

An Iterative Voting Method Based On Word Density For Text Classification

Jiaxun Wang

School of Software, Tsinghua University
100084 Beijing, China

jx-wang08@mails.tsinghua.edu.cn

Chunping Li

School of Software, Tsinghua University
100084 Beijing, China

cli@tsinghua.edu.cn

ABSTRACT

In this paper we present an iterative voting (IV) method using the density based weighting for text classification. An in-class word density is used to weight for each word in a topic, so that the word in documents has an array of weights to vote for given topics, and the highest scored topic will be labeled. During the voting process, the iteration strategy is applied for improving the classification effectiveness. This method shows the competitive performance against SVM, NB, KNN, and it has better time efficiency.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval: Classification and Clustering

General Terms

Algorithms, Experimentation

Keywords

Classification, Word density, Iterative voting

1. INTRODUCTION

Text classification is a foundation research area of data mining. Today more and more applications are relied on this technique. There are a series of algorithms can be used to classify documents, such as Naive Bayes (NB), K-Nearest Neighbor (KNN), and Support Vector Machine (SVM), etc [3,5,6]. The state-of-art method is SVM, which is widely used by many applications in information search and retrieval.

The architecture mostly used in text classification is preprocess-classify-evaluation [7]. In preprocessing, it first selects features from corpus, and then forms a vector for each document. The vectors are sent to classifier for classifying. The overall result was evaluated by MacroF1 and MicroF1, etc [9].

Feature selection is one of the most important parts of the above procedure. The general way of feature selection is to calculate the

tf-idf value of a feature, and select the top ranked features with higher tf-idf as dimensions of the vector, which will represent the instance. Besides tf-idf, there are also some other feature selection methods, such as Document Frequency (DF), CHI method, Information Gain (IG), and Mutual Information (MI), etc [8] [11].

But there are still some barriers we have to face. For example, these existing methods are troubled with the dimension disaster [1]. In feature selection, the complexity of classifier will increase exponentially with the number of selected features.

To avoid this case, the SVM method utilizes the support vectors and solves the dimension disaster to some extent [4]. But the time efficiency to calculate the support vector is still higher especially for the large scale of datasets.

In this paper, we propose an iterative voting method (IV) for text classification, which uses weighted word density to vote for instances. Our purpose is improve the time efficiency with respect to the limitations of existing data classification approaches, especially toward processing online, large scale of textual and web data sets. Our method is based on statistical computing with word density but not to need for dealing with feature selection in high dimensional data, different from traditional vector space model with feature weighting for textual data.

We take NB as a baseline, and also compare our iterative voting method with KNN and SVM at the same time. We choose Reuter-21578 and 20Newsgroup as our experimental datasets. The results show that the iterative voting method gets a better performance, as well as for the classification of imbalanced datasets. It runs much faster than NB, KNN and SVM. Meanwhile it performs up to SVM but better than NB and KNN in both MacroF1 and MicroF1 measurements.

The rest of this paper is organized as follows. We present the iterative voting method in details in Section 2, in which the concept of word density is introduced firstly in Subsection 2.1 and then we give the iteration voting strategy in Subsection 2.2. In Section 3, we show experiment results and have the evaluation and discussion by comparing with other methods. We have the concluding remarks in Section 4.

2. ITERATIVE VOTING METHOD

In the text classification, feature selection is very important to most classifiers. The performance of a classifier is highly dependent on its feature selection method. It is natural to compare human's manual approach for feature selection with current existing feature selection algorithms. In any ways, we think that the word is an essential element.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WIMS'11, May 25–27, 2011, Sogndal, Norway

Copyright © 2011 ACM 978-1—4503-0148-0/11/05... \$10.00

A word can affect the text categorization by its meaning, position and frequency, etc. If we have no grammar or semantic analysis, an effective way is to calculate the tf-idf values corresponding to words, which was proven to be an efficient metric for feature selection and weighting. A textual document can be identified as the words vector what it includes. Instead of dropping words, Human readers ignore some words which are considered not so important during reading. For instance, the word “war” and the word “and” in a text might be regarded as totally different weights to human readers. As we know that the word “war” appears more in military articles than in food articles. Based on these facts, we assume that (I).each word has a different weight in different topics; (II).in statistics, a certain type of articles use a certain mixture ratio of words. Weights are measured by both words’ in-class density and out-class density. While classifying, each word in the article votes its corresponding weight to each topic. And the highest scored topic will be labeled to the article.

