Improve Text Classification Accuracy based on Classifier Fusion Methods

Ali Danesh

Dept. of Elec. & Comp. Eng.
University of Tehran
Tehran, Iran
ali_danesh_ir@yahoo.com

Behzad Moshiri

Control & Intelligent Processing Center
of Excellence, University of Tehran
Tehran, Iran
moshiri@ut.ac.ir

Omid Fatemi

Dept. of Elec. & Comp. Eng.
University of Tehran
Tehran, Iran
omid@fatemi.net

Abstract - *Naïve-Bayes and k-NN classifiers are two machine learning approaches for text classification. Rocchio is the classic method for text classification in information retrieval. Based on these three approaches and using classifier fusion methods, we propose a novel approach in text classification. Our approach is a supervised method, meaning that the list of categories should be defined and a set of training data should be provided for training the system. In this approach, documents are represented as vectors where each component is associated with a particular word. We propose voting methods and OWA operator and Decision Template method for combining classifiers. Experimental results show that these methods decrease the classification error 15 percent as measured on 2000 training data from 20 newsgroups dataset.*

Keywords: Text Classification, Naïve-Bayes, K-NN, Rocchio, TFIDF, OWA, Decision Template, Voting, Classifier Fusion

1 Introduction

Document retrieval, categorization, routing, and filtering systems are often based on text classification. A typical classification problem can be stated as follows: Given a set of labeled examples belonging to two or more categories (training data), classify a new test sample to a category with the highest similarity.

Text classification has become a more and more important application of machine learning. Unlike the conventional machine learning domains, text classification has many special traits. Firstly, it is very often for a typical text classification application that there are hundreds of thousands features to be considered, while most of them are sparse in the document collections. Secondly, many features in the text classification tasks are redundant, which make classifiers prone to over fitting [12].

Text categorization is the problem of automatically assigning one or more predefined categories to free text documents. While more and more textual information is available online, effective retrieval is difficult without

good indexing and summarization of document content. Document categorization is one solution to this problem. A growing number of statistical classification methods and machine learning techniques have been applied to text categorization in recent years.

2 Previous works

A number of classification methods have been discussed in the literature for document classification. These include, naïve Bayes classifier, decision trees [13], k-nearest neighbor classifier [3], linear discriminant analysis (LDA) [4], logistic regression [5] and neural networks [5], support vector machines [5], rule learning algorithms [6], relevance feedback [10], and neural networks [8].

Most of research in text categorization has been devoted to binary problems, where a document is classified as either *relevant* or *not relevant* with respect to predefined topic. However, there are many sources of textual data, such as Internet News, electronic mail and digital libraries, which are composed of different topics and which therefore pose a multi-class categorization problem.

The common approach for multi-class text categorization is to break the task into disjoint binary categorization problems, one for each class. To classify a new document, one needs to apply all the binary classifiers and combine their predictions into a single decision. The end result is a ranking of possible topics.

In what follows we describe three algorithms for text categorization that have been proposed and evaluated in the past, and then describe our proposed algorithm, but first some general notation is given: Let $\mathbf{d} = \{d^1, d^2, \dots, d^M\}$ be the document vector to be classified and $w_1, w_2, \dots, w_M$ are all possible words and let $C = \{c_1, c_2, \dots, c_K\}$ be the possible topics. Further assume that we have a training set consisting of N document vectors $\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N$ with true classes $y_1, y_2, \dots, y_N$. N_j is then the number of training document for which the true class is c_j .

In this paper, we use three different classification methods: naïve Bayes classifier and k-nearest neighbor classifier and Rocchio algorithm. Our approach is based

on combining these methods by voting algorithms and OWA operator and Decision Template method.

2.1 Feature Selection

In the field of machine learning many have argued that maximum performance is often not achieved by using all available features, but by using only a good subset of those. The problem of finding a good subset of features is called feature selection. Applied to text classification this means that we want to find a subset of words which helps to discriminate between classes. Having too few features can make it impossible to formulate a good hypothesis. But having features which do not help discriminate between classes adds noise.

In this paper a combination of three feature selection methods is used, which are explained in more detail below:

1. Pruning of infrequent words.
2. Pruning of high frequency words.
3. Choosing words which have high mutual information with the target concept.

Pruning of infrequent words means that words are only considered as features, if they occur at least a times in the training data. In the experiments described in this paper words had to occur at least a times. This removes most spelling errors and speeds up the following stages of features selection.

