An Efficient Mining Model For Enhancing Text Classification Using k-NN

T.SIVAKUMAR
Assistant Professor
Department of Computer Science and Engineering
Maharaja Institute of Technology
Coimbatore

A.KALEESWARAN
Assistant Professor
Department of Computer Science and Engineering
Park College of Engineering and
Tekhnology, Coimbatore

***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 project, apply machine learning methods for classification. In this regard, first try to exert some text pre-process in different dataset, and then extract a feature vector for each new document by using feature weighting and feature selection algorithms for enhancing the text classification accuracy. After that train our classifier by Naïve Bayesian (NB) and K-nearest neighbor (KNN) algorithms. In experiments, both algorithms show acceptable results for text classification.**

***Index Terms*—Text processing, classification, vector construction, Naïve Bayesian, K-nearest neighbor.**

I.INTRODUCTION

Text Classification or Categorization, the problem of automatically assigning semantic categories to natural language text, has become one of the most important methods for organizing textual information. Since the classification by hand is costly and in most cases highly unpractical due to the increasing number of documents and categories in many corpora, most state of the art approaches employ machine learning techniques to automatically learn text classifiers from training examples. Unlike many other classification tasks, text classification involves also preprocessing steps, eg, stemming and dimensionality reduction, which have an important influence on the effectiveness of the actual classification outcome.

Categorizing text documents means to discover their category or topic from a set of predefined categories, eg. `sports' or `economics'. Text categorization is an important field within natural language processing. Its application areas are many and the need for them is increasingly important as the amounts of information continue to grow. Junk mail filtering has been an important area for text categorization the last decade, as have portals with hierarchies of web sites, digital libraries and more. But the general task of placing a text document in the correct location or spotting its correct topic will exist as long as digital written texts are being produced. Other examples include publishing newspaper articles in the correct category or storing a digital document correctly in an archive or library. Automatic text categorization was first done as early as the sixties, though the lack of computer power made it infeasible for a long time. During the last decade or so however, we have seen a lot of efforts in the area. While computers today are capable of learning and performing text categorization within reasonable time limits, growing amounts of data makes TC challenging today as well. When classifying text documents one considers the features of a document, typically these correspond to terms. Not all features are equally helpful for deciding which category a document belongs to. One can say that they convey less information, while some features may even be regarded as noise. Selecting a good subset of these features has emerged as a research field itself, named feature selection. Selecting a subset of features can give both huge savings in computation time and increase in accuracy. Many methods for ranking and selecting features have been presented, and the main task of this assignment is to compare promising methods in the same system. Feature selection methods have been compared before, but the number of methods compared in each paper is often sparse. Also, several methods have been presented and there is a so far unfulfilled need to compare these against each other and the classic methods.

Text categorization (also known as *text classification*) is, quite simply, the automated assignment of natural language texts to predefined categories based on their content. Its applications include indexing texts to support document

retrieval, extracting data from texts, and aiding humans in these tasks. The performance of standard text categorization techniques on standard test corpora has been quite encouraging. For example, reported an 87.8% precision/recall breakeven point for the Reuters-21578 corpus.

Nearest Neighbor Analysis is a method for classifying cases based on their similarity to other cases. In machine learning, it was developed as a way to recognize patterns of data without requiring an exact match to any stored patterns, or cases. Similar cases are near each other and dissimilar cases are distant from each other. Thus, the distance between two cases is a measure of their dissimilarity. Cases that are near each other are said to be "neighbors." When a new case (holdout) is presented, its distance from each of the cases in the model is computed. The classifications of the most similar cases – the nearest neighbors – are tallied and the new case is placed into the category that contains the greatest number of nearest neighbors. specify the number of nearest neighbors to examine; this value is called k. The pictures show how a new case would be classified using two different values of k. When k = 5, the new case is placed in category 1 because a majority of the nearest neighbors belong to category 1. However, when k = 9, the new case is placed in category 0 because a majority of the nearest neighbors belong to category 0.

