CCSE
Computer and Information Science
www.ccsenet.org/journal.html
Vol. 1, No. 4
November 2008

Text Document Pre-Processing Using the Bayes Formula for Classification Based on the Vector Space Model

Dino Isa
Faculty of Engineering and Computer Science
University of Nottingham, Malaysia Campus
Tel: 60-3-89248116 E-mail: Dino.Isa@nottingham.edu.my

Lam Hong, Lee
Faculty of Engineering and Computer Science
University of Nottingham, Malaysia Campus
E-mail: kcx4lhl@nottingham.edu.my

V. P. Kallimani
Faculty of Engineering and Computer Science
University of Nottingham, Malaysia Campus
Tel: 60-3-89248141 E-mail: VP. Kallimani@nottingham.edu.my

R. Rajkumar
Faculty of Engineering and Computer Science
University of Nottingham, Malaysia Campus
E-mail: Rajprasad.Rajkumar@nottingham.edu.my

Abstract

This work utilizes the Bayes formula to vectorize a document according to a probability distribution based on keywords reflecting the probable categories that the document may belong to. The Bayes formula gives a range of probabilities to which the document can be assigned according to a pre determined set of topics (categories). Using this probability distribution as the vectors to represent the document, the text classification algorithms based on the vector space model, such as the Support Vector Machine (SVM) and Self-Organizing Map (SOM) can then be used to classify the documents on a multi-dimensional level, thus improving on the results obtained using only the highest probability to classify the document, such as that achieved by implementing the naïve Bayes classifier by itself. The effects of an inadvertent dimensionality reduction can be overcome using these algorithms. We compare the performance of these classifiers for high dimensional data.

Keywords: Document Classification, Naïve Bayes, Support Vector Machines, Self-Organizing Map

1. Introduction

Document classification can be defined as the task of automatically categorizing collections of electronic documents into their annotated classes, based on their contents. In recent years this has become important due to the advent of large amounts of data in digital form. For several decades now, document classification in the form of text classification systems have been widely implemented in numerous applications such as spam filtering [5], [6], [7], [23], [25], [27], [31],, e-mails categorizing [16], [32], knowledge repositories [11], and ontology mapping [28], contributed by the extensive and active researches. An increasing number of statistical and computational approaches have been developed for document classification, including decision tree [26], rule induction [26], k-nearest-neighbor classification [10], naïve Bayes classification [20], and support vector machines [15].

Each of the above mentioned document classification schemes have their own unique set of properties. The decision tree induction algorithm and rule induction algorithm are simple to understand and interpret after a brief explanation. However, these algorithms do not work well when the number of distinguishing features is large [26]. *k*-nearest neighbor algorithm is easy to implement and shows its effectiveness in a variety of problem domains [10]. A major drawback of the k-NN algorithm is computationally intensive, especially when the size of the training set grows [10]. Support vector machines (SVM) can be used as a discriminative document classifier and has been shown to be more accurate than most other techniques for classification tasks [4], [15]. The main problem associated with using the support vector machine for document classification is the effort needed to transform text data to numerical data. We call this essential step, the "vectorization" step. Techniques such as TF-IDF vectorize the data easily enough, however, dimensionality then becomes an issue since each vectorized word is used to represent the document. This leads to the number of dimensions being equal to the number of words. For linear classifiers based on the principle of Empirical Risk Minimization, capacity is equal to data dimensionality. For these learning machines, high dimensionality leads to "over-fitting" and the hypothesis becomes too complicated to implement computationally. On the other hand, over fitting does not occur in the SVM since its capacity is equal to the margin of separation between support vectors and the optimal hyper-plane instead of the dimensionality of the data. What is still of concern, however, is the high training time and classification time associated with high dimension data.

The self-organizing map (SOM) is a clustering method which clusters data, based on a similarity measure related to the calculation of Euclidean distances. The idea of this principle is to find a winner-takes-all neuron to find the most closely matching case. The SOM was proposed by Kohonen, and is based on the idea that systems can be designed to emulate the collective co-operation of the neurons in the human brain. Collectivism can be realized by feedback and thus can also be realized in the network, where many neighboring neurons react collectively upon being activated by events. If neurons are activated in the learning process, the neighboring neurons are also affected. The network structure is defined by synapses and has a similar total arrangement after a phase of self organization as the input data of the event space [33]. Consequently, the SOM is an established paradigm in AI and cognitive modeling being the basis of unsupervised learning. This unsupervised machine learning method is widely used in data mining, visualization of complex data, image processing, speech recognition, process control, diagnostics in industry and medicine, and natural language processing [34]. As similar to the SVM, the SOM needs a pre-processing step to transform the text documents into a suitable format for input into the classifier. Hence, the same problem is faced by the SOM, the high dimensionality of vectorized data that represents a text document requires a computational extensive processes during the training and classifying stages.

In this work, we reduce the dimensions from thousands (equal to the number of words in the document, such as when using TF-IDF) to typically less than 20 (number of categories the document may be classified to) through the use of the Bayes formula and then feed this probability distribution to the SVM and SOM for training and classification purposes. With this, we hope to reduce training and classification time to a feasible level (reduce dimensions from thousands to 10 or 20) while maintaining acceptable generalization (using the optimal hyper-plane) and accuracy (10 or 20 dimensions used for classification instead of 1 as is the case with the naïve Bayes classifier).

By itself, the naïve Bayes is a relatively accurate classifier if trained using a large data set. However, as in many other linear classifiers, capacity control and generalization remains an issue. In our case, however, the naïve Bayes is used as a pre-processor in the front end of the classification algorithms based on the vector space model, in our case, the SVM and the SOM, to vectorize text documents before the training and classifying stages are carried out. This is done to improve the generalization of the overall system, while still maintaining a comparatively feasible training and categorization time afforded through the dimensionality reduction imposed by the Bayes formula.

