Dimensionality Reduction in Text Classification using Feature Reduction Approaches

P. Kiruthiga
Assistant Professor
Department of Computer science & Engineering
PSNA college of engineering & Technology,
Dindigul,TamilNadu
kiruthigapitchai@gmail.com

***Abstract* - Text classification is a supervised technique that uses labeled training data to learn the classification system and then automatically classifies the remaining text using the learned system. Classification plays a vital role in many information management and retrieval tasks. Classification includes different parts such as text processing, feature extraction, feature vector construction and final classification. In this paper, we apply machine learning methods for classification. In this regard, we first try to exert some text preprocess in different dataset, and then we extract a feature vector for each new document by using feature weighting and feature selection algorithms for enhancing the text classification accuracy. After that our classifier is trained by Naïve Bayesian (NB) and K-nearest neighbor (KNN) algorithms. In Experiments, although both algorithms show acceptable results for text classification.**

***Keywords* - Text classification, Feature selection,TextCategorization,Dimensionality Reduction**

I. INTRODUCTION

In Text classification, the dimensionality of the feature vector is usually huge. The dimensionality of feature vectors for text classification can be reduced by the technique called preprocessing. It is practical to remove words which appear too often (in every or almost every document) and thus support no information for the task. Good examples for this kind of words are prepositions, articles and verbs. Text categorization (TC) is a *supervised learning problem* where the task is to assign a given text document to one or more predefined categories. It is a well-studied problem and still continues to be topical area in information retrieval (IR), because of the ever increasing amount of easily accessible digital documents on the Web, and, the necessity for organized and effective retrieval. High dimensionality of feature space is a major problem in TC. The number of terms (i.e., features) present in a collection of documents, in general, is large and few are informative. Feature selection for Text classification is the task of reducing dimensionality of feature space by identifying informative features and its primary goals are improving classification effectiveness, computational efficiency, or both. The performance of a classifier is affected by the employed feature selection mechanism.

II. RELATED WORK

Feature selection is an important method for improving the efficiency and accuracy of text categorization algorithms by removing redundant and irrelevant terms from the corpus Feature selection is the process of selecting a subset of the terms occurring in the training set and using only this subset as features in text classification. Feature selection serves two main purposes. First, it makes training and applying a classifier more efficient by decreasing the size of the effective vocabulary. Feature selection often increases classification accuracy by eliminating noise features. A noise feature is one that, when added to the document representation, increases the classification error on new data.

III. FEATURE WEIGHTING

In the selection process, each feature (term or single word) is assigned with a score according to a score-computing function. Then those with higher scores are selected. These mathematical

definitions of the score-computing functions are often defined by some probabilities which are estimated by some statistic information in the documents across different categories. For the convenience of description, we give some notations of these probabilities below;

1. *P(t):the probability that a document x contains term t;*
2. *P(Ci): the probability that a document x not belong to category Ci;*
3. *P(t,Ci): the joint probability that a document x contains term t and also belongs to category Ci;*
4. *P(Ci/t): the probability that a document x belongs to category Ci,under the condition that it contains term t;*
5. *P(t/Ci): the probability that, a document not contain term t with the condition that x belongs to category Ci;*

A. Document Frequency(DF)

DF is the number of documents in which a term occurs. It is defined as

$$DF = \sum_{i=1}^{m} (A_i)$$

Document frequency is the number of documents in which a term occurs. We computed the document frequency for each unique term in the training corpus and removed from the feature space those terms whose document frequency was less than some predetermined threshold. DF thresholding is the simplest technique for vocabulary reduction. It easily scales to very large corpora, with a computational complexity approximately linear in the number of training documents.

B. Mutual Information (MI)

Mutual Information can be proven equal to Information Gain for binary problems. For mutli-class problems (with global feature lists) like we present in this report however, the two are not equal (although rather similar). Thus we present Mutual Information with its own equation as a separate feature selection algorithm here.

$$MI(F,C_k) = \sum_{v_f \in \{1,0\}} \sum_{v_{C_k} \in \{1,0\}} P(F = v_f, C_k = v_{C_k}) \ln \frac{P(F = v_f, C_k = v_{C_k})}{P(F = v_f)P(C_k = v_{C_k})}$$

where F is the discrete random variable `feature' that takes the value vF = f1; 0g (feature F occurs in document or not), Ck is the discrete random variable `category' that takes the values vCk = f1; 0g (document belongs to category Ck or not).