Nearest neighbor analysis can also be used to compute values for a continuous target. In this situation, the average or median target value of the nearest neighbors is used to obtain the predicted value for the new case.

- There are some noise reduction techniques that work only for k-NN that can be effective in improving the accuracy of the classifier.
- In situations where an explanation of the output of the classifier is useful, k-NN can be very effective if an analysis of the neighbors is useful as explanation.
- The Naive Bayes algorithm affords fast, highly scalable model building and scoring. It scales linearly with the number of predictors and rows. The build process for Naive Bayes is parallelized. (Scoring can be parallelized irrespective of the algorithm.
- Naive Bayes can be used for both binary and multiclass classification problems.

II. PROBLEM DEFINITION

The short version of the assignment text reads: Information retrieval and text mining methods operate on the terms found in text documents. As such, every term found in a collection is analyzed and used for further processing. The process of feature selection is performed in order to reduce the number of terms to be used in further analysis i.e. to identify the most important terms beforehand. The task of this project is to compare a range of feature selection techniques with the goal of a thorough performance evaluation. The main goal of the assignment is evidently to compare several feature selection techniques.

The main purpose of the program is to provide a framework to which more classifiers (eg. neural network classification, other statistical methods, case- and rule-based systems) can easily be added, and to give the user the opportunity to compare and evaluate different preprocessing techniques like stemming, term weighting, and dimensionality reduction. In research areas where quality is determined almost entirely based on empirical results, the standardization of every step from preprocessing to classification is essential. Furthermore, can aid the exploration and analysis of corpora using term/document/category tables and graphical tools.

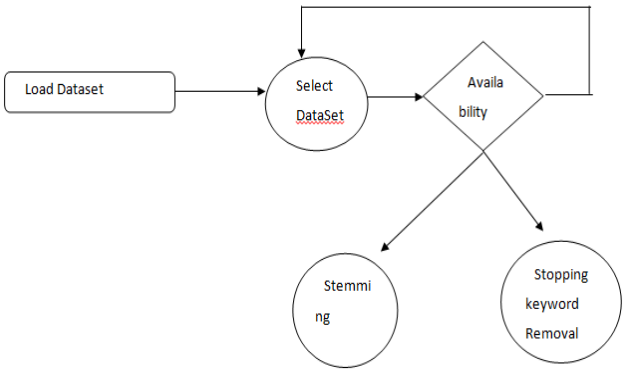
III. PROBLEM SOLVING

1.Load data set

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.

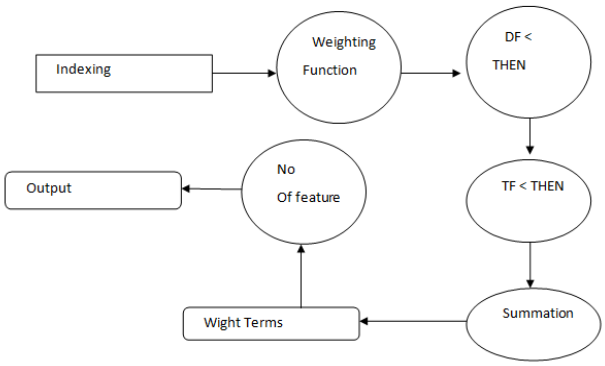
The Reuters 21578 corpus

The second corpus already included in the system is the frequently used Reuters 21578 corpus. The corpus is freely available on the internet uses an XML parser, it was necessary to convert the 22 SGML documents to XML, using the freely available tool SX. 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.



2.Text preprocessing

In most of the applications, 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 like "be" and "go". If the box "Apply stop word removal" is checked, all the words in the file "swl.txt" are considered as stop words and will not be loaded. This file contains currently the 100 most used words in the English language which on average account for a half of all reading in English. If the box "Apply stop word removal" is unchecked, the stop word removal algorithm will be disabled when the corpus is loaded.