In the context of document classification, the Bayes theorem uses the fact that the probability of a particular document being annotated to a particular category, given that the document contains certain words in it, is equal to the probability of finding those certain words in that particular category, times the probability that any document is annotated to that category, divided by the probability of finding those words in any document, as illustrated in equation (1).

$$\Pr\,(Category \mid Word) = \frac{\Pr(Word \mid Category).\Pr(Category)}{\Pr(Word)} \tag{1}$$

Each document contains words which are given probability values based on the number of its occurrence within that particular document. Naïve Bayes classification is predicated on the idea that electronic documents can be classified based on the probability that certain keywords will correctly identify a piece of text document to its annotated category.

At the basic level, a naïve Bayes classifier examines a set of text documents that have been well organized and categorized, and then compares the content of all categories in order to build a database of words and their occurrences. The database is used to identify or predict the membership of future documents to their right category, according to the probability of certain word occurring more frequently for certain categories. It overcomes the obstacles faced by more static technologies, such as blacklist checking, and word to word comparisons using static databases filled with pre-defined keywords.

In other words, the overall classifier is robust enough to ignore serious deficiencies in its underlying naïve probability model [12], ie, the assumption of the independence of probable events. This robustness is encapsulated in the relative magnitudes of the probabilities (relatively constant for the same category) which is important for the SVM as opposed to an accurate "absolute" value of probability for each category.

In this paper we emphasize the hybrid classification approach which utilizes the simplicity of the Bayes formula as a vectorizer and the good generalization capability of the text classification algorithm based on the vector space model, such as the SVM and the SOM as a classifier. The results of the naïve Bayes-SVM hybrid approach and the naïve Bayes-SOM hybrid approach have been compared to distinguish the performance of both the hybrid classification approaches in terms of the classification effectiveness and efficiency.

2. The Hybrid Classification Approach

We propose, design, implement and evaluate a hybrid classification method by integrating the naïve Bayes vectorizer and text classifiers based on the vector space model to take advantage of the simplicity of the Bayes technique and the accuracy of the SVM and the SOM classification approaches. These hybrid approaches shows an improved classification performance as compared to the traditional naïve Bayes's performance by itself. This has overcome the major drawback of using the traditional naïve Bayes technique alone as a classifier.

The text classification algorithms based on the vector space model are unable to handle the data in text format. An electronic text document may contain a huge number of features such as keywords, which are very useful in terms of the classification task. Therefore, to classify text documents by using the SVM or the SOM, keywords need to be transformed into vectors which are suitable for input to the classifiers.

Generally, the words of a text documents are directly vectorized thereby transforming the text documents into a numerical format. With TF-IDF for example, the huge number of features transformed from the text data of a single document makes running the classifiers a long affair as it takes time to process vectorized data with large dimensions.

Many methods have been carried out to overcome this problem by reducing the dimension of the features. In order to reduce the number of features, we introduce the naïve Bayes as the vectorizer for the text documents by using the probability distribution described earlier, where the dimension of the features is based on the number of available categories in the classification task.

Since the naïve Bayes classifier is able to handle raw text data via the probability distribution calculated from key word occurrence, and the text classifiers based on the vector space model such as the SVM and the SOM typically requires preprocessing to vectorize the raw text documents into numerical values, it is natural to use the naïve Bayes as the vectorizer for the classifiers based on the vector space model. The structure of the proposed classification approach is illustrated as Figure 1.

For the purposes of using the SVM or the SOM as the classifier, after the naïve Bayes vectorizer has been trained, each training document is vectorized by the trained naïve Bayes classifier through the calculation of the posterior probability of the training documents for each existing category based on the Bayes formula. For example, the probability value for a document *X* to be annotated to a category *C* is computed as *Pr(C|X)*. As an assumption that we have a category list as *Cat1*, *Cat2*, *Cat3*, *Cat4*, *Cat5*, ……., *CatN*, thus, each document has *N* associated probability values, where document *X* will have *Pr(Cat1|X)*, *Pr(Cat2|X)*, *Pr(Cat3|X)*, *Pr(Cat4|X)*, *Pr(Cat5|X)*, ……, *Pr(CatN|X)*.

All the probability values of a document are combined to construct a multi-dimensional array, which represents the probability distribution of the document in the feature space. In this way, all the training documents are vectorized by their probability distribution in feature space, in the format of numerical multi-dimensional arrays, with the number of dimensions depending on the number of categories. With this transformation, the training documents are suitable for use in constructing the vectorized training dataset for the classifiers based on the vector space model. Our approach uses the same training dataset for both the naïve Bayes vectorizer and the classifier, where the naïve Bayes vectorizer uses the raw text document for training purposes, and the classifier uses the vectorized training data supplied by the naïve Bayes vectorizer.

As for the classification session, the input to the trained naïve Bayes system is now replaced by the unknown text documents. The output from the naïve Bayes vectorizer, which is the vectorized data of the text documents, in the format of multi-dimensional numerical probability values, is used as the input for the SVM or the SOM for the final classification steps.

2.1 The Naïve Bayes Classification Approach for Vectorization

Our proposed naïve Bayes classifier [13] performs its classification tasks starting with the initial step of analyzing the text document by extracting words which are contain in the document. To perform this analysis, a simple word extraction algorithm is used to extract each individual word from the document to generate a list of words. This list is always helpful when the probabilistic classifier calculating the probability of each word being annotated to each category. The list of words is constructed with the assumption that input document contains words *w 1, w 2, w 3, ………, w n-1, w n,* where the length of the document (in terms of number of words) is *n.*

The list of words is then used to generate a table, containing the probabilities of the word in each category. The column of "Word" is filled with words which are extracted from the input document. For the columns of probabilities of the particular word for each category, the values to be filled will be calculated by the probability classifier in the following stage. The tables below illustrate the use of this method for the input document in Figure 2.

