(); j++){

System.out.println(peopleList.get(i).emailList.get(j).fileName);
                                Email e =
peopleList.get(i).emailList.get(j);

                                List mata=null;
                                mata=e.mata;

                                for(int k=0;k<mata.size();k++){

System.out.println(mata.get(k));
                                }

                        }
                }

System.out.println("-----------------------------------------------
```

```
----------------------------------------------------------------");

System.out.println("-----------------------------------------------
----------------------------------------------------------------");
        }

        /**
         *
         *
         * @param peopleList
         */
        public static void showList(List<People> peopleList){
                int number=0;
                System.out.println("Number  "+"NumnerOfEmail
"+"People");
                for(int n=0;n<peopleList.size();n++){

number=peopleList.get(n).numberOfEmail+number;

System.out.println(peopleList.get(n).lableNumber+"
"+peopleList.get(n).numberOfEmail+"     "+peopleList.get(n).name);
                }
                System.out.println(number);

System.out.println("-----------------------------------------------
----------------------------------------------------------------");

System.out.println("-----------------------------------------------
----------------------------------------------------------------");
        }

        /**
         *
         *
         * @param peopleList
         */
        public static void showData (List<People> peopleList){
                System.out.println("总邮箱地址
数："+peopleList.size());
                for(int i=0;i<peopleList.size();i++){
                        People person=peopleList.get(i);


                        System.out.println("邮箱:"+person.name+"\n
编号:"+person.lableNumber+"\n共发送邮件数："+person.numberOfEmail);
                        List<Email> personEmail=person.emailList;
                        for(int j=0;j<personEmail.size();j++){
                                Email email= personEmail.get(j);

System.out.print(person.lableNumber+" ");
                                email.showEmailValue();
                        }
                        System.out.println();
```

```
System.out.println("-----------------------------------------------
------------------------------------------------------------------
--");
                }
        }

        /**
         *
         *
         * @param peopleList
         */
        public static void showTrainData (List<People> peopleList){
                //
System.out.println("totalnumber: "+peopleList.size());
                for(int i=0;i<peopleList.size();i++){
                        People person=peopleList.get(i);
                        //System.out.println("邮
箱:"+person.name+"\n编号:"+person.lableNumber+"\n共发送邮件
数: "+person.numberOfEmail);
                        List<Email> personEmail=person.emailList;
                        for(int j=0;j<personEmail.size();j++){
                                Email email= personEmail.get(j);

System.out.print(person.lableNumber+" ");
                                email.showEmailValue();
                        }
                        //System.out.println();
                        //
System.out.println("-----------------------------------------------
------------------------------------------------------------------
--");
                }
        }

        /**
         *
         *
         * @param peopleList
         */
        public static  void savePeopleList(List<People> peopleList)
{
                try {
                        File f=new File("listOfPeople.dat");
                        if(f.exists()){
                                f.delete();
                        }
                        ObjectOutputStream oos = new
ObjectOutputStream(new FileOutputStream("listOfPeople.dat", true));
                        oos.writeObject( peopleList);
                        oos.flush();
                        oos.close();
                } catch (FileNotFoundException e) {
```

```
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

        public static void createPeoleListText(List<People>
peopleList) throws  IOException{
                for(int i=0;i<peopleList.size();i++) {
                        File detail = new File("emailDetails/" +
peopleList.get(i).lableNumber+"--"+peopleList.get(i).name+".txt");
                        FileWriter fw = new FileWriter(detail);
                        BufferedWriter bw = new
BufferedWriter(fw);
                        People people = peopleList.get(i);
                        List<Email> emailList=people.emailList;