Stemming

Stemming or lemmatization is a technique for the reduction of words into their root. Many words in the English language can be reduced to their base form or stem e.g. agreed, agreeing, disagree, agreement and disagreement belong to agree. Furthermore are names transformed into the stem by removing the " s". The variation "Peter's" in a sentence is reduced to "Peter" during the stemming process. The result of the removal may lead to an incorrect root. However, these stems do not have to be a problem for the stemming process, if these words are not used for human interaction. The stem is still useful, because all other inflections of the root are transformed into the same stem. Case sensitive systems could have problems when making a comparison between a word in capital letters and another with the same meaning in lower case. Following a selection of suffixes and prefixes for removal during stemming

- **suffixes:** ly, ness, ion, ize, ant, ent , ic, al , ical, able, ance, ary, ate, ce, y, dom , ed, ee, eer, ence, ency, ery, ess, ful, hood, ible, icity, ify, ing, ish, ism, ist, istic, ity, ive, less, let, like, ment, ory, ty, ship, some, ure
- **prefixes:** anti, bi, co, contra, counter, de, di, dis, en, extra, in, inter, intra, micro, mid, mini, multi, non, over, para, poly, post, pre, pro, re, semi, sub, super, supra, sur, trans, tri, ultra, un.

However, most stemming algorithms do not remove the prefix of a term. The reason of this is the huge impact for the meaning of a sentence. For instance, stemming the word nonhazardous to hazardous is a parlous change. There are different types of stemming methods. The simplest one is the brute force method. This method requires a dictionary which contains the inflections of a word. The dictionary is used as a lookup table. This approach has some serious disadvantages. Firstly the speed for the word ascription is very low and the whole stemming process requires many resources in storage. This is the result of a missing algorithm which could increase the transformation speed. The other aggravating disadvantage is the problem that the look-up table usually does not contain all inflections for each root. The need for a comprehensive dictionary is fundamental for acceptable results. The quality of the result is directly derived from it. Nevertheless, brute force solutions are used for languages with a higher grammatical complexity. The English language has quite simple inflections which can be easily stemmed via an algorithm. However, languages such as Romanic languages (French, Spanish, Italian, Portuguese, etc.) have inflections with a change of the root of a word. Pre- and suffix removing algorithms do not have the capability to handle this kind of stemming problem. The solution is often a stemming process which uses a suffix stripping algorithm combined with one or more dictionaries. This combination reduces the disadvantages which each would cause if used separate.

The outcome of the stemming process always requires the right balance. Neither too much nor too less stemming is a benefit for Information Retrieval (IR). A small set of terms will lead to less accurate relations between documents with many connections. In contrast to this a large set of terms will enable very accurate relations between documents, but only few connections.

Porter stemming algorithmThe idea of this algorithm is the removal of all pre- and suffixes to get the root of a

word. The main field of application for the Porter Stemmer is languages with simple inflections, such as English. The algorithm is favored and often used because of the simplicity and the small amount of rules.Following an explanation of the algorithm, based on the publication of Martin F. Porter. The algorithm makes a distinction between consonants and vowels in a word. Therefore the selection of the applying rules during the stemming process is based on the sequence of consonants and vowels.

A word is represented by the form

[C]VCVC ... [V]

Where the notation of a sequence of VC is written as (VC) {m}, with VC repeated m times. An example for a repetition with m = 0 is sea, for m = 1 is cat, for m = 2 is garden and so on.

The further processing of the suffix stripping is decided by several conditions. One of the conditions was mentioned in the sentences before, the repletion of VC in a word.

The other conditions for the Porter Stemming are:

- *S - the stem ends with S (and similarly for the other letters).
- *v* - the stem contains a vowel.
- *d - the stem ends with a double consonant (e.g. -TT, -SS).
- *o - the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP).

Furthermore, combinations of these conditions are possible (using and, or and not). Following, the rules with some examples, divided into 5 steps. Only the application of one rule for a step is allowed. This rule has to remove the longest matching suffix.

