Efficient Text Classification Using Best Feature Selection and Combination of Methods

M. Srinivas[1], K.P. Supreethi[2], E.V. Prasad[3], and S. Anitha Kumari[1]

[1] JNTUACE, Anantapur, India
[2] JNTUHCE, Hyderabad, India
[3] JNTUKCE, Kakinada, India
sreenu2521@gmail.com, supreethi.pujari@gmail.com,
drevprasad@yahoo.com, anitha.mahi7@gmail.com

Abstract. Lsquare and k-NN classifiers are two machine learning approaches for text classification. Rocchio is the classic method for text classification in information retrieval. 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 method and OWA operator and Decision Template method for combining classifiers. In these we use an effective and efficient new method called variance-mean based feature filtering method of feature selection. Best feature selection method and combination of methods are used to do feature reduction in the representation phase of text classification is proposed. Using this efficient feature selection method and best classifier combination method we improve the text classification performance.

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.

M.J. Smith and G. Salvendy (Eds.): Human Interface, Part I, HCII 2009, LNCS 5617, pp. 437–446, 2009.

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], knearest 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], Lsquare, 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. 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 d = $\{d_1, d_2 \ldots, d_M\}$ be the document vector to be classified and w_1, $w_2 \ldots, w_M$ are all possible words and let C = $\{c_1, c_2 \ldots, c_K\}$ be the possible topics. Further assume that we have a training set consisting of N document vectors d_1, $d_2 \ldots$, d_N with true classes y_1, $y_2 \ldots$, 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: Lsquare [20] 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. There is various feature selection methods are there such as Document Frequency (DF), Chi-square (CHI), Information Gain (IG), Mutual Information (MI), Term Strength (TS) GSS Coefficient, odds ratio. But in these we can use an effective and efficient new method called variance-mean based feature filtering method of feature selection to do feature reduction in the representation phase for text classification is proposed. This method is more efficient than remaining feature selection methods.

2.1 Feature Selection Variance-Mean Based Filtering Method of Feature Selection

While decreasing the dimension is the task of feature selection, how to select out the features that best characterize the text that belongs to a particular class is its main purpose. Consider the text class set $\{c_1, c_2, \ldots, c_n\}$ assuming each of the classes contains the same number of documents n in the training corpus1. Thus, we get the following matrix:

$$\begin{pmatrix} d_{11} & \ldots & d_{1n} \\ \ldots & \ldots & \ldots \\ d_{m1} & \ldots & d_{mn} \end{pmatrix} \tag{1}$$

Where d_{ij} is the j^{th} document that belongs to the i^{th} class. And for convenience of statement, let d_{ij} also stands for the weight $weight_w$ (d_{ij}) of the candidate feature w (also called term) in that document denoted as in the training corpus. So, for every candidate feature in the original feature space, there is such a matrix. Each d_{ij} in the matrix (1) for a specified term w is computed by the formula defined in the weighting scheme, e.g.

$$d_{ij} = p_{ij}(w) \tag{2}$$

p_{ij} is the probability that term w will occur in document d_{ij}, which is approximated by the frequency of the occurrences of term w in document d_{ij}. In fact any (more

complicated) weighting scheme can be employed here. And here we use the term frequency as the weight for the feature in a text's feature vector for further constructing our evaluation function just because of convenience. We then compute the mean and variance of the values for each row, producing the following vectors:

$$1)\ \begin{bmatrix} E_{r1} \\ \dots \\ E_{rm} \end{bmatrix} \qquad 2)\ \begin{bmatrix} D_{r1} \\ \dots \\ D_{rm} \end{bmatrix} \tag{3}$$