Before the probabilistic classifier performs the calculation of words' probability for each category, it needs to be trained with a set of well-categorized training dataset. Each individual word from all training documents in the same category are extracted and listed in a list of words occurrence for the particular category, by using a simple data structure algorithm.

Based on the list of word occurrence, the trained probabilistic classifier calculates the posterior probability of the particular word of the document being annotated to particular category by using the formula which is shown as equation (2), since each word in the input document contributes to the document's categorical probability.

$$\Pr\,(Category \mid Word) = \frac{\Pr(Word \mid Category).\Pr(Category)}{\Pr(Word)} \tag{2}$$

The derived equation above shows that by observing the value of a particular word, *wj*, the prior probability of a particular category, *Ci*, Pr(*Ci*) can be converted to the posterior probability, Pr(*Ci*|*wj*), which represents the probability of a particular word, *wj* being a particular category, *Ci*. The prior probability, Pr(*Ci*) can be computed from equation (3) or equation (4):

$$\Pr(Category) = \frac{Total\_of\_Words\_in\_Category}{Total\_of\_Words\_in\_Training\_Dataset} \tag{3}$$

$$= \frac{Size\_of\_Category}{Size\_of\_Training\_Dataset} \tag{4}$$

Meanwhile, the evidence, which also known as the normalizing constant of a particular word, *wj*, Pr(*wj*) is calculated by using equation (5):

$$\Pr(Word) = \frac{\sum occurrence\_of\_Word\_in\_every\_category}{\sum occurrence\_of\_all\_words\_in\_every\_category} \tag{5}$$

The total occurrence of a particular word in every category can be calculated by searching the training data base, which is composed from the lists of word occurrences for every category. As previously mentioned, the list of words occurrence for a category is generated from the analysis of all training documents in that particular category during the initial training stage. The same method can be used to retrieve the sum of occurrence of all words in every category in the training data base.

To calculate the likelihood of a particular category, *Ci* with respect to a particular word, *wj*, the lists of words occurrence from the training data base is searched to retrieve the occurrence of *wj* in *Ci*, and the sum of all words in *Ci*. These information will contribute to the value of Pr(*wj* | *Ci*) given in equation (6).

$$\Pr(Word|Category) = \frac{occurrence\_of\_Word\_in\_Category}{\sum occurrence\_of\_all\_words\_in\_Category} \tag{6}$$

Based on the derived Bayes' formula for text classification, with the value of the prior probability Pr(*Category*), the likelihood Pr(*Word*|*Category*), and the evidence Pr(*Word*), the posterior probability, Pr(*Category* | *Word*) of each word in the input document annotated to each category can be measured.

The posterior probability of each word to be annotated to each category is then filled to the appointed cells in the table as illustrated in Figure2. After all the cells of "Probability" have been filled, the overall probability for an input document to be annotated to a particular category, *Ci* is calculated by dividing the sum of each of the "Probability" column with the length of the query, ***n***, which is shown in equation (7).

$$\Pr(Category \mid Document) = \frac{\Pr(\text{Category} \mid \text{w 1, w 2, w 3,......, w n - 1, w n})}{n} \tag{7}$$

where ***w** 1, **w** 2, **w** 3,………, **w** n-1, **w** n*, are the words which are extracted from the input document.

Typically, the ordinary naïve Bayes classifier is able to determine the right category of an input document by referring to the associated probability values calculated by the trained classifier based on the Bayes formula. The right category is represented by the category which has the highest posterior probability value, Pr(*Category*|*Document*), as stated in the Bayes Classification Rule. As the ordinary naïve Bayes classification algorithm has been proven as one of the poor performing classifier, we have extended the classification step for input documents by performing the SVM, for the purpose of increasing the classification accuracy.

2.2 Support Vector Machines for Text Classification

The classification problem here can be restricted to the consideration of the two-class problem without loss of generality. The goal is to produce a classifier that will work well on unseen examples, i.e. it generalizes well. Consider the example in Figure 3. Here there are many possible linear classifiers that can separate the data, but there is only one that maximizes the margin (maximizes the distance between it and the nearest data point of each class). This linear classifier is termed the optimal separating hyperplane.

We will start with the simplest case: linear machines trained on linearly separable data. Therefore consider the problem of separating a set of training vectors $\mathbf{x}_i{}^d$ belonging to different classes y_i {-1, 1}. We wish to separate this training set with a hyperplane [12]:

$$\mathbf{w}.\mathbf{x} + b = 0 \tag{8}$$

There are actually an infinite number of hyperplanes that could partition the data into two sets (dashed lines on Figure 3). According to the SVM principle, there will just be one optimal hyperplane: the hyperplane lying half-way in between the maximal margin (we define the margin as the sum of distances of the hyperplane to the closest training points of each class). The solid line on Figure 3 represents this optimal separating hyperplane, the margin in this case is d1+d2.

Note that only the closest points of each class determine the Optimal Separating Hyperplane. These points are called Support Vectors (SV). As only the Support vectors determine the Optimal Separating Hyperplane they is a certain way to represent them for a given set of training points. It has been shown that the maximal margin can be found by minimizing ½ $||\mathbf{w}||^2$ [12].

$$\min \{½ ||\mathbf{w}||^2 \} \tag{9}$$

The Optimal Separating Hyperplane can thus be found by minimizing (9) under the constraint (10) that the training data is correctly separated.

$$y_i.(x_i.w + b) \;\; . \; 1 \, , \, i \tag{10}$$

The concept of the Optimal Separating Hyperplane can be generalized for the non-separable case by introducing a cost for violating the separation constraints (10). This can be done by introducing positive slack variables $_i$ in constraints (10), which then become:

$$yi.(x_i.w + b) \;\; . \; 1 - \;\; _i \, , \, I \tag{11}$$

If an error occurs, the corresponding $_i$ must exceed unity, so $_i$ $_i$ is an upper bound for the number of classification errors. Hence a logical way to assign an extra cost for errors is to change the objective function (9) to be minimized into:

$$\min \{ ½ ||\mathbf{w}||^2 + C. (\;\; _i \;\; _i) \} \tag{12}$$

where C is a chosen parameter. A larger C corresponds to assigning a higher penalty to classification errors. Minimizing (12) under constraint (11) gives the *Generalized Optimal Separating Hyperplane.* This is a Quadratic Programming (QP) problem which can be solved here using the method of Lagrange multipliers.