The probabilities can be estimated by using the various document counts from the training set.

$$MI(F,C_k) = \frac{N_{F,C_k}}{N} \ln \frac{NN_{F,C_k}}{N_F N_{C_k}} + \frac{N_{F,\overline{C_k}}}{N} \ln \frac{NN_{F,\overline{C_k}}}{N_F N_{\overline{C_k}}} + \frac{N_{\overline{F},C_k}}{N} \ln \frac{NN_{\overline{F},C_k}}{N_{\overline{F}} N_{C_k}} + \frac{N_{\overline{F},\overline{C_k}}}{N} \ln \frac{NN_{\overline{F},\overline{C_k}}}{N_{\overline{F}} N_{\overline{C_k}}}$$

Then the values can be weighted and summarized to create a global ranked list of features:

$$MI(F) = \sum_{k=1}^{|C|} \frac{N_{C_k}}{N} MI(F,C_k)$$

C. Information Gain (IG)

Here both class membership and the presence/absence of a particular term are seen as random variables, and one computes how much information about the class membership is gained by knowing the presence/absence statistics (as is used in decision tree induction. Indeed, if the class membership is interpreted as a random variable *C* with two values, positive and negative, and a word is likewise seen as a random variable *T* with two values, *present* and *absent*, then using the information-theoretic definition of mutual information we may define Information Gain as:

IG(t) = H(C) – H(C|T) = Στ,c P(C=c,T=τ) ln[P(C=c,T=τ)/P(C=c)P(T=τ)].

Here, τ ranges over {present, absent} and c ranges over {c+, c–}.

D. Chi-Square Statistics(CHI)

Feature Selection by X2 testing is based on Pearson's X2 (chi square) test. The X2 test is often used to test the independence of two variables. The null-hypothesis is that the two variables are completely independent of each other. The higher value of the X2 test, the closer relationship the variables have.

In feature selection, the X2 test measures the independence of a feature and a category. The null-hypothesis here is that the feature and category are completely independent, i.e. that the feature is useless for categorizing documents. The higher X2 value for a (feature, category) pair, the less independent they are. Hence, the features with the highest X2 values for a category should perform best for categorizing documents.

$$\chi^2(F,C_k) = \frac{N \times \left(\left(N_{F,C_k} \times N_{\overline{F},\overline{C_k}}\right) - \left(N_{F,\overline{C_k}} \times N_{\overline{F},C_k}\right)\right)^2}{N_F \times N_{\overline{F}} \times N_{C_k} \times N_{\overline{C_k}}}$$

E. Odds Ratio(OR)

Odds Ratio compares the odds of a feature occurring in one category with the odds for it occurring in another category. It gives a positive score to features that occur more often in one category than in the other, and a negative score if it occurs more in the other. A score of zero means the the odds for a feature to occur in one category is exactly the same as the odds for it to occur in the other, since ln (1) = 0.
The original Odds Ratio algorithm for binary categorization:

$$OR(F,C_k) = \ln \frac{P(F|C_k)(1-P(F|\overline{C_k}))}{P(F|\overline{C_k})(1-P(F|C_k))} = \ln \frac{\left(\frac{N_{F,C_k}}{N_{C_k}}\right)\left(1-\frac{N_{F,\overline{C_k}}}{N_{\overline{C_k}}}\right)}{\left(\frac{N_{F,\overline{C_k}}}{N_{\overline{C_k}}}\right)\left(1-\frac{N_{F,C_k}}{N_{C_k}}\right)}$$

$$P(F|C_k) = \frac{N_{F,C_k}}{N_{C_k}}$$

Let $P(t|c)$ be the probability of a randomly chosen word being *t*, given that the document it was chosen from belongs to a class *c*. Then $odds(t|c)$ is defined as $P(t|c)/[1-P(t|c)]$ and the Odds Ratio equals to

$OR(t) = \ln[odds(t|c+)/odds(t|c-)]$.

Obviously, this scoring measure favors features that are representative of positive examples. As a result a feature that occurs very few times in positive documents but never in negative documents will get a relatively high score. Thus, many features that are rare among the positive documents will be ranked at the top of the feature list. Odds Ratio is known to work well with the Naïve Bayes learning algorithm.

F. Term frequency Document Frequency (TFDF)

A method based on the term frequency combined with the document frequency threshold (Section 3.6.4) is presented. They call it Term Frequency Document Frequency, and prove it better than DF thresholding.

$$TFDF(F) = \left(n_1 \times n_2 + c\left(n_1 \times n_3 + n_2 \times n_3\right)\right)$$

where c is a constant c >=1, n1 is the number of documents without the feature, n2 is the number of documents where the feature occurs exactly once, n3 is the number of documents where the feature occurs twice or more.