Where E_{r1} is the mean of the values in the i^{th} row in the above matrix (1) and D_{r1} is the variance of those in the i^{th} row. Thirdly, we compute the variance of the components' data in vector 1), denoted as D(E_{r1}), which shows the degree of dispersion among classes the term w can demonstrate and the mean of the components' data in vector 2), denoted as E(D_{r1}), which shows the average level of the degree of variability within every class the term w can show. The bigger the value of D (E_{r1}), the more distinguishable among classes using that term w is; the smaller the value of E (D_{r1}), the more cohesive within each single class averagely using that term w is. And the more distinguishable among classes and cohesive within each single class using that term w is, the more possible the term w should be remained. So D (E_{r1}) and E (D_{r1}) based criterion can be used to evaluate the importance of the candidate term w. Then the evaluation function could be:

$$1)\quad \beta * D(E_{r1})/E(D_{r1}) \tag{4}$$

Where β is the tuning parameter. If for term w, the F value is more than the threshold f, a.k.a. F>f, the term w is selected. And the threshold f is set according to the experiment. Or it could be

$$2)\quad E(D_{r1}) \tag{5}$$

The features whose E (D_{r1}) values are bigger than a threshold got by experiment are filtered. Or the filtering function

$$3)\quad D(E_{r1}) \tag{6}$$

can be applied, and the features satisfying D (E_{r1}) > d (where d is also a threshold) will be kept. The strategy of setting threshold on these evaluation functions' values to determine the size of the feature vector is called THR [2]. In fact another two strategies called PFC and MVS can also be adopted on these evaluation functions. PFC (Predefined Feature Count) selects the first several (the predefined feature count) features that have the largest or smallest evaluation function values. MVS (Mladenic' Vector Size) strategy proposed by Mladenic [2] decides the number of the features to be remained by defining its ratio to the total number of original feature items that appear in the training corpus. Any of the above feature evaluation functions and feature cutting strategies can be chosen to combine together to finish feature filtering in our method. And in our experiment, PFC and MVS strategies are used on feature evaluation function 3) in formula (6).

The computing complicity of the algorithm in accordance with our method is linear to the original feature space's size and can be largely cut down by using techniques in computing statistics. And even if it is time-consuming, compared with its effectiveness in better characterizing the text and its efficiency gained by decreasing the dimension of the vector, it is worthwhile especially when it is computed off-line and is once and for all. The classical methods of taking document frequency or document frequency incorporated measure as evaluation function to evaluate the importance of candidate features such as DF, CHI and IG, which are reported in [19] having better performance among all other methods. Overlook the fact that the properties of the candidate feature in different documents and different classes are different, because they only take whether the term w occurs in a text into consideration when computing the evaluation function, which is not enough to show that kind of differences of term w among different classes. For example, term t1 and term t2 may have the same document frequency, but they may appear different times in a single document. But the capability of a feature in characterizing a text as belonging to a class is closely related to its ability to express that kind of differences among different classes. Thus such classical methods reported having better performance in [19] seem to show no privilege. But if we take the variance of the average index values for each class as the evaluation function as our method does, the differences will be reflected, and the performance will be improved.

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. In these paper we can proposed new an effective and efficient feature selection algorithm can be used. Using this feature selection with the efficient classifiers combination method (OWA) we can increase the classifiers performance. So our classification is more accurate and efficient

3.1 Classifiers Combining Methods

3.1.1 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 Lsquare, Knearest neighbor and Rocchio. The output of these classifiers used as input for voting combiner.

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 Lsquare classifier as output of voting classifier, because Lsquare has a better accuracy rather than the other classifiers. (voting).

Table 1. Classification rate for single classifiers based on test data

classifier	Accuracy
Rocchio	86%
K-NN	87.50%
Lsquare	89.60%

3.1.2 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

$$F(a_1,....,a_n) = \sum_{j=1}^{n} w_j b_j \quad (14)$$

Where b_j is the jth 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 jth largest of a_i then $a_{id(j)} = b_j$ and

$$F(a_1,....,a_n) = \sum_{j=1}^{n} w_j a_{id(j)}$$

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

Minimum: $W = [0, 0,\ldots., 1]^T$

Maximum: $W = [1, 0,\ldots., 0]^T$

Median: $W = \begin{cases} \left[\underbrace{0,...,0}_{\frac{L-1}{2}},0,1,\underbrace{0,0,...,0}_{\frac{L-1}{2}}\right]^T ; L \ is \ odd \\ \left[\underbrace{0,...,0}_{\frac{L}{2}-1},.5,.5,\underbrace{0,...,0}_{\frac{L}{2}-1}\right]^T ; L \ is \ even \end{cases}$

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

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

Table 2. Classification rate for OWA

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

3.1.3 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].

$$DT_j = \frac{1}{N_j} \sum_{z_k \in w_j} DP(Z_k) \tag{15}$$

$$\mu_j(x) = S(DP(x), DT_j), \quad j = 1, ..., c. \tag{16}$$

Where S defined as below:

$$\mu_j(x) = 1 - \frac{1}{L \times C} \sum_{i=1}^{L} \sum_{k=1}^{c} \left[DT_j(i,k) - d_{i,k}(x) \right]^2 \tag{17}$$

4 Experimental Results

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 3). 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.