Lancaster stemming : Stemming is a well-known technique for information retrieval. The use of stems for searching has the advantage of increasing recall by retrieving terms that have the same roots but different endings. A major disadvantage of stemming is a decrease of precision as compared to the use of truncated terms. When searching with stems, it is not uncommon to retrieve many irrelevant terms that have similar roots but which are not related to the object of the search. For accurate retrieval, the search stems should be as long as necessary to achieve precision, but short enough to increase recall. Several commonly-used stemming programs and algorithms were evaluated to try to select a stemmer suitable for information retrieval of large databases. The evaluation was narrowed down to two stemmers: 1) the Paice/Husk stemmer developed at Lancaster University which features a rule execution mechanism and externally stored rules, and 2) the Porter stemmer which uses algorithmic rules rather than externally stored rules. Neither of these stemmers could be used in their original form because some of the stems generated were not substrings of actual words or the resulting stems were too short. Both of these are important requirements for accurately searching existing large databases.

The flexibility of being able to specify a new set of rules without extensive programming changes made the Paice/Husk stemmer more attractive than the Porter stemmer. The Paice/Husk stemmer is basically a rewrite rule interpreter which may be configured as a finite state automaton by using the appropriate rules. The C-language implementation by Andrew Stark of the Paice/Husk stemmer works adequately, but is not well suited for developing and experimenting with a new set of rules. Consequently, the program was modified to improve the handling of errors in the rules, allow interactive testing, provide more precise stems, and add some flexibility for implementing finite state automata. Fewer than 50 lines of code were added or altered without counting the replacement of the driver. The new driver and debugging options make it possible to test the execution of the rules interactively. This is important because it is possible for the execution of the rules to get in an infinite loop! For example, the rule "e,e,continue" will loop forever when a word ending in "e" is input. The interaction of several rules may also result in infinite loops when they all use the *continue* flag. The code was modified to prevent infinite loops by stopping when the number of rules executed exceeds twice the number of characters in the input word. The new debugging options helped to solve the mystery of why the original rules generated the stem "abud" from "abusively":

<abusively> 100->abusive 13->abusiv 94->abuj 27->abud

Affix removal conflation techniques are referred to as stemming algorithms and can be implemented in a variety of different methods. All remove suffices and/or prefixes in an attempt to reduce a word to its stem.. The algorithms that are discussed in the following sections, and those that will be implemented in this project, are all suffix removal stemmers.During the development of a stemmer the issues of iteration and context awareness must be addressed. Suffices that are concatenated to words are often done so in a certain order, such that a set of order-classes will exist among suffices. An iterative stemming algorithm will remove suffices one at a time, starting at the end of the word and working towards the beginning. An issue also exists about whether a stemmer should be context-free or

context-sensitive. A context-sensitive algorithm involves a number of qualitative contextual restrictions that are developed to prevent the removal of endings that, in certain situations, can lead to erroneous stems being produced. A context free algorithm removes endings with no restrictions placed on the circumstances of the removal.

3.Feature weighting and reduction

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 odds for a feature to occur in one category is exactly the same as the odds for it to occur in the other, since in (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.

Information gain

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:

$IG(t) = H(C) - H(C|T) = \Sigma\tau,c\ P(C=c,T=\tau) \ln[P(C=c,T=\tau)/P(C=c)P(T=\tau)]$.

Here, τ ranges over {present, absent} and c ranges over {c+, c–}. As pointed out above, this is the amount of information about C (the class label) gained by knowing T (the presence or absence of a given word).

Document frequency (df) thresholding

One of the simplest methods of vocabulary reduction, and hence vector dimensionality reduction, is the Document Frequency Thresholding,

$DF(F) = N_F$

The number of documents containing a feature in the training set is counted. This is done for every feature in the training set, before removing all features with a document frequency less than some specified threshold and features with a frequency higher than some other threshold. Alternatively, the document frequency can be used as any other feature selection method where it creates a ranked list, and returns the highest ranked features.

