A COMPARATIVE ANALYSIS OF CLASSIFICATION ALGORITHMS FOR TEXT CLASSIFICATION USING FUZZY SELF-CONSTRUCTING FEATURE

J.SARANYA

10PCA19

A Project work submitted to Avinashilingam Deemed University for Women,Coimbatore in partial fulfillment of the requirement for the Master's Degree in Computer Applications

MAY 2013

ACKNOWLEDGEMENT

I would like to express our sincere thanks to God Almighty, for his constant love and grace that has showered upon me.

I are very grateful to **Dr.T.S.K.MeenakshiSundaram, M.A.,M.Phil, Chancellor**, Avinashilingam Deemed University for Women, Coimbatore for his support and encouragement during the course of my study.

I heartily thank **Dr. (Mrs). Sheela Ramachandran M.Sc., P.G. Dip., Ph.D., Vice Chancellor,** Avinashilingam Deemed University for Women, Coimbatore, for extending all resources that facilitated the conduct of the present study.

I express our humble gratitude **to Dr. (Mrs). GowriRamakrishnan M.Sc., M.Phil, Ph.D., Registrar**, Avinashilingam Deemed University for Women,Coimbatore, for providing all facilities necessary for the study.

I extend our heartiest gratitude to **Dr. (Mrs). R. Parvatham M.Sc., Dip.Ed, M.Phil., Ph.D., Dean Faculty of Science**, Avinashilingam Deemed University for Women, Coimbatore, for granting all the facilities required.

I wish to place on record my deep sense of gratitude to **Dr. G. Padmavathi M.Sc., M.Phil., Ph.D., Professor and Head, Department of Computer Science**, for providing all facilities to complete the project successfully.

I take this unique opportunity to express our sincere thanks to our project coordinator **Mrs.V.Srividhya M.Sc., M.Phil., Ph.D., Assistant Professor, Department of Computer Science**, for her kind advice and knowledgeable suggestion, which helped us to complete my project successfully.

I would like to show the immense deal of gratitude to my esteemed guide **Dr.(Mrs) I.Elizabeth Shanthi M.Sc.,M.Phil.,Ph.D., Associate Professor, Department of Computer Science**, for imparting the incredible assistance, unequivocal ideas, and well-timed support to take my project to great heights of victory.

I express my gratitude to **Dr.(Mrs).Saroja Prabakaran, M.A., Dip.Ed., Ph.D., Director,Hall of Residence,** for her support in completion of the project.

I would extend my hearty thanks to one and all who helped me directly or indirectly for successful completion of my project.

SYNOPSIS

Text classification is a supervised technique that uses labeled training data to learn the classification system that automatically classifies the remaining text using the learned system. This work tries to exert some text preprocess in different datasets and then extract a feature vector for each new document by using feature weighting and feature selection algorithms for enhancing the text classification accuracy. Feature clustering is a powerful method to reduce the dimensionality of feature vectors for text classification. In this project machine learning methods for classification is applied to design a fuzzy similarity-based self-constructing algorithm for feature clustering. The words in the feature vector of a document set are grouped into clusters, based on similarity test. Words that are similar to each other are grouped into the same cluster. Each cluster is characterized by a membership function with statistical mean and deviation. When all the words have been fed in, a desired number of clusters are formed automatically. We then have one extracted feature for each cluster. The extracted feature, corresponding to a cluster, is a weighted combination of the words contained in the cluster. The algorithm uses Naïve Bayesian (NB) and K-Nearest Neighbor (KNN) algorithms to train the classifier. The derived membership functions match closely with the real distribution of the training data. The experimental results show that both algorithms show acceptable results for text classification.

CONTENTS

1. INTRODUCTION

Text Classification or Categorization focuses on the problem of automatically assigning semantic categories to natural language text which 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, e.g., 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, e.g. `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 amount 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 handling 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. Though modern computers today are capable of learning and performing text categorization within reasonable time limits, growing amounts of data still makes Text Classification challenging today as well.

In text classification, the dimensionality of the feature vector is usually huge. For example, 20 Newsgroups and Reuters21578 top-10, which are two real-world data sets, both have more than 15,000 features. Such high dimensionality can be a severe obstacle for classification algorithms. To alleviate this difficulty, feature reduction approaches are applied before document classification tasks are performed. Two major approaches, feature selection and feature extraction have been proposed for feature reduction.

In general, feature extraction approaches are more effective than feature selection techniques, but are more computationally expensive. Therefore, developing scalable and efficient feature extraction algorithms is highly demanded for dealing with high-dimensional document data sets.