2.1 Word Density and Weights

A word’s in-class density is defined as follows.

$$d_{in}(x) = \frac{n_t(x)}{n_t} \quad (1)$$

where x represents a word, t a topic, n_t means the number of words in topic t , and $n_t(x)$ the number of word x in topic t .

A word’s out-class density is defined as follows.

$$d_{out}(x) = \frac{n_{all}(x) - n_t(x)}{n_{all} - n_t} \quad (2)$$

where $n_{all}(x)$ means the number of the word x in all topics, and n_{all} is the number of total words in all topics.

The weight of a word is defined as follows.

$$w_t(x) = \frac{d_{in}(x)}{\alpha + d_{out}(x)} \quad (3)$$

where α is a smooth factor in case the out-class density is 0. We observed that the performance is sensitive to α , and in general it is calculated as follows.

$$\alpha = \frac{1}{n_f} \quad (4)$$

Here, n_f is the number of features in the training set.

2.2 Iterative Voting Strategy

Training process is to calculate the occurrence number of each word in a topic. During the training process, documents are scanned sequentially. For given a document, every word is recorded. If a new word appears, it is added to the feature set. If the word appeared before, it can be found in the feature set, and then we add the occurrence number of its corresponding topic.

We then have a set of features which can be calculated for their in-class densities and out-class densities in each topic. With the feature’s densities, we can get its weights according to Formula (3).

In testing process, each word in the testing document votes its corresponding weights for given topics. After voting, we choose the highest scores topic as the result. The process is shown in the following example.

Train instance:

{FOOD: milk sucks}

{ECONOMIC: milk price goes up}

After training:

Topic: {food, economic}

Features: {milk, price, go, suck, up}

Weights: described in Table 1.

Table 1. Word weights in toy example

	food	economic
Milk	1.11	0.36
Price	0	1.25
Go	0	1.25
Suck	2.5	0
Up	0	1.25

Test instance:

{ECONOMIC: milk price is high.}

So while voting, “milk” votes 1.11 for “food” and 0.36 for “economic”. “price” votes 0 for “food” and 1.25 for “economic”. Other words votes 0 (they don’t appear in the training set). The result is “food” gets 1.11 and “economic” gets 1.61. So this instance is labeled for “economic” in the right way.

Iteration is an efficient way to improve the performance of voting method. The strategy is that after voting for the first time, we get a candidate list of topics.

We cut off the last few topics according to a threshold user defined. Then we vote again with the new topic candidate list. Because some topics are eliminated at this time, $n_{all}(x)$ will be changed, and so do the out-class density and weights. The smaller the amount of topic become, the bigger their difference will be. It just likes using exclusive method to solve a choice question. This strategy helps a lot in getting a better result. The brief algorithm description of training and testing process is shown in Algorithm 1 and Algorithm 2.