After performing the required calculations [12], the QP problem can be solved by finding the LaGrange multipliers, α_i, that maximizes the objective function in (13),

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathrm{x}_\mathrm{i}^\mathrm{T} \mathrm{x}_\mathrm{j} \right) \tag{13}$$

subject to the constraints,

i=1,…..,n, $0 \le \alpha_i \le C,$ and $\sum_{i=1}^{n} \alpha_i y_i = 0.$

The new objective function is in terms of the Lagrange multipliers, α_i only. It is known as the dual problem: if we know **w**, we know all α_i. if we know all α_i, we know **w**. Many of the α_i are zero and so w is a linear combination of a small number of data points. x_i with non-zero α_i are called the support vectors [9]. The decision boundary is determined only by the SV. Let t_j (j=1, ..., s) be the indices of the s support vectors. We can write,

$$w = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} x_{t_j} \tag{14}$$

So far we used a linear separating decision surface. In the case where decision function is not a linear function of the data, the data will be mapped from the input space (i.e. space in which the data lives) into a high dimensional space (feature space) trough a non-linear transformation. In this (high dimensional) feature space, the (Generalized) Optimal Separating Hyperplane is constructed. This is illustrated on Figure 4.

By introducing the kernel function,

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle, \tag{15}$$

it is not necessary to explicitly know Φ (.) [3]. So that the optimization problem (13) can be translated directly to the more general kernel version,

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

subject to

$$C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{16}$$

After the α_i variables are calculated, the equation of the hyperplane, $d(\mathbf{x})$ is determined by,

$$d(x) = \sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x}, \mathbf{x_i}) + b \tag{17}$$

The equation for the indicator function, used to classify new data is given below where the new data **z** is classified as class 1 if *i*>0, and as class 2 if *i* <0 [17].

$$i_F(\mathbf{x}) = \text{sign}\ [\text{d}(\mathbf{x})] = \text{sign}\left[\sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x}, \mathbf{x_i}) + b\right] \tag{18}$$

Note that the summation is not actually performed over all training data but rather over the support vectors, because only for them do the Lagrange multipliers differ from zero. This particular characteristic of the SVM makes it ideal for applications like text classification, where there can be millions of documents. The SVM uses these documents for training and only retains a smaller portion of the entire set as its support vectors. This makes SVM computationally more efficient, needing less memory resources as compared to other techniques like neural networks. However, in a system with the SVM in combination with the naïve Bayes vectorizer multi-dimensional categorization of the documents is possible leading to more accurate results. This is achieved thru the mechanism by which an optimal hyperplane is constructed using only the closest data points or "support vectors".

The SVM uses only some of these documents (represented by feature vectors) for training and retains a small proportion of the entire set as its support vectors. This makes the SVM computationally more efficient, needing less memory resources as compared to other techniques like neural networks which need relatively larger training data sets in order to generalize accurately. This good generalization performance of the SVM is one of the main reasons for using it in text classification. Although millions of documents may be available for training, the burden will be on the high training time. Since the SVM needs a comparatively small training set to isolate the support vectors which form the margins of the optimal hyper plane, training time is reduced compared to other margin classifiers. However, the higher the data dimensionality, the longer the SVM training time and we have addressed this be using the Bayes formula as a vectorizer.

As mentioned previously, another factor which sets the SVM apart is the fact that through its implementation of Structural Risk Minimization, capacity is now independent of data dimensionality. The time to train the SVM is however still proportional to data dimensionality. However, in our case the vectorization process reduces the dimensions of the data to the number of available categories. In vectorization techniques such as TF-IDF however, the number of dimensions is equal to the number of words in that document. As will be shown later, this sacrifice does not detrimentally affect classification accuracy, but buys us a further improved SVM training time which can be significant for large documents. Using this method we find the middle ground between the single dimension classification technique afforded by the naïve Bayes classifier (SVM is multi dimensional) and the high training time needed for the techniques like TF-IDF SVM hybrids (we reduce data dimensionality and therefore training and classification time).

2.3 The Self-Organizing Map for Text Classification

Knowledge discovery tasks can be broken down into two general steps: pre-processing and classification. In the pre-processing step, data is transformed into a format which can be processed by a classifier. The self organizing map (SOM) can be used to carry out the classification tasks effectively, especially for the analysis and visualization of a variety of economical, financial, scientific, and manufacturing data sets [24] [30]. The first step in designing the SOM is to decide on what prominent features are to be used in order to effectively cluster the data into groups. The criterion for selecting the main features plays an important role in ensuring that the SOM clusters properly and thus supports goal based decision making. Traditionally statistical cluster analysis is an important step in improving feature extraction and is done iteratively. An alternative to these statistical methods is the SOM. Kohonen's principle of topographic map formation, states that the spatial location of an output neuron in the topographic map corresponds to a particular feature of the input pattern. The SOM model, which is shown in Figure 5, provides a map which places a fixed number input patterns from an input layer into the so called Kohonen layer [30]. The system learns through self organization of random neurons whose weights are attached to the layers of neurons. These weights are altered at every epoch during the training session. The change depends upon the similarity or neighborhood between the input pattern and the map pattern [34]. The topographic feature maps reduce the dimensions of data to two dimensions simplifying viewing and interpretation.