1.1 PROBLEM DEFINITION

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 classify the text using KNN and Naïve-Bayesian algorithms with the goal of a thorough performance evaluation.

- The result of the removal may lead to an incorrect root.

- 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.
- Pre- and suffix removing algorithms do not have the capability to handle this kind of stemming problem.
- Neural Networks because the decision boundary in a text categorization problem may not be linear, nonlinear classifiers such as artificial neural networks may produce better results than linear models.

1.2 OVERVIEW OF THE PROJECT

The main purpose of the program is to provide a framework to which more classifiers (e.g. 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. 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.

The various steps of the project include the following:

1. Feature Selection method, Information Gain is used on more than one classifier, e.g. a Naive Bayes classifier and a SVM.

2. Comparing both supervised (using the category information in a pre classified training set) and unsupervised selection methods.

3. Comparing several recently proposed methods that have not yet been parts of large-scale testing.

4. As there are so many variable parameters, performance results from one Researcher cannot be compared to results from another, even if the same Collection and classifier is used. Hence, comparing as many feature selection methods as possible in one system is a sub goal.

2. SYSTEM CONFIGURATION

This section describes the hardware and software specification needed for both development and implementation phases of this project.

2.1 HARDWARE SPECIFICATION

PROCESSOR	: Pentium IV
SPEED	: 2.5 GHz
RAM	: 1 GB
HARD DISK	: 80 GB
MONITOR	: 15” Color
KEYBOARD	: Logitech 104 Keys
MOUSE	: Logitech

2.2 SOFTWARE SPECIFICATION

TOOL	: Java
IDE	: Microsoft Windows XP
DATASET	: 20 Newsgroups and Reuters21578 top-10

2.3 SOFTWARE DESCRIPTION

Overview Software

Java was developed at Sun Microsystems. The Java compiler translates Java source code into instructions that are interpreted by the runtime Java Virtual Machine.

Main features of JAVA:

Object-Oriented:

Java is an Object-Oriented Programming Language. Java classes are comprised of methods and variables. Classes in Java can be defined as abstract. An abstract class is a class that collects generic state and behavioral information.

Distributed:

Java facilitates the building of distributed applications by a collection of classes for use in networked applications. By using Java's URL (Uniform Resource Locator) class, an application can easily access a remote server. Classes also are provided for establishing socket-level connections.

Robust:

Data type issues and problems are resolved at compile-time, and implicit casts of a variable from one type to another are not allowed. Memory management has been simplified in Java in two ways. First, Java does not support direct pointer manipulation or arithmetic. Second, Java uses runtime garbage collection instead of explicit freeing of memory.

Secure:

Closely related to Java's robustness is its focus on security. Because Java does not use pointers to directly reference memory locations, as is prevalent in C and C++, Java has a great deal of control over the code that exists within the Java environment.

Architecture-Neutral:

The Java compiler creates byte code instructions that are subsequently interpreted by the Java interpreter, architecture neutrality is achieved in the implementation of the Java interpreter for each new architecture.

Portable:

In addition to being architecture-neutral, Java code is also portable. It was an important design goal of Java that it be portable so that as new architectures (due to hardware, operating system, or both) are developed, the Java environment could be ported to them

High-Performance:

A Java application will not achieve the performance of a fully compiled language such as C or C++. However, for most applications, including graphics-intensive ones

such as are commonly found on the World Wide Web, the performance of Java is more than adequate.

3. SYSTEM STUDY AND ANALYSIS

System analysis is the general term that refers to an orderly and structured procedure for identifying and solving problems. It involves the study of existing system to understand how they function. This knowledge will helps to identify what the new system should include.

3.1 EXISTING SYSTEM:

Several methods are proposed to reduce the dimensionality of feature vectors for Text Classification. The first feature extraction method based on feature clustering was proposed by Ref [4] which was derived from the “distributional clustering”. Ref [2] used distributional clustering to generate an efficient representation of documents and applied a learning logic approach for training text classifiers. The divisive information-theoretic feature clustering algorithm was proposed by Ref [8], which is an information-theoretic feature clustering approach, and is more effective than other feature clustering methods. In these feature clustering methods, each new feature is generated by combining a subset of the original words. However, difficulties are associated with these methods. A word is exactly assigned to a subset, i.e., hard-clustering, based on the similarity magnitudes between the word and the existing subsets, even if the differences among these magnitudes are small. Also, the mean and the variance of a cluster are not considered when similarity with respect to the cluster is computed. Furthermore, these methods