In a second step the b most frequently occurring words are removed. This technique is supposed to eliminate non content words like "the", "and", or "for".

In a final step the remaining words are ranked according to their mutual information $I(T, w)$ with the target concept. Mutual information measures the reduction in entropy that is achieved, if one random variable is conditioned on another one. In our case we look at how well the occurrence of a word predicts whether an article is in a certain category or not. Mutual information can be calculated as:

(1)

$$\begin{aligned}
 I(T, t) &= E(T) - E(T | t) \\
 &= - \sum_{c \in C} \Pr(T(d) = c) \cdot \log \Pr(T(d) = c) \\
 &\quad + \sum_{c \in C} \Pr(T(d) = c, t = 0) \cdot \log \Pr(T(d) = c, t = 0) \\
 &\quad + \sum_{c \in C} \Pr(T(d) = c, t = 1) \cdot \log \Pr(T(d) = c, t = 1)
 \end{aligned}$$

$E(X)$ is the entropy of the random variable X . $\Pr(T(d)=c)$ is the probability that an arbitrary article d is in category c . $\Pr(T(d)=c, w=1)$ and $\Pr(T(d)=c, w=0)$ are the probabilities that article d is in category c and it does or does not contain word w . The 10 words with the highest mutual information for the 20 newsgroups dataset are shown in table 1. (fc is term frequency in each class. fdc is document frequency in each class.)

Table 1 : The 10 words with the highest mutual information for the 20 newsgroups dataset

#	Word	I	fdc	fc
0	autos	0.0512933	{.....65.....}	{.....133.....}
1	hockey	0.0512933	{.....65.....}	{.....120.....}
2	crypt	0.0512933	{.....65.....}	{.....99.....}
3	aramis	0.0471066	{.....64.....}	{.....64.....}
4	hedrick	0.0440467	{.....63.....}	{.....63.....}
5	athos	0.0078143	{.....33.....}	{.....109.....}
6	bike	0.00628723	{.....30.....}	{.....65.....}
7	crypto	0.00182028	{.....17.....}	{.....30.....}
8	wiretap	0.00160074	{.....16.....}	{.....20.....}
9	--clh	0.00103443	{.....13.....}	{.....13.....}

2.2 Rocchio's algorithm

Rocchio is the classic method for document routing for filtering in information retrieval. This type of classifier is based on the Rocchio relevance feedback algorithm and uses TFIDF (Term Frequency Inverse Document Frequency) word weights. There are a number of algorithms in this family which differ in their selection of word weighting method and similarity measure [6]. The variant presented here seems to be the most straightforward application of this type of algorithm to text categorization domains with more than two categories. Similar algorithms have been used in [14].

The basic idea of the algorithm is to represent each document $\mathbf{d}$ as a vector in a vector space so that documents with similar content have similar vectors.

The values of the vector elements for a document $\mathbf{d}$ are calculated as a combination of the statistics $TF(w, \mathbf{d})$ and $IDF(w)$. The term frequency $TF(w, \mathbf{d})$ is the number of times word w occurs in document $\mathbf{d}$. The document frequency $IDF(w)$ is the number of documents in which the word w occurs at least once. The inverse document frequency $IDF(w)$ can be calculated from the document frequency.

$$IDF(w) = \log \left(\frac{|D|}{DF(w)} \right) \quad (2)$$

$|D|$ is the total number of documents. The inverse document frequency of a word is low if it occurs in many documents and is highest if the word occurs in only one. The value d^i of feature w_i for document $\mathbf{d}$ is then calculated as the product:

$$d^i = TF(w_i, \mathbf{d}) \cdot IDF(w_i) \quad (3)$$

The TFIDF algorithm learns a class model by combining document vectors into a prototype vector $\mathbf{c}$ for every class C . Prototype vectors are generated by adding the document vectors of all documents in the class.

$$\mathbf{c} = \sum_{\mathbf{d} \in C} \mathbf{d} \quad (4)$$

A document vector $\mathbf{d}$ is classified by calculating the distance between $\mathbf{d}$ and each of the prototype vectors. The distance, for example, can be calculated by dot product.

$$c^* = \arg \max_{c \in C} \frac{\mathbf{d}' \cdot \mathbf{c}}{\|\mathbf{d}'\| \cdot \|\mathbf{c}\|} \quad (5)$$