                        bw.write("Name:"+people.name+"
"+"Numbering:"+people.lableNumber+"        "+"The total number of
messages:"+people.numberOfEmail);
                        bw.newLine();

bw.write("-----------------------------------------------------------
---------------------------");
                        for(int j=0;j<emailList.size();j++){

bw.write("-----------------------------------------------------------
---------------------------\r\n");
                                Email email=emailList.get(j);
                                bw.newLine();
                                bw.write("1.Email name:"
+email.emailName);
                                bw.newLine();

bw.write("-----------------------------------\r\n");
                                /*
                                bw.write("2.Email text:");
                            bw.newLine();
                                bw.write(email.emailText);
                                bw.newLine();

bw.write("-----------------------------------\r\n");
                                */
                                bw.write("2.Word frequency:
"+"total words"+email.totalNumberOfWords);
                                bw.newLine();
                                String wordsFr=new String();
                                Map<String,Integer>
words=email.wordsMap;
                                int k=11;
                                for (Map.Entry<String, Integer>
```

```
entry : words.entrySet()) {
                                                bw.write(entry.getKey()
+":"+entry.getValue()+"            ");
                                                if(k%10==0){
                                                        bw.newLine();
                                                }
                                                k++;
                                        }
                                        bw.newLine();

bw.write("-----------------------------------\r\n");
                                        bw.write("3.Other feature:\r\n"+
"totalNumberOfWords:"+email.totalNumberOfWords+"
numberOfShortWord:"+email.numberOfShortWord+"
vocabularyRichness:"+email.vocabularyRichness+"\r\nnumberOfline:"+em
ail.numberOfline+"       numberOfPeriod:"+email.numberOfPeriod+"
numberOfSemicolon:"+email.numberOfSemicolon+"\r\nnumberOfSuotation:"
+email.numberOfSuotation+"       numberOfDash:"+email.numberOfDash+"
numberOfColon:"+email.numberOfColon);
                                        bw.newLine();

bw.write("-----------------------------------------------------------
---------------------------");
                                }
                                bw.close();
                        }

                }

                /**
                 *
                 *
                 * @param p
                 */
                public static String[] fileList (String p)throws
FileNotFoundException, IOException {
                        String [] filelist=null;
                        File allfile;

                        allfile=new File(p);
                        if(allfile.exists())
                                filelist=allfile.list();

                        return filelist;
                }

                /**
                 *
                 *
                 * @param filename
                 */
                public static String openFile(String filename)throws
IOException{
                        String name;
```

```
                File file =new File(filename);
                BufferedReader br=new BufferedReader(new
FileReader(file));
                br.readLine();
                br.readLine();
                name=br.readLine();
                name=name.substring(6);

                br.close();

                return name;
        }

}
```

Email.java

```
import java.io.*;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Email implements Serializable {
        //private final int FIRST = 7;
        //private final int SECOND = 20;
        //private final int THIRD = 33;
        //private final int FOUTH = 10;

        String emailName =null;
        List mata=new ArrayList();
        String fileName;
        String peopleName;
        Integer peopleLable;
        String filePath;
        Integer[] emailValue=new Integer[53];
        String emailText=new String();

        Map<String,Integer> wordsMap;

        int totalNumberOfWords=0;

        int numberOfShortWord=0;
        int numberOfDigits=0;

        int vocabularyRichness=0;

        int numberOfline=0;
        //int numberOfBlankLine=0;

        //int numberOfparagraph=0;

        int numberOfPeriod=0;
        int numberOfSemicolon=0;
        int numberOfSuotation=0;
        int numberOfDash=0;
        int numberOfColon=0;

        public Email(String email,String name,Integer lable,String
path){
                for(int i=0;i<53;i++){
                        emailValue[i]=0;
                }
                peopleName=name;
                peopleLable=lable;
                fileName=email;
                filePath=path;
                this.emailName
=email.substring(0,email.length()-4);
        }