require the number of new features be specified in advance by the user. Ref [15] showed that SVM is better than other methods for text categorization.SVM is a kernel method, which finds the maximum margin hyperplane in feature space separating the images of the training patterns into two groups. To make the method more flexible and robust, some patterns need not be correctly classified by the hyperplane, but the misclassified patterns should be penalized. SVM is a binary classifier. To do a multi-class classification, pair-wise classifications can be used (one class against all others, for all classes). Computationally expensive, thus runs slow. Ref [5] proposed an experiment with n-grams for text categorization on the Reuters dataset. They define an n-gram as an alphabetically ordered sequence of n stems of consecutive words in a sentence (after stop words were removed). The authors use both unigrams and bigrams as document features. They extract the top-scored features using various feature selection methods including Mutual Information. Their results indicate that in general bigrams can better predict categories than unigrams. However, despite the fact that bigrams represent the majority of the top-scored features, the use of bigrams does not yield significant improvement of the categorization results while using the Rocchio classifier. Specifically, in 20 of the 48 reported experiments a certain increase in the accuracy is observed, while in 28 others the accuracy decreases, sometimes quite sharply. Ref [9] investigates the problem of authorship attribution which is a special case of the text categorization problem. They apply SVM on two text representations: BOW and a bag of all the functional words and bigrams of functional words in the text. By functional words they mean all the parts of speech excluding nouns, verbs and adjectives. The later document representation is supposed to preserve the style while suppressing the topic. The results show that the

simple-minded BOW outperforms the sophisticated representation based on unigrams and bigrams of functional words.

DISADVANTAGES OF EXISTING SYSTEM

There are some issues pertinent to most of the existing feature clustering methods.

- First, the parameter k, indicating the desired number of extracted features, has to be specified in advance. This gives a burden to the user, since trial-and-error has to be done until the appropriate number of extracted features is found.
- Second, when calculating similarities, the variance of the underlying cluster is not considered. Intuitively, the distribution of the data in a cluster is an important factor in the calculation of similarity.
- Third, all words in a cluster have the same degree of contribution to the resulting extracted feature. Sometimes, it may be better if more similar words are allowed to have bigger degrees of contribution.

Our feature clustering algorithm is proposed to deal with these issues.

3.2 PROPOSED SYSTEM

Given a set of documents and their associated class labels, text classification is the problem of finding the true class label of a new document. There are several algorithms used for text classification, ranging from the simple but effective Naïve Bays algorithm to the more computationally demanding Support Vector Machines. To classify the Text, an KNN and Naïve-Bayesian classification is built. A common characteristic of text data is its extremely high dimensionality. A standard procedure to reduce feature dimensionality is feature selection and feature extraction. An alternative approach is to reduce feature dimensionality by grouping “similar” words into a much smaller number of word-clusters, and use these clusters as features. In the first stage clusters that capture the information about the set of documents are extracted as features, and in the second stage these features are used for clustering the documents in an unsupervised manner. The results clearly showed that this representation of the documents, significantly improved the accuracy of unsupervised document classification. In general, feature extraction approaches are more effective than feature selection techniques, but are more computationally expensive. Feature clustering is a powerful method to reduce the dimensionality of feature vectors for text classification. Feature clustering is the way to reduce the dimensionality of features presents in the text documents and it is highly important for text categorization problems. The performance of the text classification is degraded when the dimensionality of input text is huge. Feature clustering is a powerful alternative to feature reduction approaches. The first task is to calculate the word patterns for each feature present in the text document. The word patterns are calculated by using the frequency of a certain feature in the text document and the class label information.

The second task is to calculate the membership function by the mean and deviation of the word patterns. The third one is to generate the clusters based on the membership function. Evaluation results for these tasks show that the proposed methodology obtains reliable performance for text classification tasks.

ADVANTAGES

- A fuzzy feature clustering (FFC) algorithm which is an incremental clustering approach to reduce the dimensionality of the features in text classification.
- Determine the number of features automatically.
- Match membership functions closely with the real distribution of the training data.
- Runs faster than other methods.
- Better extracted features than other methods.

4. SYSTEM DESIGN

Design is the first process in the development phase of any engineered system. System design is a modeling process. The inputs to the design are the software requirements and the output will be the design specification applicable to all software design. System design is to deliver the requirements as specified in the feasibility report. The main objectives of the design are

- Practicality
- Efficiency
- Cost
- Flexibility

- Security

System design is a process of planning a new system or to the complement of the existing system. The design is based on the limitations of the existing system and the requirements specification gathered in the phase of system analysis. The major steps in the design phase are input design and output design.