2.3 Naïve Bayes Classifier

The naïve Bayes (NB) classifiers use the joint probabilities of features to estimate the class probabilities of a document following the Bayesian formula [7]. The naïve Bayes classifiers have been successfully applied in text classification [8]. The underlying assumption of the naïve Bayes approach is that for a given class, the probabilities of words occurring in a document are independent of each other. There are two different generative models in common usage of the naïve Bayes classifiers [9]. One model specifies that a document be represented by a vector of binary attributes indicating which words occur and which not occur in the document. We call this the binary naïve Bayes classifiers. The second model specifies that a document be represented by the set of word occurrences from the document. The latter one is called multinomial naïve Bayes classifiers [10]. The naïve Bayes classifier [4] is constructed by using the training data to estimate the probability of each class given the document feature values of a new instance. We use Bayes theorem to estimate the probabilities:

$$P(c_j | \mathbf{d}) = \frac{P(c_j) P(\mathbf{d} | c_j)}{P(\mathbf{d})} \quad (6)$$

The denominator in the above equation does not differ between categories and can be left out. Moreover, the naïve part of such a model is the assumption of word independence, i.e. we assume that the features are conditionally independent, given the class variable. This simplifies the computations yielding

$$P(c_j | \mathbf{d}) = P(c_j) \prod_{i=1}^M P(d_i | c_j) \quad (7)$$

An estimate $\hat{P}(c_j)$ for $P(c_j)$ can be calculated from the fraction of training documents that is assigned to class c_j :

$$P(c_j) = \frac{N_j}{N} \quad (8)$$

Moreover, an estimate $\hat{P}(d_i | c_j)$ for $P(d_i | c_j)$ is given by:

$$\hat{P}(d_i | c_j) = \frac{1 + N_{ij}}{M + \sum_{k=1}^M N_{kj}} \quad (9)$$

Where N_{ij} is the number of times word i occurred within documents from class c_j in the training set.

Despite the fact that the assumption of conditional independence is generally not true for word appearance in

documents, the naïve Bayes classifier is surprisingly effective.

2.4 K-Nearest Neighbor Classifier

To classify an unknown document vector $\mathbf{d}$, the k-nearest neighbor (k-NN) algorithm ranks the document's neighbors among the training document vectors, and uses the class labels of the k most similar neighbors to predict the class of the input document. The classes of these neighbors are weighted using the similarity of each neighbor to $\mathbf{d}$, where similarity may be measured by for example the Euclidean distance or the cosine between the two document vector.

Figure 1 : k-NN algorithm for $k = 3$

Algorithm:

- Calculate similarity between test document and each neighbour in training set.
- Select k nearest neighbours of a test document among training set.
- Assign test document to the class which contains most of the neighbours.

$$i^* = \arg \max_i \sum_{j=1}^k \text{sim}(D_j, D) * \delta(C(D_j), i) \quad (10)$$

Where:

$$\text{sim}(D_j, D) = \frac{1}{\sqrt{(\mathbf{d}_j - \mathbf{d})(\mathbf{d}_j - \mathbf{d})}} \quad (11)$$

$$\delta(a, b) = \begin{cases} 1; & a = b \\ 0; & \text{otherwise} \end{cases} \quad (12)$$

3 Proposed Algorithm

Many researchers have investigated the techniques of combining the predictions of multiple classifiers to produce a single classifier [1, 11]. By combining classifiers we are aiming at a more accurate classification decision at the expense of increased complexity. Voting

algorithms and OWA operator and Decision Template are three classifier fusion methods that we use in this paper.

3.1 Voting Algorithms

Voting algorithms take the outputs of some classifiers as input and select a class which has been selected by most of the classifiers as output.

We use three text classifiers, including Naïve Bayes, K-nearest neighbor and Rocchio. The output of these classifiers used as input for voting combiner.

3.1.1 Majority Voting

If two or three classifiers are agree on a class for a test document, the result of voting classifier is that class. But if each classifier has a different output, we select output of K-nearest neighbor classifier as output of voting classifier, because K-nearest neighbor has a better accuracy rather than the other classifiers. (voting1)

Table 2 : Classification rate for single classifiers based on test data

Classifier	Accuracy
Rocchio	86%
Naïve bayes	86.71%
k-nearest neighbor	87.57%

$$C_{\text{voting}} = \begin{cases} C_{NB} & ; C_{NB} = C_{KNN} \neq C_{Rocchio} \\ C_{KNN} & ; C_{KNN} = C_{Rocchio} \neq C_{NB} \\ C_{Rocchio} & ; C_{Rocchio} = C_{NB} \neq C_{KNN} \\ C_{KNN} & ; C_{NB} = C_{KNN} = C_{Rocchio} \\ C_{KNN} & ; C_{NB} \neq C_{KNN} \neq C_{Rocchio} \end{cases} \quad (13)$$

3.1.2 Weighted Majority Voting

In this method, the weights are specific for each class.

$$c^* = \arg \max_c \sum_{j=1}^L w_{ij} d_{i,j} \quad (14)$$

where $d_{i,j}$ is support of classifier i for class c_j . We propose two methods for calculating weights.