Algorithm 1 Training

read in documents

count word number for each word in each topic

save the result in a table

Algorithm 2 Testing

repeat

 read in a document

 initialize the candidate topic list

 initialize the table {calculate the word density}

while number of candidate topic > 1 **do**

while document is not end **do**

 read a word

 look it up in the table

 vote corresponding weights for each topic

end while

 rank the topic by their scores

 eliminate some poor ranking topics from candidate list

 recalculate the word density

 update the table

end while

 output the result topic

until all document classified

3. EXPERIMENT AND EVALUATION

3.1 Experiment Settings

We choose Reuter-21578 and 20Newsgroup [10] as our experiment set. Reuter-21578 has 135 categories. We use “ModLewis” split but only choose positive instances in which labeling “TOPIC=YES”. There are 10,370 documents left after choosing. The biggest two topics are “acq”(2186 instances), and “earn”(3760 instances). The size of categories varies from 4 (we drop categories which size is smaller than 4 because they are too small to test) to 3760. So it is an imbalanced dataset. Training and Testing on an imbalanced dataset would bring more difficulties to classification algorithms. As some categories in dataset are too small and some are too large comparing with others, the large categories will dominate classifiers’ parameter learning for most general statistical-based classifiers. And these will drag down Recall in small categories and *MacroF1* in overall evaluation. Because the problem of multi-label categories [2] is not in our consideration, we just choose the single label instances from the whole dataset as the imbalanced dataset. And the balanced dataset (comparing with whole dataset) is generated from the imbalanced dataset by choosing categories contains from 10 to 200 instances, so we can control the category size rate under 20.

For 20Newsgroup, it is single-labeled balanced dataset, which contains twenty 1000-size categories. We use Naive Bayes as our baseline, and choose KNN and SVM to be compared with. We

use the NB and KNN programs implemented in Weka Tool for our experiment.

In SVM, we use LibSVM with radial basis function and CSVC (Classification SVM Type 1). We use 10-fold cross validation and the training rate was set to 0.75 in these two datasets.

3.2 Performance Evaluation

To measure the performance over the whole dataset, especially for multi-class case, we use *MacroF1* and *MicroF1* [9] for performance evaluation

F1 measure:

$$F1 = \frac{2RP}{R+P} \quad (5)$$

where R represents Recall and P Precision.

MacroF1 is defined as:

$$MacroF1 = \frac{\sum F1}{n} \quad (6)$$

where n is the number of classes. *MacroF1* is just an average value of *F1* scores with respect to various classes.

MicroF1 is defined as:

$$MicroF1 = \frac{2 * \sum R * \sum P}{\sum R + \sum P} \quad (7)$$

MicroF1 uses the total Recall and Precision values related to various classes to calculate the *F1* score. From Formula (5), (6) and (7), we can see that bigger numbers of classes contribute more *MicroF1* than the smaller ones, which the high *MicroF1* mostly depends on the good performance on big categories. To avoid this, *MacroF1* was introduced. *MacroF1* is just an average value of *F1* scores related to various classes. It won’t be affected by the size of a category, but actually depends on the real application. If we much more care for correct instances but ignore the distribution of them, it is better to use *MicroF1*. But if the application is prone to more balanced result, we can use *MacroF1* instead. Both metrics are applied in our test.

Table 2 shows the result on imbalanced Reuter-21578. We can see that the iterative voting method gets higher scores in *MacroF1* than SVM and lower in *MicroF1*. It is the same in Recall and Precision metrics. As the dataset is imbalanced, according to its principle [4], SVM favors the large topics, which will cause the *MicroF1* much higher than other methods. As for the iterative voting method, it will not be affected by the size of topics. So the performance of small topics remains the same as big ones.

Table 2. Imbalanced dataset on Reuter-21578

	MacroF1	MicroF1	Recall	Precision
NB	0.2501	0.8438	0.2380	0.3357
KNN	0.6032	0.8997	0.5831	0.6664
SVM	0.7697	0.9432	0.7335	0.8476
IV	0.7984	0.9087	0.8179	0.7801

Table 3 shows the result on balanced Reuter-21578. We can see the iterative voting method gets higher scores in both *MacroF1* and *MicroF1* than SVM.

Table 3. Balanced dataset on Reuter-21578

	MacroF1	MicroF1	Recall	Precision
NB	0.5388	0.7026	0.5058	0.6958
KNN	0.7174	0.8075	0.6972	0.7948
SVM	0.8504	0.9072	0.8191	0.9238
IV	0.8917	0.9200	0.8955	0.8881

Table 4 shows the result on 20Newsgroup. We can see that the iterative voting method scores almost the same as SVM in every metrics.