The document frequency values for our e-mail example can be read directly from Table 4.1. Ranks the e-mail example features according to their document frequency value. Note that document frequency values are naturally global, so there is no need to aggregate them in any way.

Feature	**Document Frequency Value**
wigra	5.0
save	3.0
erection	3.0
ski	3.0
Cell	2.0

Table 4.1 Document Frequency

Term frequency document frequency (tfdf)

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

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

where c is a constant c >=1, n1 is the number of documents without the feature, n_2 is the number of documents where the feature occurs exactly once, n_3 is the number of documents where the feature occurs twice or more.Use c = 10 in their experiments, and we follow this decision in our experiments. It should be noted however, that the constant can highly affect the results. Hence, in an operational setting, performance should be measured for

several levels of the constant, with the actual text collection and classification learner at hand, as to achieve the most from this feature selection method.

Mutual information

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.

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

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:

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

chi square (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.

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

NGL coefficient

The NGL coefficient presented is a variant of the Chi square metric. It was originally named a `correlation coefficient', but we follow Sebastiani 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 their experiments, it performed better than chi square. It was better than Odds Ratio and Mutual Information on some feature set sizes, and worse on other.

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

GSS coefficient

The GSS coefficient was originally presented as a `simplified chi square function'. We follow Sebastiani 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 showed far better results when using max as a globalizing strategy rather than average, hence we follow them on that:

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

4.Text classification of k-NN classifier

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. The *k*-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of its nearest neighbor.The same method can be used for regression, 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.)

The neighbors are taken from a set of objects for which the correct classification (or, in the case of regression, the value of the property) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. The *k*-nearest neighbor algorithm is sensitive to the local structure of the data.Nearest neighbor rules in effect compute the decision boundary in an implicit manner. It is also possible to compute the decision boundary itself explicitly, and to do so in an efficient manner so that the computational complexity is a function of the boundary complexity.

Algorithm

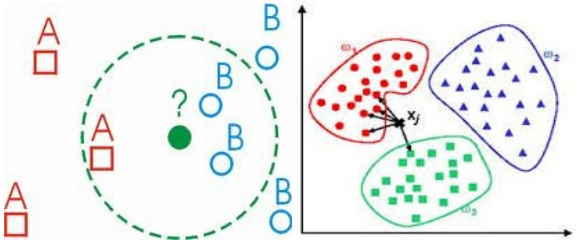


Figure: 4.1 Example of *k*-NN classification.

The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If *k = 3* it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If *k = 5* it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.In the classification phase, *k* is a user-defined constant, and an unlabelled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.Usually Euclidean distance is used as the distance metric; however this is only applicable to continuous variables. In cases such as text classification, another metric such as the overlap metric (or Hamming distance) can be used. Often, the classification accuracy of k-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighborhood components analysis.k-NN is a special case of a variable-bandwidth, kernel density "balloon" estimator with a uniform kernel.

Parameter selection : The best choice of *k* depends upon the data; generally, larger values of *k* reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good *k* can be selected by various heuristic techniques, for example, cross-validation. The special case where the class is predicted to be the class of the closest training sample (i.e. when *k* = 1) is called the nearest neighbor algorithm.The accuracy of the *k*-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into selecting or scaling features to improve classification. A particularly popular approach is the use of evolutionary algorithms to optimize feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes.In binary (two class) classification problems, it is helpful to choose *k* to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal *k* in this setting is via bootstrap method.