        /**
```

```
         *
         *
         */
        public void createEmailText()throws IOException{
                filePath
=filePath.substring(0,filePath.length()-6)+"/txt/";
                File email=new File(filePath+fileName);
                BufferedReader br=new BufferedReader(new
FileReader(email));
                emailText=new String();
                String line;
                Boolean ifEmailText=false;
                Pattern p = Pattern.compile("X-FileName:");
                Matcher m;

                line=br.readLine();
                while(line!=null){
                        if(!ifEmailText) {
                                m = p.matcher(line);
                                if (m.find()){
                                        ifEmailText=true;
                                }
                        }else {
                                emailText=line+"\n"+emailText;
                                numberOfline++;
                        }
                        line=br.readLine();
                }

                br.close();
        }

        /**
         *
         *
         */
        public void createOtherFeature (){

                numberOfShortWord=regularMatch("and|And|to|To|But|
but|an|An|so|So");
                numberOfDigits=regularMatch("\\d");

                numberOfPeriod=regularMatch("\\.");
                numberOfSemicolon=regularMatch(";");
                numberOfSuotation=regularMatch("\"|\'");
                numberOfDash=regularMatch("--");
                numberOfColon=regularMatch(":");

                differentWord(emailText);

                //numberOfparagraph=(regularMatch(" ")-
totalNumberOfWords)/2;
        }
```

```
        private void  differentWord(String str){
                wordsMap=new HashMap<String,Integer>();
                String[] arrys=str.split("[,.\\s]");
                for(int i=0;i<arrys.length;i++){

                        if(wordsMap.containsKey(arrys[i])){
                                wordsMap.put(arrys[i],
wordsMap.get(arrys[i])+1);
                        }else{
                                wordsMap.put(arrys[i],1);
                        }
                }

                Set<Map.Entry<String,Integer>>
set=wordsMap.entrySet();
                for(Map.Entry<String,Integer> se:set){
                        totalNumberOfWords=se.getValue()
+totalNumberOfWords;
                }

                vocabularyRichness=wordsMap.size();
        }

        /**
         *
         *
         */
        private  int regularMatch(String match){
                Pattern n = Pattern.compile(match);
                Matcher w=n.matcher(emailText) ;
                int count=0;
                while(w.find()){
                        count++;
                }
                return count;
        }

        /**
         *
         *
         * @param path
         */
        public void createMata(String path)throws IOException{
                File f=new File(path+ emailName +".cats");
                BufferedReader br=new BufferedReader(new
FileReader(f));
                String[] stringLine=new String[3];

                String line=br.readLine();
                while (line!=null){
                        mata.add(line);
                        stringLine=line.split(",");
                        //System.out.println(stringLine[0]+"
"+stringLine[1]+"  "+stringLine[2]);
```

```
                        line=br.readLine();
                }

                br.close();
        }

        /**
         *
         *
         * @param line
         */
        public void setEmailValue(String[] line){
                Integer[] integerLine=new Integer[3];
                integerLine[0]=Integer.valueOf(line[0]).intValue();
                integerLine[1]=Integer.valueOf(line[1]).intValue();
                integerLine[2]=Integer.valueOf(line[2]).intValue();
                switch (integerLine[0]){
                        case 1:
emailValue[integerLine[1]]=integerLine[2];
                                break;
                        case 2:
emailValue[integerLine[1]+7]=integerLine[2];
                                break;
                        case
3:emailValue[integerLine[1]+20]=integerLine[2];
                                break;
                        case
4:emailValue[integerLine[1]+33]=integerLine[2];
                                break;
                }
        }

        /**
         *
         *
         */
        public void showEmailValue(){
                System.out.print("-10:"+totalNumberOfWords+"
-9:"+numberOfColon+" -8:"+numberOfDash+" -7:"+numberOfSuotation+"
-6:"+numberOfSemicolon+" -5:"+numberOfPeriod+"
-4:"+numberOfShortWord+" -3:"+numberOfDigits+" -2:"+numberOfline+"
-1:"+vocabularyRichness+" ");
                for(int i=0;i<53;i++){
                        //if(i==27)
                        //      System.out.println();
                        if(emailValue[i]!=0)

System.out.print(i+":"+emailValue[i]+" ");
                }
                System.out.println();
        }


}
```