G. NGL Coefficient(NGL)

The NGL coefficient presented in [NGL97] is a variant of the Chi square metric. It was originally named a `correlation coefficient', but we follow Sebastiani [Seb02] and name it `NGL coefficient' after the last names of the inventors Ng, Goh, and Low. The NGL coefficient looks only for evidence of positive class membership, while the chi square metric also selects evidence of negative class membership.

Hence, it is called a `one-sided' chi square metric in [NGL97]. In their experiments, it performed better than chi square. In [RS99] it was better than Odds Ratio and Mutual Information on some feature set sizes, and worse on other.

$$NGL(F,C_k) = \frac{\sqrt{N}\left(N_{F,C_k}N_{\overline{F},\overline{C_k}} - N_{F,\overline{C_k}}N_{\overline{F},C_k}\right)}{\sqrt{N_F N_{\overline{F}} N_{C_k} N_{\overline{C_k}}}}$$

H. GSS Coefficient(GSS)

The GSS coefficient was originally presented in [GSS00] as a `simplified chi square function'. We follow [Seb02] and name it GSS after the names on the inventors Galavotti, Sebastiani, and Simi.

$$GSS(F,C_k) = N_{F,C_k}N_{\overline{F},\overline{C_k}} - N_{F,\overline{C_k}}N_{\overline{F},C_k}$$

The experiments in [GSS00] showed far better results when using max as a globalizing strategy rather than average, hence we follow them on that:

$$GSS(F) = \max_{k=1}^{|C|} GSS(F,C_k)$$

IV. CLASSIFIERS AND DOCUMENT COLLECTIONS

We selected two high-performing classifiers for the feature selection experiments:

- K-Nearest Neighbors
- Naive Bayesian

A. *K-Nearest neighbhor Algorithm*

K-Nearest Neighbor is one of the most popular algorithms for text categorization. Many researchers have found that the k-NN algorithm achieves very good performance in their experiments on different data sets. In pattern recognition, the *k*-nearest neighbor algorithm (*k*-NN) is a method for classifying objects based on closest training examples in the feature space. *k*-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. By simply assigning the property value for the object to be the average of the values of its *k* nearest neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. (A common weighting scheme is to give each neighbor a weight of 1/*d*, where *d* is the distance to the neighbor. This scheme is a generalization of linear interpolation.)

FOR ESTIMATING CONTINUOUS VARIABLES

The *k*-NN algorithm can also be adapted for use in estimating continuous variables. One such implementation uses an inverse distance weighted average of the *k*-nearest multivariate neighbors. This algorithm functions as follows:

1. Compute Euclidean or Mahalanobis distance from target plot to those that were sampled.
2. Order samples taking for account calculated distances.
3. Choose heuristically optimal *k* nearest neighbor based on RMSE done by cross validation technique.
4. Calculate an inverse distance weighted average with the *k*-nearest multivariate neighbors.

B .Naive Bayesian Algorithm

The Naïve Bayesian classifier is a straightforward and frequently used method for supervised learning. It provides a flexible way for dealing with any number of attributes or classes, and is based on probability theory. It is the asymptotically fastest learning algorithm that examines all its training input. It has been demonstrated to perform surprisingly well in a very wide variety of problems in spite of the simplistic nature of the model. Furthermore, small amounts of bad data, or "noise," do not perturb the results by much.

The Naïve Bayesian classification system is based on Bayes' rule and works as follows. There are classes, say C_k for the data to be classified into. Each class has a probability $P(C_k)$ that represents the prior probability of classifying an attribute into C_k; the values of $P(C_k)$ can be estimated from the training dataset. For n attribute values, v_j, the goal of classification is clearly to find the conditional probability $P(C_k \mid v_1 \wedge v_2 \wedge \ldots \wedge v_n)$. By Bayes' rule, this probability is equivalent to

$$\frac{P(v_1 \wedge v_2 \wedge \ldots \wedge v_n \mid C_k)P(C_k)}{P(v_1 \wedge v_2 \wedge \ldots \wedge v_n)}$$

For classification, the denominator is irrelevant, since, for given values of the v_j, it is the same regardless of the value of C_k. The central assumption of Naïve Bayesian classification is that, within each class, the values v_j are all independent of each other. Then by the laws of independent probability,

For classification, the denominator is irrelevant, since, for given values of the v_j, it is the same

regardless of the value of C_k. The central assumption of Naïve Bayesian classification is that, within each class, the values v_j are all independent of each other. Then by the laws of independent probability,

$P(v_i \mid \{\text{all the other values of } v_j\}, C_k) = P(v_i \mid C_k)$ and therefore

$P(v_1 \wedge v_2 \wedge \ldots \wedge v_n \mid Ck) = P(v_1 \mid C_k)P(v_2 \mid C_k)\ldots P(v_n \mid C_k)$.