Properties : The naive version of the algorithm is easy to implement by computing the distances from the test sample to all stored vectors, but it is computationally intensive, especially when the size of the training set grows. Many nearest neighbor search algorithms have been proposed over the years; these generally seek to reduce the number of distance evaluations actually performed. Using an appropriate nearest neighbor search algorithm makes *k*-NN computationally tractable even for large data sets. The nearest neighbor algorithm has some strong consistency results. As the amount of data approaches infinity, the algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data). *k*-nearest neighbor is guaranteed to approach the Bayes error rate, for some value of *k* (where *k* increases as a function of the number of data points). Various improvements to *k*-nearest neighbor methods are possible by using proximity graphs.

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.

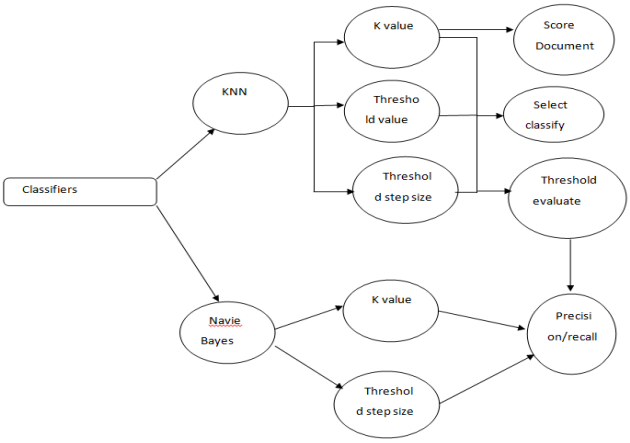
Using a weighted k-NN also significantly improves the results: the class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance between that point and the point for which the class is to be predicted.

Naive Bayes classifier

A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model".In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without believing in Bayesian probability or using any Bayesian methods.In spite of their naive design and apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, analysis of the Bayesian classification problem has shown that there are some theoretical reasons for the apparently unreasonable efficacy of naive Bayes classifiers. Still, a comprehensive comparison with other classification methods in 2006 showed that Bayes classification is outperformed by more current approaches, such as boosted trees or random forests.An advantage of the naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

Parameter estimation : All model parameters (*i.e.*, class priors and feature probability distributions) can be approximated with relative frequencies from the training set. These are maximum likelihood estimates of the probabilities. A class' prior may be calculated by assuming equiprobable classes (i.e., priors = 1 / (number of classes)), or by calculating an estimate for the class probability from the training set (i.e., (prior for a given class) = (number of samples in the class) / (total number of samples)). To estimate the parameters for a feature's distribution, one must assume a distribution or generate nonparametric models for the features from the training set. If one is dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution.

Sample correction : If a given class and feature value never occur together in the training set then the frequency-based probability estimate will be zero. This is problematic since it will wipe out all information in the other probabilities when they are multiplied. It is therefore often desirable to incorporate a small-sample correction in all probability estimates such that no probability is ever set to be exactly zero.



Constructing a classifier from the probability model : The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or *MAP* decision rule.

IV.DISCUSSION

Despite the fact that the far-reaching independence assumptions are often inaccurate, the naive Bayes classifier has several properties that make it surprisingly useful in practice. In particular, the decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality, such as the need for data sets that scale exponentially with the number of features. Like all probabilistic classifiers under the MAP decision rule, it arrives at the correct classification as long as the correct class is more probable

than any other class; hence class probabilities do not have to be estimated very well. In other words, the overall classifier is robust enough to ignore serious deficiencies in its underlying naive probability model. Other reasons for the observed success of the naive Bayes classifier are discussed in the literature cited below.

V.CONCLUSION

In this paper presented a range of novel algorithms for solving real-world text classification problems arising in different situations. The main focus of our work has been exploiting the notion of inter-class relationships in text classification systems. Proposed four inter-class relationships and developed algorithms based on them for different tasks. I learned mappings between label-sets to build better classifiers and developed Java data mining tools. Exploited the confusion between related classes to handle scalability issues in large-scale multi-class classification. Overcame the problem of overlapping class boundaries by proposing enhancements to discriminative classifiers for multi-labeled classification.