In the SOM, certain trends in clustering can be observed by changing some of the training parameters. After the incremental training of the map, the application saves the weight vectors of the map and these weights can be used as the starting weights. Once the training is over, the output mean and variance of each cluster is reported. Furthermore, the location of each cluster is also reported. Speed is a big concern in SOM clustering. By reducing dimensionality thru the use of a probability distribution of categories in feature space, instead of a raw word occurrence count we reduce computation time in both training and classification. The concerns associated with data pre-processing before the training starts and final drawing of the map once the training is over is also addressed by our hybrid system.

The SOM is trained iteratively. In each training step, a sample vector, x from the input data set is chosen randomly and the distance between x and all the weight vectors of the SOM, is calculated by using a Euclidean distance measure. The neuron with the weight vector which is closest to the input vector x is called the Best Matching Unit (BMU). The distance between x and weight vectors, is computed by using the equation below:

$$\| x - m_c \| = \min \{\| x_i - m_i \|\}$$

where ||.|| is the distance measure, typically Euclidean distance. After the BMU is found, the weight vectors of the SOM are updated so that the BMU is moved closer to the input vector in the input space. The topological neighbors of the BMU are treated similarly. The update rule for the weight vector of i is:

$$x_i(t+1) = m_i(t) + \alpha(t)\, h_{ci}(t)\, [\, x(t) - m_i(t)]$$

where x(t) is a vector which is randomly drawn from the input data set, and function $\alpha(t)$ is the learning rate and t denotes time. The function $h_{ci}(t)$ is the neighborhood kernel around the winner unit c. The dataset of manufacturing details are fed into the input layer of SOM. Learning parameter is selected between 0.0- 0.9, and the SOM is trained. The training steps will be in the range of 100000 epochs in order to obtain a trained map. These training datasets are coded with reference to their prominent features [30].

3. The Evaluations and Experimental Results

The objective of this evaluation is to determine whether our proposed hybrid approach result in better classification accuracy and performance compared to the ordinary naïve Bayes version. As mentioned in the sections above, the hybrid document classification approach utilizes the simplicity and low requirements of the naïve Bayes classifier as a vectorizer, and the superb generalization performance of the SVM as a classifier. The evaluations are made by comparing the classification accuracy of the ordinary naïve Bayes classifier (along with some specialized techniques with the naïve Bayes vectorizer) and the SVM classification approach which is proposed in this paper.

Initially we performed the experiment by implementing the naïve Bayes algorithm with different ranking schemes: the naïve Bayes with flat ranking algorithm, the naïve Bayes with single elimination tournament ranking algorithm and both the above with the High Relevance Keywords Extraction (HRKE) algorithm. These classification algorithms are discussed briefly below and also in our previous work [13]. In particular the naïve Bayes with flat ranking computes the probability distribution by considering all categories in a single round of competition. The single elimination method entails finding a winner that has not lost even once within the competition. The HRKE algorithm culls out words such as "a", "the" etc which have a low effect on the classification task because it appears in every document. The algorithms mentioned above determine the right category for input documents by referring to the associated probability values calculated by the trained classifier based on the Bayes formula. The right category is represented by the category which has the highest posterior probability value, ***Pr(Category|Document)***.

To evaluate the hybrid classification approach which is proposed in this paper, we implement the naïve Bayes classification algorithms mentioned above in the front end to vectorize raw text data using the associated probability values calculated using the Bayes formula. SVM is used to perform the rest of the classification tasks using the RBF (Radial Basis Function) and the Sigmoid kernels and the results of the classification for the hybrid approach are based on the highest accuracy values.

A dataset of vehicle characteristics extracted from Wikipedia is tested in the prototype system for the evaluation of classification performance in handling a database with four categories which have low degrees of similarity. Our selected dataset contains four categories of vehicles: Aircrafts, Boats, Cars, and Trains. All the four categories are easily differentiated and every category has a set of unique keywords. We have collected 110 documents for each category, with the total of 440 documents in the entire dataset. 50 documents from each category are extracted randomly to build the training dataset for the classifier. The other 60 documents for each category are used as the testing dataset to test the classifier. The results here were seen to be worse than those obtained using cross validation. As such we only present the worse case here. Figure 6 illustrates the table of comparison of the performance between the pure naïve Bayes classification algorithms mentioned above and the hybrid algorithms perform in conjunction with the SVM or the SOM in the back end while classifying the Vehicles dataset.

Looking at the SVM results in comparison with the Bayes classifier, the tournament structure based ranking algorithms show poor performance in terms of classification results. With the proposed hybrid approach, however, significant improvement in classification accuracy is seen since a 20.84% gain is achieved with the hybrid approach measurement compared with the pure naïve Bayes model, where both the hybrid and ordinary approaches were implemented with the single elimination tournament ranking technique.

In this experiment, we also present the classification by using SOM on the Vehicles dataset with 200 training documents. The dataset has four dimensional of information, which is categorized by different methods as shown in Figure 6. The listing of results shows the performances are carried in conjunction with SOM. The vectorized data by the front end naïve Bayes vectorizer are considered as the input data to the SOM for further clustering purposes. The results are tested for 30000 epochs of training cycle. Initial radius of 3 is considered in rough training and radius of 1 is considered in the final training.

The comparison table in Figure 6 illustrates that the recognition rate is poor, when the SOM is trained in combination with the single elimination tournament ranking. The recognition rate is only 56.66%. A similar situation was found when the hybrid approach is used with the flat ranking naïve Bayes classifier. The recognition rate is only 58.00%. The hybrid approach with the naïve Bayes vectorizer when enhanced by High Relevance Keywords Extraction Facility shows good results, with 98.33% recognition rate, a slight improvement over the pure naïve Bayes classifier classification rate.