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 4.

Table 3. Usenet newsgroups used in newsgroup dataset

#	Newsgroup name	#	Newsgroup name
0	comp.design	10	eat.nonveg.fish
1	comp.sys.hcl.pc.hardware	11	sci.computers
2	comp.os.linux	12	sci.medical
3	comp.sys.del.hardware	13	sci.chemistry
4	comp.windows.x	14	sci.space
5	rec.cars	15	soc.religion.hindu
6	rec.sport.cricket	16	talk.politics.guns
7	rec.study.ms	17	cool.drinks.7up
8	rec.education.phd	18	Watch.movies.telugu
9	rec.movies	19	play.games.comp

Table 4. Classification rate

Classifier	Classification Rate
K-Nearest Neighbor	86.71%
Rocchio	87.57%
Lsquare	88.62%
Voting	88.90%
OWA1	88.99%
OWA2	89.59%
DT	88.75%

4.1 Best Feature Selection Result

The corpus used for training and testing is the "web text classification corpus" We divide the corpus into two non-intersected sets: a training set containing 10 categories with 100 texts in each and a test set containing the same 10 categories with another 100 texts in each also. Then we apply the open source lexical processing software made by ICT, (Institute of Computing Technology) to do the word segmentation task, which is part of the pre-processing. We then apply our method to do feature reduction. That is taking D (E_{r1}) as feature evaluation function and applying the MVS and PFC strategy to cut down the feature space's size. And, then we simply adopt the word frequency as the word's weight index in the vector. After vectors are got, the vectors corresponding to the texts in the training corpus are input into ten binary classifiers. Each binary classifier is trained for each of the 10 classes by using each class' texts at a time as positive examples with the rest of the data as negative examples. Each unknown vector from the test set is presented to all the 10 binary classifiers. The output of a binary classifier is positive when it decides that an unknown test vector belongs to the class it was trained for. Since there may be several simultaneous such claims or not a single such claim, a simple maximum selector is applied to the classifiers to make the final decision for an exclusive class [6]. Or multiple tags with the first largest selectors are assigned for the text in the multiple label condition. Every binary classifier is the Lsquare classifier. According to Thorsten Joachims, it is suitable to use Lsquare as classifier for text classification task. Because the theoretical analysis concludes that Lsquare acknowledges the particular properties of text: a) high dimensional feature space b) dense concept vector (most of the features are relevant) and c) sparse instance vectors. In all experiments we use a linear kernel and let C vary automatically accordingly. The performance evaluation data gained using our method in the experiment is depicted in the following diagrams (Table.5 and Table.6). The features are ordered by D(E_{r1}) the value decreasingly, using MVS and PFC feature cutting strategy by setting parameter RtoD=0.1,0.3,0.5,0.7,0.9,1; D=100,200,…,2800 correspondingly. "D" stands for the number of features we want to keep in the above-mentioned PFC strategy, a.k.a. the first several important features ordered by its D (E_{r1}) value decreasingly. "RtoD" means the ratio of the number of features to the total number of features originally extracted from the training corpus in the MVS strategy. Because every feature will be kept if the "RtoD" value in MVS is set to 1, this value of the parameter is set to get the performance result as using no filtering process for comparison. We also give the performance evaluation data gained by using the classical feature filtering methods introduced, the document frequency (DF) and χ^2statistic

(CHI) as a comparison, keeping other conditions being identical to those when using our method in the experiment.