VI.FUTURE WORK

This work leads to some interesting avenues of future work that someone would like to explore. It is possible to theoretically understand cross-training better and devise formal ways of studying related label-sets. I would like to extend my work in detecting evolving label-sets to larger scales and devise ways to track other kinds of evolution in label-sets apart from detecting new classes. The most exciting direction of this work is the idea of building next-generation text classification platforms which could be used for research as well as real world deployment. Studying this under a formal framework thus leading to guarantees about the platform is a promising line of work.

REFERENCES

[1] Jung-Yi Jiang, Ren-Jia Liou, and Shie-Jue Lee, "A Fuzzy Self-Constructing Feature Clustering Algorithm for Text Classification", Member, IEEE March 2011
[2] Dalmau M.C and Florez O.W.M, "Experimental Results of the Signal Processing Approach to Distributional Clustering of Termson Reuters-21578 Collection," Proc. 29th European Conf. IR Research, pp. 678-681, 2007
[3] Dhillon I.S, Mallela S, and Kumar R, "A Divisive Information Theoretic Feature Clustering Algorithm for Text Classification," J. Machine Learning Research, vol. 3, pp. 1265-1287, 2003.
[4] Hisham Al-Mubaid and Syed A. Umair, "A New Text Categorization Technique Using Distributional Clustering and Learning Logic", IEEE transactions on knowledge and data engineering, vol. 18, no. 9, September 2006.
[5] Ienco D and Meo R, "Exploration and Reduction of the Feature Space by Hierarchical Clustering," Proc. SIAM Conf. Data Mining, pp. 577-587, 2008
[6] Kim .H, Howland P and Park H, "Dimension Reduction in Text Classification with Support Vector Machines," J. Machine Learning Research, vol. 6, pp. 37-53, 2005
[7] Lewis D.D, "Feature Selection and Feature Extraction for Text Categorization," Proc. Workshop Speech and Natural Language, pp. 212-217, 2005
[8] Oja E, Subspace Methods of Pattern Recognition. Research Studies Press, 2009
[9] SaulL.K. and Roweis S.T, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," Science, vol. 290, pp. 2323-2326, 2000
[10] Sebastiani F, "Machine Learning in Automated Text Categorization," ACM Computing Surveys, vol. 34, no. 1, pp. 1-47, 2002.
[11] Shawe-Taylor J and Cristianini N, Kernel Methods for Pattern Analysis. Cambridge Univ. Press, 2004
[12] Slonim N and Tishby N, "The Power of Word Clusters for Text Classification," Proc. 23rd European Colloquium on Information Retrieval Research (ECIR), 2001
[13] Tenenbaum J.B,de Silva V, and Langford J.C, "A Global Geometric Framework for Nonlinear Dimensionality Reduction", Science, vol. 290, pp. 2319-2323, 2008
[14] Yang Y and Liu X, "A Re-Examination of Text Categorization Methods," Proc ACM SIGIR, pp. 42-49, 2004
[15] Yen J and Langari , Fuzzy Logic-Intelligence, Control and Information, Prentice-Hall, 2010.

Sivakumar.T is an Assistant Professor in Department of Computer Science and Engineering, Maharaja Institute of Technology. He received his Bachelor of computer science and Master of Computer Applications from Bharathiar University and Master of Engineering in Anna University. He has 5 years of teaching experiences from various Colleges and 1 year Industrial experience. He has many publications to his credit in various international conferences and journals. He has attended a number of conferences, seminars and workshops His area of interest includes Data mining, Artificial neural network.

Kaleeswaran.A received his B.Sc (Physics) and MCA from Bharathiar University, Coimbatore and M.E (SE) in Anna University Coimbatore. He has four years of teaching experience. He is now working as a Assistant Professor in Department of Computer Science and Engineering in Park College of Engineering and Technology, Coimbatore. He has many publications to his credit in various international conferences and journals. He has attended a number of conferences, seminars and workshops His research interest is mainly focused on Data mining and Image Processing.