We have also tested our proposed hybrid classification approaches on the 20 Newsgroups dataset. The 20 Newsgroups dataset is one of the most common datasets used by many text classification research groups to evaluate the performance of their presented classification approaches. The 20 Newsgroups dataset is a collection of 20,000 Usenet articles from 20 different newsgroups with 1,000 articles per newsgroup. In our experiments using this dataset, every category was divided into two subsets. 300 documents from each category were divided for training while the remaining 700 documents were used for testing purposes. Figure 7 illustrates the table of comparison of the performance between the pure naïve Bayes classification algorithms mentioned above and the hybrid algorithms perform in conjunction with the SVM or the SOM in the back end while classifying the 20 Newsgroups dataset.

The comparison table illustrated in Figure 7 has proven again our prediction that the naïve Bayes – SVM hybrid classification approach has increased performance as compared to the naïve Bayes classifiers. Similar to the experiment using the Vehicles dataset, significant improvement of classification accuracy is gained by implementing the naïve Bayes – SVM hybrid approach against the pure naïve Bayes model implemented with tournament ranking methods. An approximately 10% gain is achieved with the naïve Bayes – SVM hybrid approach measurement compared with the pure naïve Bayes model with the implementation of the single elimination tournament ranking method. As for the pure naïve Bayes model which is implemented with the single elimination tournament ranking plus the HRKE facility, a 14.36% gain is achieved by the naïve Bayes – SVM hybrid approach as compared to the pure naïve Bayes model.

The experimental results for the naïve Bayes – SOM hybrid classification approach illustrated in the comparison table shown in Figure 7 have gone against our prediction that the implementation of the hybrid classification approach should show an improvement in the classification performance as compared to the ordinary naïve Bayes classifiers. The naïve Bayes – SOM hybrid approach has greatly reduced the classification accuracy as compared to the naïve Bayes classifiers, in the classification task on the high dimensionality 20 newsgroups dataset.

The main reason why the naïve Bayes – SOM hybrid approach shows worse results with the 20 Newsgroups dataset as compared to the naïve Bayes classifier is that the number of categories for this dataset is high. The SOM algorithm which uses a Euclidean distance measure in determining the best matching units is accurate only for a maximum of four dimensions.

4. Conclusion and Future Works

The hybrid text document classification approach through the implementation of the naïve Bayes method at the front-end for raw text data vectorization, in conjunction with a classifier based on the vector space model, in our case, we have used the SVM and the SOM at the back-end to determine the right category for the input documents, has been proposed and developed by our research group. This hybrid approach takes advantages of both the naïve Bayes system and the classifiers based on the vector space model which greatly enhances the performance of ordinary Bayes classification algorithms. The results from our experiments show that the proposed hybrid approach of naïve Bayes vectorizer and the classifiers based on the vector space model has an improved classification accuracy compared to the pure naïve Bayes classification approach. However, in the cases where the number of categories is high (such as the 20 Newsgroups dataset), the implementation of the classifiers based on the unsupervised learning, such as the SOM is impractical. This is due to the reason that due to the fact that the Euclidean distance measure is inaccurate for dimensions above four as previously stated.

References

McCallum and K. Nigam, *A Comparison of Event Models for Naïve Bayes Text Classification,* Journal of Machine Learning Research 3, pp.1265-1287, 2003.

Kamens, Bayesian Filtering: Beyond Binary Classification, *Fog Creek Software, Inc, 2005*.

Burges, *A Tutorial on Support Vector Machines for Pattern Recognition*, Bell Laboratories, Lucent Technologies, Data Mining and Knowledge Discovery, 1998.

O'Brien and C. Vogel, *Spam Filters: Bayes vs Chisquared; Letters vs Words,* In proceedings of the 1st international symposium on Information and communication technologies, 2002.

Isa, L. H. Lee, V. P. Kallimani and R. Rajkumar, *A Polychotomizer for Case-Based Reasoning Beyond The Traditional Bayesian Classification Approach*, Journal of Computer and Information Science, Canadian Centre of Science and Education (CCSE), 2008.

D. Isa, R. Rajkumar and K. C. Woo, *Defect Detection in Oil and Gas Pipelines using the Support Vector Machines*, In Proceeding the Circuits, Systems, Electronics and Communication Conference, WSEAS, Cairo, Egypt, 2007.

H. Han, G. Karypis and V. Kumar, *Text Categorization Using Weight Adjusted k-Nearest Neighbour Classification*, Department of Computer Science and Engineering, Army HPC Research Center, University of Minnesota, 1999.

Brucher, G. Knowlmayer, and M. A. Mittermayer, *Document Classification Methods for Organizing Explicit Knowledge*, University of Bern, Institute of Information System, Research Group Information Engineering, Engehaldenstrasse 8, CH-3012 Bern, Switzerland, 2002.

J. Provost, *Naïve-Bayes vs Rule-Learning in Classification of E-mail*, Department of Computer Science, The University of Austin, 1999.

J. R. Quinlan, C4.5: *Program for Machine Learning*. Morgan Kaufmann, San Mateo, Ca, 1993.

K. Nigam, J. Lafferty and A. McCallum, *Using Maximum Entropy for Text Classification,* IJCAI-99 Workshop on Machine Learning for Information Filtering pp. 61—67, 1999.

K. Wei, *A Naïve Bayes Spam Filter*, Faculty of Computer Science, University of Berkely, 2003.

M. Hartley, D. Isa, V. P. Kallimani and L. H. Lee, *A Domain Knowledge Preserving in Process Engineering using Self-Organizing Concept*, Intelligent System Group, Faculty of Engineering and Computer Science, University of Nottingham Malaysia Campus, Malaysia, 2006.