4.1 INPUT DESIGN

The main objective of the input design is to provide user friendly interaction. The user has to make a minimum input when a whole process is automated. Input design is concerned with the input screen design. Input design is the process of converting user-oriented inputs to a computer-based format. The main interface is the heart of the program. The System is already able to load two corpora: The”selfmade”corpus and the Reuters 21578 corpus. For the development used a small self-made corpus since the running time needed to be as short as possible. Articles are collected 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. Fig 8.Category Table of Appendix B. The second corpus already included in the system is the frequently used Reuters 21578 corpus. The corpus is freely available on the internet (Lewis 1997). uses an XML parser, it was necessary to convert the 22 SGML documents to XML, using the freely available tool SX(Clark 2001). Fig 2. Data Flow Diagram of Appendix A.

4.2 OUTPUT DESIGN

Output design generally refers to the results and information that are generated by the system. As dataset are taken as input, various techniques of data mining are being applied to generate variety of output. The outputs generated by the system are checked for its consistency and the output is provided simple so that the user can handle them with ease. Since the test documents are loaded together with the training documents, the user can then chose an external file, containing an arbitrary document, which is immediately classified by the K-NN and Naïve-Bayesian algorithm, giving a report of the results in the console window. Fig 1 SystemFlow Diagram of Appendix A.

5. SYSTEM DEVELOPMENT

5.1 LIST OF MODULES

- Preprocessing
- Feature Weighting And Extraction
- Text Classification
- Performance Evaluation

5.2 MODULE DESCRIPTION

5.2.1 PREPROCESSING

The first step towards the final classification, is to load the training corpus i.e., to read all the corpus documents, and count up the term and document frequencies for every term. The term and document frequencies are saved for every document as well as for

every category and the entire corpus. This is necessary since these values will be of need in later calculations. The term frequency of a term in a document/category/corpus is the number of times the term occurs in the document/category/corpus. The document frequency of a term in a category/corpus is the number of documents which both, contain the term at least once and belong to the category/corpus, whereas the document frequency of a term in a document is a binary value which indicates if the term occurs in the document or not. Fig 5 and 6 Term and Document Frequency of Appendix B.

REMOVAL OF STOP WORDS:

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.

Fig 1 Text Preprocessing of Appendix B.

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. Fig 1.Text Preprocessing of Appendix B.

A. PORTER STEMMING

The Porter Stemming algorithm. The 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.

B. 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 untruncated 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 (Paice 1990) which features a rule execution mechanism and externally stored rules, and

2) The Porter stemmer (Porter 1980) who 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.

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.

5.2.2 FEATURE WEIGHTING AND EXTRACTION

For many machine learning algorithms it is necessary to reduce the dimensionality of the feature space, if the original dimensionality of the space is very high. In most of the cases this improves not only the performance but also the accuracy of the classification itself. The feature weighting is the process of assigning a score for each feature i.e., term or a single word using a score computing function. The features that have high scores are selected. The scores are assigned for each feature in a selected subset of relevant features. These mathematical definitions of the score-computing functions are often defined by some probabilities which are estimated by some statistic information in the documents across different categories. Feature Weighting is the process of assigning values to all the terms in the corpus according to their importance for the actual classification part. Here, importance is defined as the ability of the term to distinguish between different categories in the corpus. Usually, the more important a term is the higher is the assigned weight value. Fig 2.Feature Weighting and Extraction of Appendix B.

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).

FEATURE CLUSTER

Feature clustering algorithm is an incremental, self-constructing learning approach. Word patterns are considered one by one. The user does not need to have any idea about the number of clusters in advance. No clusters exist at the beginning, and clusters can be created if necessary. For each word pattern, similarity of this word pattern to each existing cluster is calculated to decide whether it is combined into an existing cluster or a new cluster is created. Existing feature clustering methods are referred as hard clustering, the words are exactly belongs to a particular cluster. But the proposed algorithm is referred as soft clustering since the features have the similarity to more than one cluster. Once a new cluster is created, the corresponding membership function should be initialized. The membership function is calculated using a measure called fuzzy similarity measure with mean and deviation. Evaluation results for these tasks show that this fuzzy clustering obtains reliable performance for text classification tasks.

Two cases may occur.

- First, there are no existing fuzzy clusters on which the word pattern has passed the Similarity test.
- Second, if there are existing clusters on which the word pattern has passed the Similarity test.