Table 5. The performance evaluation data given by macro-precision M-P, macro-recall M-R and macro-f1 M-F1 using MVS strategy on filtering function $D(E_{r1})$ with the original dimension 28,260

RtoD	D	M-P	M-R	M-F1
0.1	2826	0.876	0.856	0.866
0.3	8478	0.912	0.898	0.905
0.5	14130	0.917	0.902	0.909
0.7	19782	0.938	0.9377	0.934
0.9	25434	0.938	0.9378	0.934
1	28260	0.67	0.67	7293

Table 6. The performance evaluation data given by macro-precision M-P, macro-recall M-R and macro-f1 M-F1 using the PFC strategy on filtering function $D(E_{r1})$

D	M-P	M-R	M-F1	D	M-P	M-R	M-F1
100	0.903	0.884	0.8934	1500	0.8498	0.83	0.8398
200	0.9126	0.894	0.9032	1600	0.8514	0.836	0.8436
300	0.916	0.9	0.9079	1700	0.8574	0.838	0.8476
400	0.9252	0.912	0.9018	1800	0.8617	0.842	0.8517
500	0.6434	0.668	0.6555	1900	0.8644	0.846	0.8551
600	0.6753	0.69	0.6826	2000	0.8654	0.846	0.8556
700	0.6624	0.681	0.6718	2100	0.8654	0.846	0.8556
800	0.7988	0.78	0.7893	2200	0.8679	0.848	0.8578
900	0.8089	0.788	0.7983	2300	0.8693	0.848	0.8585
1000	0.8138	0.796	0.8048	2400	0.8666	0.844	0.8551
1100	0.8274	0.808	0.8176	2500	0.8722	0.852	0.862
1200	0.8335	0.812	0.8226	2600	0.8722	0.852	0.862
1300	0.8421	0.822	0.8319	2700	0.8728	0.852	0.8623
1400	0.8405	0.824	0.8322	2800	0.8763	0.856	0.866

From the above diagrams, it can be seen that: 1) Feature reduction is really important because without it, the performance value is only 0.73 of macro-f1, much lower than the one gained with reduced feature space' size. 2) Our method shows a good property. The performance evaluation data are still high when the dimension is reduced to 100, as shown in Table 6. The performance still gradually goes up until the dimension reaches almost the original size, as shown in Table5. Although the performance gained when the dimension is 400 is not a real maximum, compared with the one gained at dimension 0.9*28260 shown in Table.5, Using feature vector of dimension 400 greatly decreases the computing time. With other conditions being identical, training a Lsquare using feature vectors of dimension 0.9*28260 costs about

25min while using dimension 400 only needs 10sec on Intel Celeron 2.4GHZ CPU with memory of 512M, and furthermore using vector of dimension 0.9*28260 to represent the incoming unknown free text for classification costs almost 120 times the time deciding the class of that text using vector of dimension 400. And the macro-f1 value gained at dimension 0.9*28260 is only slightly bigger. 3) Compared with the classical document frequency incorporated feature filtering methods such as DF, CHI used in our experiment, it shows in Figure.2, that our method can gain higher performance at a very low dimension, and quickly reach a peak, which means much less computing time and almost best performance than other methods.

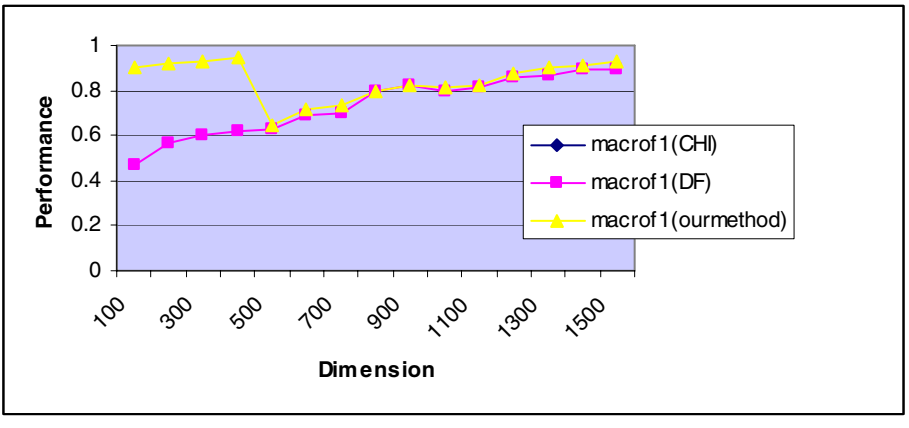

Fig. 2. Performance comparison among DF, CHI and ours. DF and CHI have almost the same performance curves.