M. Negnevitsky, Artificial Intelligennce: *A guide to Intelligent Systems*, 2002.

M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, *A Bayesian Approach to Filtering Junk E-Mail*, In AAAI-98 Workshop on Learning for Text Categorization, 1998.

P. A. Flach, E. Gyftodimos and N. Lachiche, *Probabilistic Reasoning with Terms*, University of Bristol, Loius Pasteur University, Strasbourg, 2002.

P. Cunningham, N. Nowlan, S. J. Delany and M. Haahr, *A Case-Based Approach in Spam Filtering that Can track Concept Drift*, In The ICCBR'03 Workshop on Long-Lived CBR Systems, Trondheim, Norway, 2003.

P. Kontkanen, P. Myllymaki, T. Silander and H. Tirri, *A Bayesian Approach for Retrieving Relevant Cases*, In proceedings of the EXPERSYS-97 Conference, Sunderland, UK, pp. 67-72, 1997.

R. S. Michalski, I. Bratko and M. Kubat, *Machine Learning and Data Mining Methods and Applications*, Wiley, 2002

R. S. Michalski, I. Bratko, and M. Kubat, *Machine Learning and Data Mining Methods and Applications*, Wiley, 1999.

S. B. Kim, H. C. Rim, D. S. Yook and H. S. Lim, *Effective Methods for Improving Naïve Bayes Text Classifiers*, In Proceeding of the 7th Pacific Rim International Conference on Artificial Intelligence, Volume, 2417, 2002.

S. Block, D. Medin and D. Osherson, *Probability from Similarity*, Northwestern University; Rice University, 2002.

S. Chakrabarti, S. Roy and M. V. Soundalgekar, *Fast and Accurate Text Classification via Multiple Linear Discriminant Projection*, The International Journal on Very Large Data Bases (VLDB), pp. 170-185, 2003.

S. Gunn, *Support Vector Machines for Classification and regression*, University of Southampton, ISIS, 1998.

S. H. Wang, *Cluster Analysis using a Validated Self Organizing Method: Cases of Problem Identification*, International Journal of Intelligent Systems in Accounting, Finance and Management, pp 127, 2001.

S. Haykin, *Neural Networks. A Comprehensive Foundation.* 2nd Edition, Prentice Hall, 1999.

S. J. Delany, P. Cunningham and L. Coyle, *An Assessment of Case-Based Reasoning for Spam Filtering,* Journal of Artificial Intelligence Review, Volume 24, Numbers 3-4, pp. 359-378, 2005.

S. J. Delany, P. Cunningham, A. Tsymbal and L. Coyle, *A Case-Based Technique for Tracking Concept Drift in Spam Filtering*, Journal of Knowledge Based Systems, 18 (4-5), Elsevier, pp. 187-195, 2004.

T. Joachims, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features,* In Machine Learning: ECML-98, Tenth European Conference on Machine Learning, pp. 137-142, 1998.

V. A. Petrushin, *Mining rare and frequent events in multi-camera surveillance video using self organizing maps,* In Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005.

V. Kecman, *Support Vector Machines Basics*, School Of Engineering Report 616, University of Auckland, 2004.

V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, NewYork, 1995.

X. Su, *A Text Categorization Perspective for Ontology Mapping*, Department of Computer and Information Science, Norweigian University of Science and Technology, Norway, 2002.

Y. Xia, W. Liu, and L. Guthrie, *Email Categorization with Tournament Methods*, In Proceeding International Conference on Application of Natural Language, 2005.

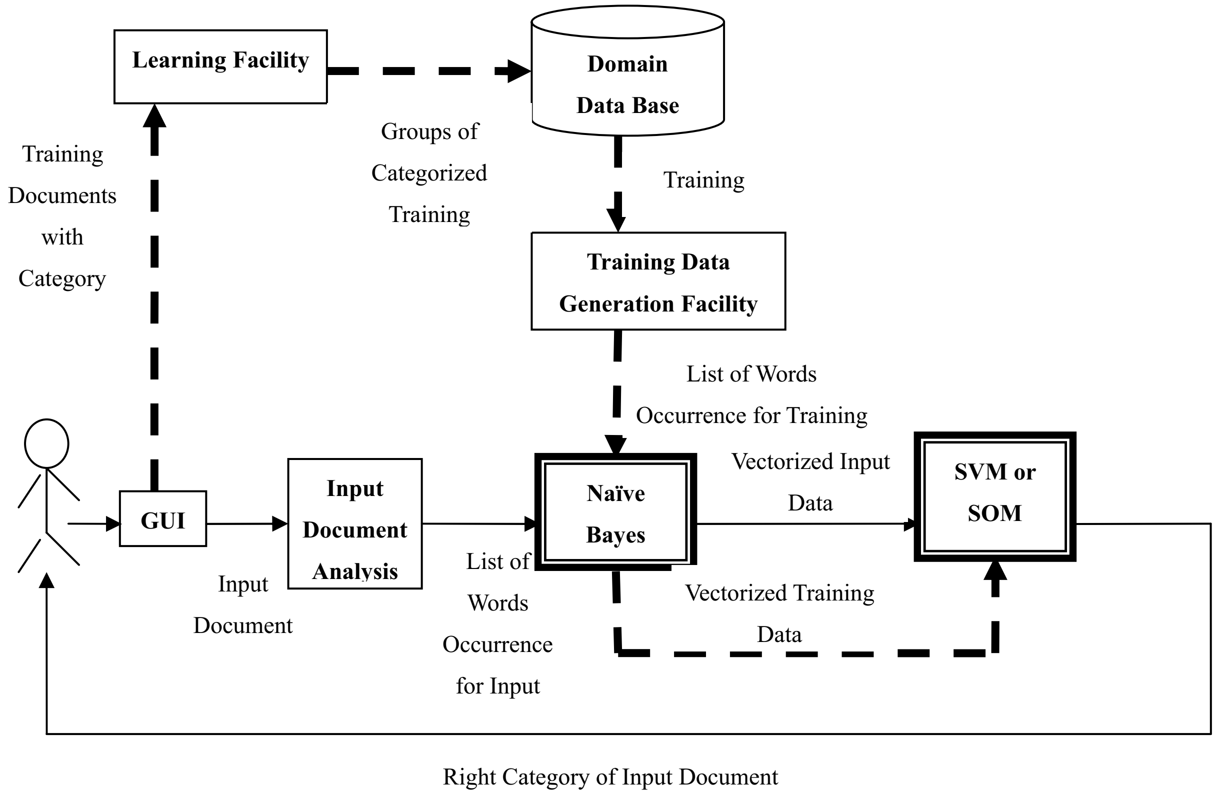
Learning Facility
Domain Data Base
Groups of Categorized Training
Training Documents with Category
Training
Training Data Generation Facility
List of Words Occurrence for Training
Vectorized Input Data
SVM or SOM
GUI
Input Document Analysis
Naïve Bayes
Input Document
List of Words Occurrence for Input
Vectorized Training Data
Right Category of Input Document