Each factor on the right-hand side of this equation can be determined from the training data, because (for an arbitrary v_i),

$P(v_i \mid C_k) \approx [\#(v_i \wedge C_k)] / [\#(C_k)]$

where "#" represents the number of such occurrences in the training set data. Therefore, the classification of the test set can now be estimated by,
$P(C_k \mid v_1 \wedge v_2 \wedge \ldots \wedge v_n)$ which is proportional to

$P(C_k)\, P(v_1 \mid C_k)\, P(v_2 \mid C_k)\, P(v_3 \mid C_k) \ldots P(v_n \mid C_k)$.

As mentioned above, the central assumption in Naïve Bayesian classification is that given a particular class membership, the probabilities of particular attributes having particular values are independent of each other.

V. Performance Metric

Precision and recall alone do not say much about the effectiveness of the classifier. Hence, it is necessary to compute different standard values which combine precision and recall, to derive a robust measure of the effectiveness of the classifier. It is able to calculate the breakeven point, the 11-point precision and "average precision" a value which is easy to compute "on the fly", when the first two values are calculated. What the program internally does, is to evaluate the classification for a threshold ranging from 0 (recall = 1) up to a value where the precision value equals 1 and the recall value equals 0, incrementing the threshold with a given threshold step size. The breakeven point is the point where recall meets precision and the eleven point precision is the averaged value fo r the precision at the points where recall equals the eleven values 0.0, 0.1, 0.2, ..., 0.9, 1.0. "Average precision" refines the eleven point precision, as it approximates the area "below" the precision/recall curve:
average precision = Xm i=1 pr(ti) + pr(ti−1) 2 |re(ti) − re(ti−1)| with m is the first value n for which precision(n) = 1 and recall(n) = 0, ti+1 = ti + step size, t0 = 0, re(t0) = 1, and pr(ti) and re(ti) are the precision and recall values for threshold ti. The following integral is the size of the area under the precision/ recall curve, which is approximated by the "average precision" sum:
Z 1 recall=0 pr(recall) d recall where pr(recall) is the precision value corresponding to the recall value recall. But this is just another measure I thought could be interesting to have. In the program, the user can start the computation of these values by giving the threshold step size (the
smaller the more accurate are the results, but the longer takes the calculation) and by clicking on the "Precision/Recall" button. When the computation is finished, the results are written to the console. Additionally, it is possible to open the precision-recall graph (see Figure 1), commonly used in

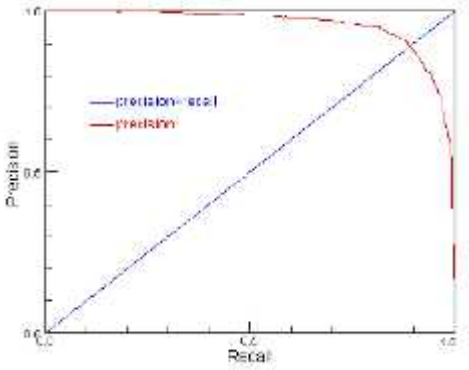

Fig. 1 Precision Recall Graph

text classification literature, in a new window (Menu: Classifiers ! k-nearest neighbor or Naive Bayes (Bernoulli) !
Precision/Recall Graph).

VI. Experiments and Results

Data Set 1: Self Made
For the development used a small self-made corpus since the running time needed to be as short as possible. I collected articles online from the New York Times, Washington Post and CNN.com out of the standard categories
"Science", "Business", "Sports", "Health", "Education", "Travel", and "Movies". This includes easy (e.g. Sports $ Business) and more

difficult (Education $ Science $ Health) classification tasks. I collected 150 documents with the following categories: Sports {30 Training Documents}, Health {30}, Science {27}, Business {23}, Education {24}, Travel {6}, Movies {10}, with in average 702 words per document.