Note that the word patterns in a cluster have a high degree of similarity to each other. Besides, when new training patterns are considered, the existing clusters can be adjusted or new clusters can be created, without the necessity of generating the whole set of clusters from the scratch. Note that the order in which the word patterns are fed in influences the clusters obtained. We apply a heuristic to determine the order. Sort all the patterns, in decreasing order, by their largest components. Then the word patterns are fed in this order. In this way, more significant patterns will be fed in first and likely become the core of the underlying cluster.

FEATURE EXTRACTION

Word patterns have been grouped into clusters, and words in the feature vector W are also clustered accordingly. For one cluster, we have one extracted feature. Since we have k clusters, we have k extracted features. The elements of T are derived based on the obtained clusters, and feature extraction will be done. We propose three weighting approaches: hard, soft, and mixed. In the hard-weighting approach, each word is only allowed to belong to a cluster, and so it only contributes to a new extracted feature. In the soft-weighting approach, each word is allowed to contribute to all new extracted features, with the degrees depending on the values of the membership functions. The mixed-weighting approach is a combination of the hard-weighting approach and the soft-weighting approach. Fig 2.Feature Weighting and Extraction of Appendix B.

5.2.3 TEXT CLASSIFICATION

K-NEAREST NEIGHBOR CLASSIFICATION

The k-NN algorithm is a so called non-linear, lazy learner, and belongs to the class. The k-NN algorithm uses the cosine similarity as similarity function. When the document vectors are normalized, only the vector product has to be calculated, which can be done faster than the computation of the Euclidian distance, especially for sparse vectors. The cosine similarity measures the cosine of the angle between two vectors. The bigger the value, the smaller is the actual angle and the more similar are the two vectors. This makes the distance metric also independent of the length of the documents, as document vectors of different length, but with the same angle to each other, will have zero distance. The K-NN algorithm is especially suitable to compare the accuracy of the weighting functions in the dimensionality reduction step.

Since the test documents are loaded together with the training documents, the test document can now be ”scored” by clicking on the ”Score test documents” button in the ”Classifiers” interface. For every test document, the 50 most similar training documents are then saved. This makes it possible to test different values for k (up to 50) very fast in the actual evaluation step. Additionally, it is possible to save this scoring to an external file. (Menu: Classifiers->k nearest neighbors-> Save Scoring). Since the scoring of the test documents can take a very long time (e.g., for the Reuters 21578 ModApt´e split the scoring takes _ 20 hours on a Pentium M 1.5 GHz processor) the scoring is ”valuable” information and should be save and loadable. To load a scoring from an external file (Menu: Classifiers-> k-nearest neighbor -> Load Scoring), the corpus belonging to the scoring has to be loaded before.

Independently from the scoring of the test documents, it is also possible to classify documents with the button “Classify file”. The user can then chose an external file, containing an arbitrary document, which is immediately classified by the k-nearest neighbor algorithm, giving a report of the results in the console window. However, the classification of an external file does only work, if the feature space has been reduced before (that is, the training documents are holding the normalized TF·IDF values, according to the current feature space). Fig 9.KNN Classification of Appendix B.

NAÏVE-BAYESIAN CLASSIFICATION ALGORITHM

The Bayesian Classification represents a supervised learning method as well as a statistical method for classification. Assumes an underlying probabilistic model and it allows us to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. It can solve diagnostic and predictive problems. This Classification is named after Thomas Bayes (1702-1761), who proposed the Bayes Theorem. Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined. Bayesian Classification provides a useful perspective for understanding and evaluating many learning algorithms. It calculates explicit probabilities for hypothesis and it is robust to noise in input data.

The naive Bayesian classifier is the probability based classifier, based on the features independent probability value is calculated for each and every model. Naive Bayes classifiers are among the most successful known algorithms for learning to classify text documents. The naïve Bayesian classifier construct a set of class i.e., set of category label and probability for that category. The scoring is done by ranking each category probabilities for every test document rather than a training document for every test

document. The classification for the instance is the class with the highest probability value.

The Naïve-Bayes Bernoulli classifier calculates for every category C and every test document d_t the normalized likelihood, that the document belongs to the category, given the conditional class probabilities P(t_k|C), the prior class probabilities P(C), and the binary values indicating if a feature term t_k occurs in the document dt or not. For a discussion of this classifier and a comparison with the slightly advanced multinomial Naïve-Bayes classifier. The scoring of the test documents works analogous to the scoring for the case of the k-nearest neighbor classifier, with the exception, that the scoring is a ranking of category probabilities for every test document rather than a training document ranking for every test document. Scores can be saved and loaded over the menu (Classifier -> Naïve-BayesBernoulli -> Load/Save Scoring) and a document in an external file can be classified in the same way as for the k-nearest neighbor classifier.