Table 4. Experiment on 20Newsgroup

	MacroF1	MicroF1	Recall	Precision
NB	0.7283	0.7545	0.7775	0.7542
KNN	0.7844	0.7861	0.7908	0.7859
SVM	0.8309	0.8305	0.8325	0.8305
IV	0.8309	0.8357	0.8355	0.8339

Table 5 shows time efficiency for different algorithms on both datasets. We can see that in time efficiency, the iterative voting method is much faster than other methods.

Table 5. Time Efficiency(s)

	Reuter-21578	20Newsgroup
NB	284.656	1467.83
KNN	4255.36	24630
SVM	1017.88	5809.58
IV	127.891	931.5

3.3 Discussion

From the aspect of performance testing, we can see that in every metrics, the iterative voting method is better than NB and KNN. Comparing with SVM, its Recall is much better but Precision is worse. For the reason, we found that the iterative voting method gets less correct instances than SVM in imbalanced dataset, but its distribution is more balanced (See Figure 1). Meanwhile, for small categories, the performance gets a great improvement. In Figure 2, we can see that in SVM, for small-size categories, performance is not satisfactory. In 15 categories, which size is under 10, 6 of them did not get a single correct instance. On the contrary, their performance is much better in the iterative voting method. LibSVM uses one-versus-one way to deal with multi-topics. This method trains hyper-planes between every two topics. When instances come, all hyper-planes vote. The one which gets most votes will label the instances. It works well in the large categories. But for small categories, there are only a few support vectors to be used for calculation. So the hyper-planes between each pair of categories vary greatly and results in either great or terrible performance for small categories.

However, in the iterative voting method, all categories are treated as the same. Each instance, no matter where it locates in the “vector space”, gets the same opportunities to be classified. This is the main reason why the iterative voting gets more corrects in small categories than LibSVM, but is less in big categories.

We test our iterative voting method both with and without stop word and the result shows that our method is not sensitive to stop word, because for common word like “a” or “and”, the weights are very small in every categories, they do not affect the voting result.

Both Reuter-21578 and 20Newsgroup datasets contain multiple categories. It is hard for a classifier to decide which categories the test instance belongs to. As above, SVM uses “one-versus-one” voting method to decide. In the iterative voting method, we use iterative voting to reduce candidate number, so the decision will be easier to make. Every round we drop the last few categories. Though the correct categories may be dropped for some instance, this strategy does help the classifier a lot.

The main advantage of the iterative voting method is its time efficiency. Its algorithm complexity is $O(n)$. Along with its good performance, the iterative voting method is more suitable for massive classification online for textual materials.

4. CONCLUSIONS

In this paper, we present a novel method for text classification, which uses word density as weights for voting. And an iteration strategy is given. This method shows a good performance on traditional experimental datasets, i.e., Reuter-21578 and 20Newsgroup, by comparing with SVM, NB and KNN. And it is also with the extraordinary time efficiency. According to its features, it can be more suitable for massive data classification as an online classifier.

Our future study on the iterative voting method will be mainly focused on two topics. First, as the iterative voting method is based on word density, it’s sensitive to word. In experiment, we found that very shorter article will be harder for classifying. How can it performs on shorter article will highly influence its application in online classifier. Second, its principle and advantage in efficiency will be suitable for incremental classification, which is also necessary for an online classifier. We will improve its algorithm to further apply for the incremental classification.

5. ACKNOWLEDGMENTS

This work was supported by National Natural Science Funding of China under Grant No. 90718022.