Figure 1. Block diagram of proposed hybrid classification approach.

Word	*Probability* *Category 1*	*Probability* *Category 2*	*Probability* *Category 3*	… …	*Probability* *Category k-1*	*Probability* *Category k*
w 1	Pr(*C1*|*w1*)	Pr(*C2*|*w1*)	Pr(*C3*|*w1*)	…	Pr(*C k*-1|*w1*)	Pr(*C k*|*w1*)
w 2	Pr(*C1*|*w2*)	Pr(*C2*|*w2*)	Pr(*C3*|*w2*)	…	Pr(*C k*-1|*w2*)	Pr(*C k*|*w2*)
w 3	Pr(*C1*|*w3*)	Pr(*C2*|*w3*)	Pr(*C3*|*w3*)	…	Pr(*C k*-1|*w3*)	Pr(*C k*|*w3*)
w n-1	Pr(*C1*|*w n-1*)	Pr(*C2*|*w n-1*)	Pr(*C3*|*w n-1*)	…	Pr(*C k*-1|*w n-1*)	Pr(*C k*|*w n-1*)
w n	Pr(*C1*|*wn*)	Pr(*C2*|*wn*)	Pr(*C3*|*wn*)	…	Pr(*C k*-1|*wn*)	Pr(*C k*|*wn*)
Total	∑Pr(*C1*|*W*)	∑Pr(*C2*|*W*)	∑Pr(*C3*|*W*)	…	∑Pr(*C k*-1|*W*)	∑Pr(*C k*|*W*)

Figure 2. Table of words occurrence and probabilities.

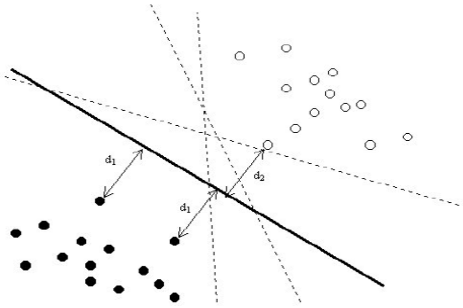

Figure 3. Optimal Separating Hyper-plane.

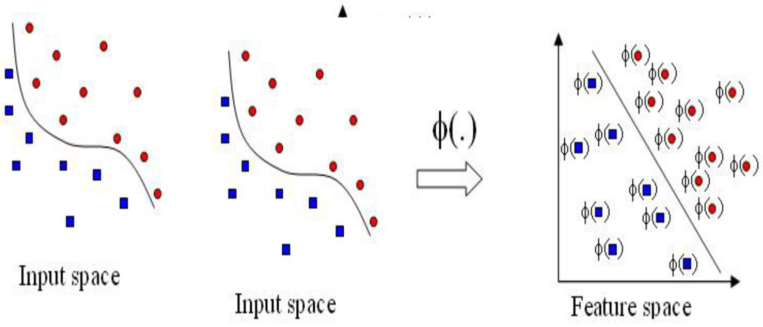

Figure 4. Mapping onto higher dimensional feature space.

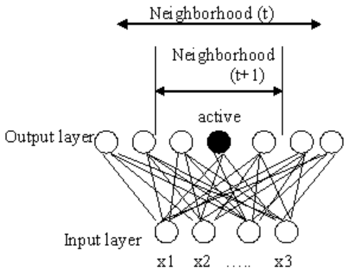

Figure 5. The SOM model.

Classification Optimization Techniques	Naïve Bayes (Accuracy)	Naïve Bayes – SVM Hybrid (Accuracy)	Naïve Bayes – SOM Hybrid (Accuracy)
Flat Ranking	81.25%	85.42%	58.00%
Flat Ranking with HRKE Facility	96.25%	94.58%	98.33%
Single Elimination Tournament Ranking	64.58%	85.42%	56.66%
Single Elimination Tournament Ranking with HRKE Facility	76.66%	85.83%	57.00%

Figure 6. Comparison table for classification accuracy using different methods

(Dataset: Vehicles-Wikipedia).

Classification Optimization Techniques	Naïve Bayes (Accuracy)	Naïve Bayes – SVM Hybrid (Accuracy)	Naïve Bayes – SOM Hybrid (Accuracy)
Flat Ranking	70.13%	78.99%	20%
Flat Ranking with HRKE Facility	77.82%	79.55%	34%
Single Elimination Tournament Ranking	40.68%	50.55%	27%
Single Elimination Tournament Ranking with HRKE Facility	65.05%	79.41%	14%

Figure 7. Comparison table for classification accuracy using different methods (Dataset: 20 Newsgroups).