Figure 2 : Accuracy of classification in each class based on train data

1. first weighting method : (voting2)

$$w_{ij} = \frac{\sum_{kl(z_k)=j} \delta(D_i(z_k), j)}{N_j} \quad (15)$$

2. second weighting method : (voting3)

$$w_{ij} = \frac{\sum_{kl(z_k)=j} \delta(D_i(z_k), j)}{N_j} - \frac{\sum_{kl(z_k) \neq j} \delta(D_i(z_k), j)}{N - N_j} \quad (16)$$

In this weighting method, error of each classifier is calculated.

Figure 3 : Error of classification in each class based on train data

Table 3 : Classification rate for Voting

Classifier	Accuracy
Voting1	88.29%
Voting2	88.14%
Voting3	89.29%

3.2 OWA Operators

Here we briefly review the class of aggregation operators called the OWA operators. An OWA operator defined on

the unit interval I and having dimension N , is a mapping $F: I^n \rightarrow I$ such that

$$F(a_1, \dots, a_n) = \sum_{j=1}^n w_j b_j \quad (17)$$

where b_j is the j th largest of the a_j and w_j are a collection of weights such that $w_j \in [0,1]$ and $\sum_{j=1}^n w_j = 1$.

Note. If $id(j)$ is the index of the j th largest of a_i then $a_{id(j)} = b_j$ and $F(a_1, \dots, a_n) = \sum_{j=1}^n w_j a_{id(j)}$.

Note: If $\mathbf{W}$ is an n vector whose j th components is w_j and $\mathbf{B}$ is an n vector whose j th components are b_j then $F(a_1, \dots, a_n) = \mathbf{W}^T \mathbf{B}$. In this formulation $\mathbf{W}$ is referred to as the OWA weighting vector and $\mathbf{B}$ is called to ordered argument vector. The OWA operator is parameterized by the weighting vector $\mathbf{W}$ [2].

(18)

$$\text{Minimum: } \mathbf{W} = [0, 0, \dots, 1]^T$$

$$\text{Maximum: } \mathbf{W} = [1, 0, \dots, 0]^T$$

$$\text{Median: } \mathbf{W} = \begin{cases} \left[\underbrace{0, \dots, 0}_{\frac{L-1}{2}}, 0, 1, \underbrace{0, \dots, 0}_{\frac{L-1}{2}} \right]^T & ; \quad L \text{ is odd} \\ \left[\underbrace{0, \dots, 0}_{\frac{L-1}{2}}, 0.5, 0.5, \underbrace{0, \dots, 0}_{\frac{L-1}{2}} \right]^T & ; \quad L \text{ is even} \end{cases}$$

$$\text{Average: } \mathbf{W} = \left[\frac{1}{L}, \frac{1}{L}, \dots, \frac{1}{L} \right]^T$$

$$\text{Competition jury: } \mathbf{W} = \left[0, \frac{1}{L-2}, \dots, \frac{1}{L-2}, 0 \right]^T$$

Table 4 : Classification rate for OWA

Classifier	Accuracy
OWA1 (W Maximum)	87.86%
OWA2 (W Median)	89.57%

3.3 Decision Template

The idea of the decision templates (DT) combiner is to remember the most typical decision profile for each class w_j , called the decision template, DT_j , and then compare it with the current decision profile $DP(x)$ using some similarity measure S . The closest match will label x [1].

$$DT_j = \frac{1}{N_j} \sum_{z_k \in \omega_j} DP(z_k) \quad (19)$$

$$\mu_j(x) = S(DP(x), DT_j), \quad j = 1, \dots, c. \quad (20)$$

where S defined as below:

$$\mu_j(x) = 1 - \frac{1}{L \times c} \sum_{i=1}^L \sum_{k=1}^c [DT_j(i, k) - d_{i,k}(x)]^2 \quad (21)$$

4 Experimental Result

An experiment was performed to show the performance of fusion methods on real data. The dataset that we used consisted of Usenet articles collected from 20 different newsgroups (table 5). Over a period of time 100 articles were taken from each of the newsgroups, which make an overall number of 2000 documents in this collection. Each document exactly belongs to one newsgroup. The task is to learn which newsgroup an article was post to.