6. REFERENCES

- [1] Han, J. and Kamber, M. 2006. *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publisher
- [2] Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. 2004. Learning multi-label scene classification, *Pattern Recognition*, 38:1757–1771
- [3] Yang, Y. and Liu, X. 1999. A re-examination of text categorization method, *SIGIR-99*
- [4] Joachims, T. 1999. Making large-scale support vector machine

learning practical, *Advances in Kernel Methods: Support Vector Learning*, MIT Press

[5] McCallum, A and Nigam, K. 1998. A comparison of event models for Naive Bayes text classification, *AAAI/ICML-98 Workshop on Learning for Text Categorization*

[6] Joachims, T. 1998. Text categorization with support vector machines: learning with many relevant features, *ECML'98*

[7] Kotsiantis, S. 2007. Supervised machine learning: A review of classification techniques, *Journal of Informatica*, 31:249–268

[8] Lewis, D. 1992. Feature selection and feature extraction for

text categorization, *Speech and Natural Language*, Morgan Kaufmann Publisher: 212-217

[9] Liu, Y., Loh, H. T., and Sun, A. 2009. Imbalanced text classification: A term weighting approach. *Expert Systems with Applications* 36: 690–701

[10] Bay, S. 1999. The UCI KDD Archive [<http://kdd.ics.uci.edu>]. Irvine, CA: University of California, Department of Information and Computer Science

[11] Yang, Y. and Pedersen, J. 1997. A comparative study on feature selection in text categorization, *ICML'97*

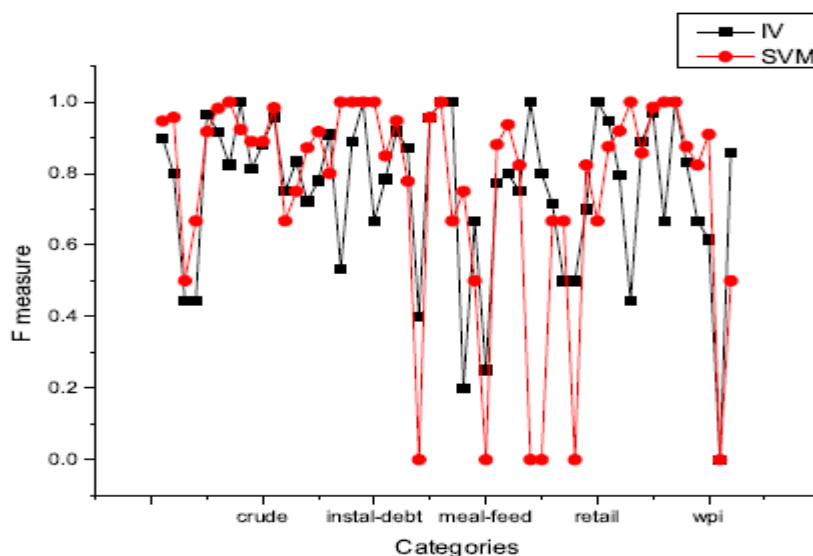


Figure 1. F measures of SVM and Iterative Voting (IV) on Reuter-21578

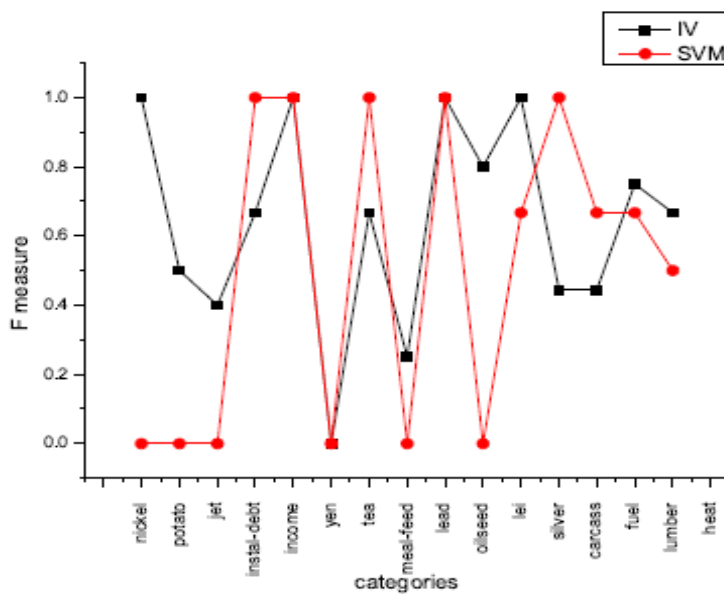


Figure 2. F measures of topics with size < 10