Fig 11.Naïve-Bayesian Classification of Appendix B.

5.2.4 PERFORMANCE EVALUATION

The evaluation of a classifier is done using the precision and recall measures .To derive a robust measure of the effectiveness of the classifier It is able to calculate the breakeven point, the 11-point precision and ”average precision” . To evaluate the classification for a threshold ranging from 0 (recall = 1) up to a value where the precision value equals 1 and the recall value equals 0, incrementing the threshold with a given threshold step size. The breakeven point is the point where recall meets precision and the eleven point precision is the averaged value for the precision at the points where recall

equals the eleven values 0.0, 0.1, 0.2... 0.9, 1.0. Fig 10 and 12. K-NN and Naïve-Bayesian Performance of Appendix B.”Average precision” refines the eleven point precision, as it approximates the area “below” the precision/recall curve. Fig.13 Precision/Recall graph of Appendix B.

6. CONCLUSION

This project focuses on implementing an K-Nearest Neighbor and Naïve-Bayesian 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. Features that are similar to each other are grouped into the same cluster. Each cluster is characterized by a membership function with statistical mean and deviation. If a word is not similar to any existing cluster, a new cluster is created for this word. Similarity between a word and a cluster is defined by considering both the mean and the variance of the cluster. When all the words have been fed in, a desired number of clusters are formed automatically. We then have one extracted feature for each cluster. The extracted feature corresponding to a cluster is a weighted combination of the words contained in the cluster. By this algorithm, the derived membership functions match closely with and describe properly the real distribution of the training data. Besides, the user need not specify the number of extracted features in advance, and trial-and-error for determining the appropriate number of extracted features can then be avoided. Experiments on three real-world data sets have demonstrated that our method can run faster and obtain better extracted features than other methods.

7. SCOPE FOR FUTURE ENHANCEMENT

Text classification is a technology developed at the Xerox Research Centre Europe, which uses the textual information in a document. Machine-Learning based, it learns from a number of samples the vocabulary which is representative of each class it has to deal with (here “responsive” vs. “non-responsive). As a future work, we need the additional research for applying the more structural information of document to text categorization techniques and testing the proposed method on other types of texts such as newspapers with fixed form.

Another opening is a more thorough comparison of maximum entropy to other state-of-the-art text classification algorithms on several domains. Other future works include testing our method in other inductive transfer models, more intelligent methods of retrieving and using Wikipedia features.

BIBLIOGRAPHY

WEB REFERENCE

- http://java.sun.com
- http://www.sourcefordgde.com
- http://www.networkcomputing.com/
- http://people.csail.mit.edu/jrennie/20Newsgroups/, 2010.

- http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html. 2010.

BOOK REFERENCE

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

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

Ref [3] Caropreso M.F, Matwin .S, and Sebastiani .F, In Amita G. Chin, editor, *Text Databases and Document Management: Theory and Practice*, pages 78–102, Idea Group Publishing, Hershey, US, 2001, "A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization".

Ref [4] Combarro E.F,Montanes.E, Ranilla.J and Fernandez.J, *Proc. International Work-Conference on Artificial and Natural Neural Network IWANN2003*, volume 2687, Pages 743-749, Menorca, Spain, 2003, "A Comparison of the Performance of on SVM and ARNI Text Categorization whit New Filtering Measures on an Unbalanced Collection".

Ref [5] Diederich.J, Kindermann.J.L, Leopold.E, and Paaß.G, *Applied Intelligence*, 19(1/2):109–123, 2003, "Authorship attribution with support vector machines".

Ref [6] Hans-Peter Kriegel, Alexey Pryakhin, and Matthias Schubert "Multi-represented kNN-Classification for Large Class Sets".

Ref [7] Hisham Al-Mubaid and Syed A. Umair, *IEEE transactions*, vol 18, no. 9, sep 2006, "A New Text Categorization Technique Using Distributional Clustering and Learning Logic".

Ref [8] *IEEE Transactions on knowledge and data engineering*, vol. 23, no. 3, march 2011,"A Fuzzy Self-Constructing Feature Clustering Algorithm for Text Classification".

Ref [9] Joachim's.T, Technical Report LS-8- 23, Univ. of Dortmund, 1998, "Text Categorization with Support Vector Machine: Learning with Many Relevant features".

Ref [10] Kim.H, Howland.P, and Park.H, *J. Machine Learning Research*, vol. 6, pp. 37-53, 2005,"Dimension Reduction in Text Classification with Support Vector Machines".