Table 5 : Usenet newsgroups used in newsgroup dataset

#	Newsgroup Name	#	Newsgroup Name
0	alt.atheism	10	rec.sport.hockey
1	comp.graphics	11	sci.crypt
2	comp.os.ms-windows.misc	12	sci.electronics
3	comp.sys.ibm.pc.hardware	13	sci.med
4	comp.sys.mac.hardware	14	sci.space
5	comp.windows.x	15	soc.religion.christian
6	misc.forsale	16	talk.politics.guns
7	rec.autos	17	talk.politics.mideast
8	rec.motorcycles	18	talk.politics.misc
9	rec.sport.baseball	19	talk.religion.misc

The documents in this dataset have the typical properties of Usenet articles. A random subset of 65% of the data considered in an experiment was used for training and 35% of the data considered for testing. The result has been shown in table 6.

Table 6 : Classification rate

Classifier	Classification Rate
Naïve Bayes	86%
K-Nearest Neighbor	86.71%
Rocchio	87.57%
Voting1	88.29%
Voting2	88.14%
Voting3	89.29%
OWA1	87.86%
OWA2	89.57%
DT	87.86%

Figure 4 : Classification rate

5 Conclusion and Future Works

In this paper, we have proposed a novel approach for text classification. Our approach is based on combining classifiers. We combined Rocchio and k-Nearest Neighbour and Naïve Bayes classifiers by voting algorithms and OWA operators and Decision Template method and achieve a better classification rate which experimental results show that the classification error decreased 15 percent. We use 2000 documents from 20 different newsgroups for testing our proposed methods. There are other combination methods such as Behaviour Knowledge Space, Werneche, Naïve Bayes Combiner, Dempster Shafer [1] and etc for future works.

References

- [1] Kuncheva, LI. "Combining Pattern Classifiers Methods and Algorithms", John Wiley & Sons, 2004.
- [2] Ronald R. Yager, "An extension of naïve Bayesian classifier", Information Sciences 176, pp. 577-588, 2006.
- [3] S. Weiss, S. Kasif, and E. Brill. Text Classification in USENET Newsgroup: A Progress Report. In AAAI Spring Symposium on Machine Learning in Information Access Technical Papers, Palo Alto, March 1996.
- [4] D. Hull, J. Pedersen, and H. Schutze. Document Routing as Statistical Classification. In AAAI Spring Symposium on Machine Learning in Information Access Technical Papers, Palo Alto, March 1996.
- [5] H. Schutze, D. Hull, and J. Pedersen. A Comparison of Classifiers and Document Representations for the Routing Problem. In SIGIR'95, pages 229-237, Washington D.C., 1995.
- [6] G. Salton and M. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.
- [7] D. D. Lewis, Representation and Learning in Information Retrieval, University of Massachusetts Amherst, MA, 1992.
- [8] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail, Learning for Text Categorization, AAAI Press, 1998, 55-62.
- [9] A.K. McCallum, K. Nigam, A comparison of event models for naïve Bayes text classification, Proc. of AAAI-98 Workshop on Learning for Text Categorization, 1998.
- [10] Yan-Shi Dong, Ke-Song Han, A Comparison of Several Ensemble Methods for Text Categorization, Proc. of the 2004 IEEE International Conference on Services Computing (SCC'04).
- [11] Ludmila I. Kuncheva, Switching between Selection and Fusion in Combining Classifiers: An Experiment, IEEE Transaction on Systems, Man, and Cybernetics - part B: Cybernetics, vol. 32, No. 2, April 2002.
- [12] L. Douglas Baker, Andrew McCallum, Distributional clustering of words for text classification. Proc. of the 21st Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval, ACM Press, 1998, 96-103.
- [13] D. Lewis and M. Ringutte. A comparison of Two Learning Algorithm for Text Categorization. In Third Annual Symposium on Document Analysis and Information Retrieval, pp. 81-93, Las Vegas, NV, 1994.
- [14] David A. Bell, J.W. Guan, and Yaxin Bi, "On Combining Classifier Mass Functions for Text Categorization", IEEE Transactions on Knowledge and Data Engineering, pp. 1307-1319, vol. 17, No. 10, October 2005