Data Set 2: Self Made
The Reuters 21578 corpus
The second corpus already included in the system is the frequently used Reuters 21578 corpus. The corpus is freely available on the internet (Lewis 1997). uses an XML parser, it was necessary to convert the 22 SGML documents to XML, using the freely available tool SX(Clark 2001). After the conversion I deleted some single characters which were rejected by the validating XML parser as they had decimal values below 30. This does not affect the results since the characters would have been considered as whitespaces anyway.

TABLE I
DIMENSIONALITY OF FEATURE SPACE=500

	Chi square	Information gain	MSF
Break even	0.511	0.686	0.519
11 point Avg	0.561	0.716	0.548
precision	0.563	0.741	0.550

TABLE III
DIMENSIONALITY OF FEATURE SPACE=1000

	Chi square	Information gain	MSF
Break even	0.664	0.696	0.678
11 point Avg	0.690	0.731	0.703
precision	0.708	0.754	0.729

TABLE IIIII
DIMENSIONALITY OF FEATURE SPACE=2000

	Chi square	Information gain	MSF
Break even	0.674	0.683	0.679
11pt Avg	0.712	0.721	0.712
	0.740	0.748	0.735

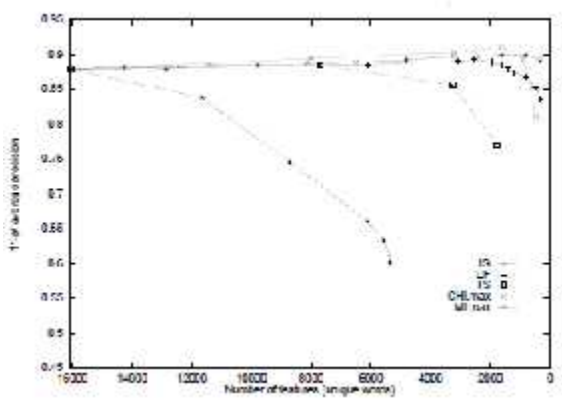

Fig. 2 Average Precision of KNN

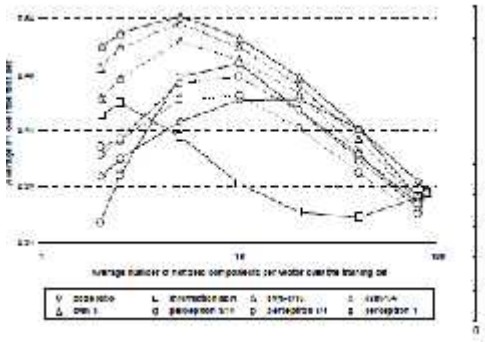

Fig. 3 Naïve Bayesian classifier

VII. CONCLUSION & FUTURE ENHANCEMENT

K-nearest neighbor classification algorithm is proposed that learns importance of attributes and utilizes them in the similarity measure. As our experimental results have shown, our algorithm is very effective in the text categorization task. Some methods adopted the same performance measures i.e., microaveraged accuracy or microaveraged F1. The Naive Bayes algorithm affords fast, highly scalable model building and scoring. Naive Bayes can be used for both binary and multiclass classification problems. As a future work, we need the additional research for applying the more structural information of document to text categorization techniques and testing the proposed method on other types of texts such as newspapers with fixed form. Future works also include testing our method in other inductive transfer models, more intelligent methods of retrieving and using Wikipedia features.

REFERENCES

[1]H. Al-Mubaid and S.A. Umair, "A New Text Categorization Technique Using Distributional Clustering and Learning Logic," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 9, pp. 1156-1165, Sept. 2006.

[2]R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "Distributional Word Clusters versus Words for Text Categorization," J. Machine Learning Research, vol. 3, pp. 1183-1208, 2003.

[3]D. Ienco and R. Meo, "Exploration and Reduction of the Feature Space by Hierarchical Clustering," Proc. SIAM Conf. Data Mining, pp. 577-587, 2008.

[4]H. Kim, P. Howland, and H. Park, "Dimension Reduction in Text Classification with Support Vector Machines," J. Machine Learning Research, vol. 6, pp. 37-53, 2005.

[5]F. Sebastian, "Machine Learning in Automated Text Categorization," ACM Computing Surveys, vol. 34, no. 1, pp. 1-47, 2002.

[6]"The Power of Word Clusters for Text Classification" 23rd European Colloquium on Information Retrieval Research, 2001.

[7]"A Fuzzy Self-Constructing Feature Clustering Algorithm for Text Classification" IEEE transactions on knowledge and data engineering, vol. 23, no. 3, March 2011.