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 Lsquare classifiers by voting algorithm 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. Good representation of a text is very important, of which the feature reduction has an obvious effect on the final performance of text classification. In these we can use efficient and effective feature selection method used. This method is called variance-mean based feature filtering method. Using this method we can also increasing the classifiers performance. We can also use an efficient combination method for combine the different classifiers. In these we can compare our method (OWA) with different methods. It will be give better result than compare to remaining methods. Using these two methods we can increase the classifiers performance more afflictive.

References

1. Kuncheva, L.I.: Combining Pattern Classifiers Methods and Algorithms. John Wiley, Chichester
2. Soucy, P., Mineau, G.W.: Feature selection strategies for text categorization. In: Xiang, Y., Chaib-draa, B. (eds.) Canadian AI 2003. LNCS (LNAI), vol. 2671, pp. 505–509. Springer, Heidelberg (2003)
3. Weiss, S., Kasif, S., Brill, E.: Text Classification in USENET Newsgroup: A Progress Report. In: AAAI Spring Symposium on Machine Learning in Information
4. Hull, D., Pedersen, J., Schutze, H.: Document Routing as Statistical Classification. In: AAAI Spring Symposium on Machine Learning in Information Access Technical Papers
5. Schutze, H., Hull, D., Pedersen, J.: A Comparison of Classifiers and Document Representations for the Routing Problem. In: SIGIR 1995, Washington, DC, pp. 229–237 (1995)
6. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill, New York
7. Lewis, D.D.: Representation and Learning in Information Retrieval, University of Massachusetts Amherst, MA (1992)
8. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A bayesian approach to filtering junk e-mail, Learning for Text Categorization, pp. 55–62. AAAI Press, Menlo Park (1998)
9. McCallum, A.K., Nigam, K.: A comparison of event models for naïve Bayes text classification. In: Proc. of AAAI 1998 Workshop on Learning for Text Categorization (1998)
10. Dong, Y.-S., Han, K.-S.: A Comparison of Several Ensemble Methods for Text Categorization. In: Proc. of the 2004 IEEE International Conference on Services Computing (2004)
11. Kuncheva, L.I.: Switching between Selection and Fusion in Combining Classifiers: An Experiment. IEEE Transaction on Systems, Man, and Cybernetics - part B: Cybernetics
12. Douglas Baker, L., McCallum, A.: Distributional clustering of words for text classification. In: Proc. of the 21st Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 96–103. ACM Press, New York (1998)
13. Lewis, D., Ringutte, M.: A comparison of Two Learning Algorithm for Text Categorization. In: Third Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, pp. 81–93 (1994)
14. Bell, D.A., Guan, J.W., Bi, Y.: On Combining Classifier Mass Functions for Text Categorization. IEEE Transactions on Knowledge and Data Engineering
15. Dumais, S.T., Platt, J., Heckerman, D., Sahami, M.: Inductive Learning Algorithms and Representations for Text Categorization. In: Proc. Seventh Int'l Conf. Information and Knowledge Management (1998)
16. Felici, G., Sun, F., Truemper, K.: A Method for Controlling Errors in Two-Class Classification. In: Proc. 23rd Ann. Int'l Computer Software and Applications Conf.
17. Felici, G., Truemper, K.: A Minsat Approach for Learning in Logic Domains. Informs J. Computing 14(1) (Winter 2002)
18. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398. Springer, Heidelberg (1998)
19. Yang, Y., Pedersen, J.O.: A Comparative Study on Feature Selection in Text Categorization. In: Proceedings of 14th International Conference on Machine Learning, San Francisco, pp. 412–420 (1997)
20. Al-Mubaid, H., Umair, S.A.: A New Text Categorization Technique Using Distributional Clustering and Learning Logic. Proc. of IEEE Transactions on Knowledge and Data Engineering 18(9) (September 2006)