Ref [11] Koster C.H and Seutter .M, In Sebastiani.F, editor, *Proceedings of ECIR'03, 25th European Conference on Information Retrieval*, pages161–176, Pisa, IT, 2003,Springer Verlag, "Taming wild phrases".

Ref [12] Noam Slonim and Naftali Tishby, "The Power of Word Clusters for Text Classification".

Ref [13] Oh-Woog Kwon and Jong-Hyeok Lee,*Information Processing and Management* 39(2003)25-44, "Text categorization based on k-nearest neighbor approach for Web site classification".

Ref [14] Suneetha Manne, Sita Kumari Kotha and Dr. S. Sameen Fatima, *IJCA* (0975-8887),volume 32- No 7, 2011, "A Query based Text Categorization using K- Nearest Neighbor Approach".

Ref [15] Zhang.D and Lee W.S, In Callan.J, Cormack.G, Clarke.C, Hawking.D, And Smeaton.A, editors, *Proceedings of SIGIR'03, 26th ACM International* Toronto,*Conference on Research and Development in Information Retrieval*, pages 26–32,CA, 2003.ACM Press, New York, US, "Question classification using support vectormachines".

APPENDIX A

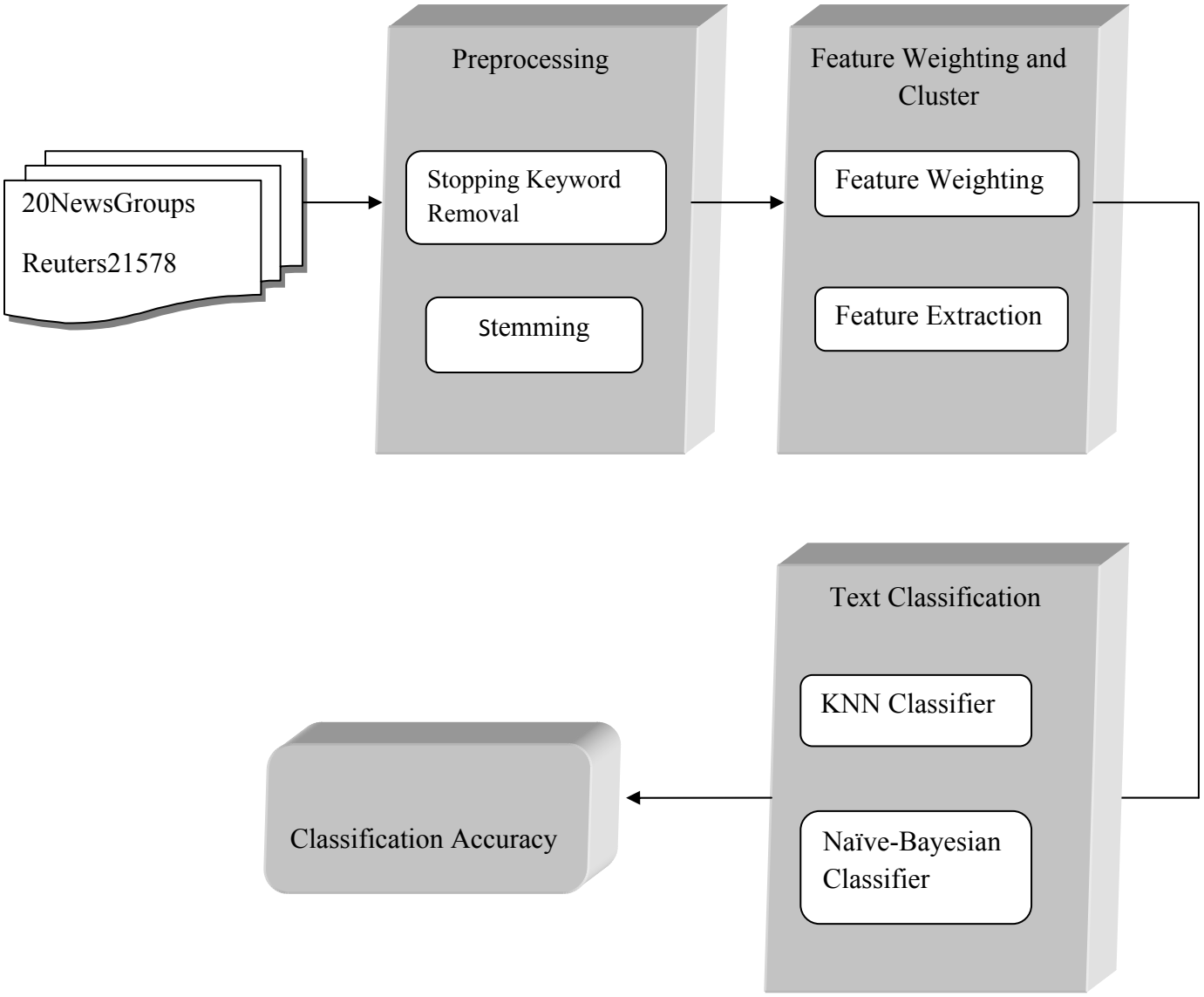


Fig 1.SystemFlow Diagram

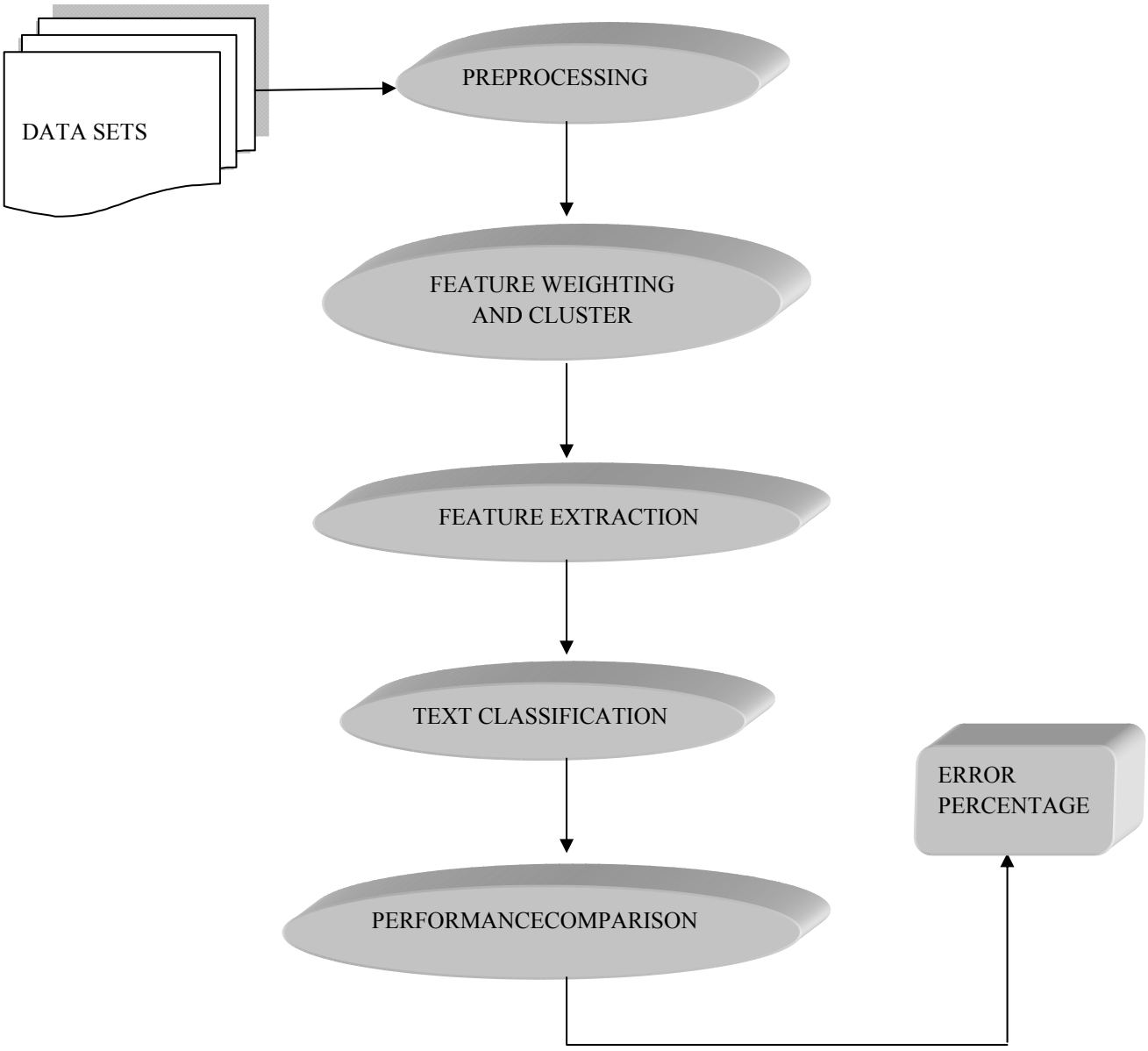


Fig 2. Data Flow Diagram

APPENDIX B

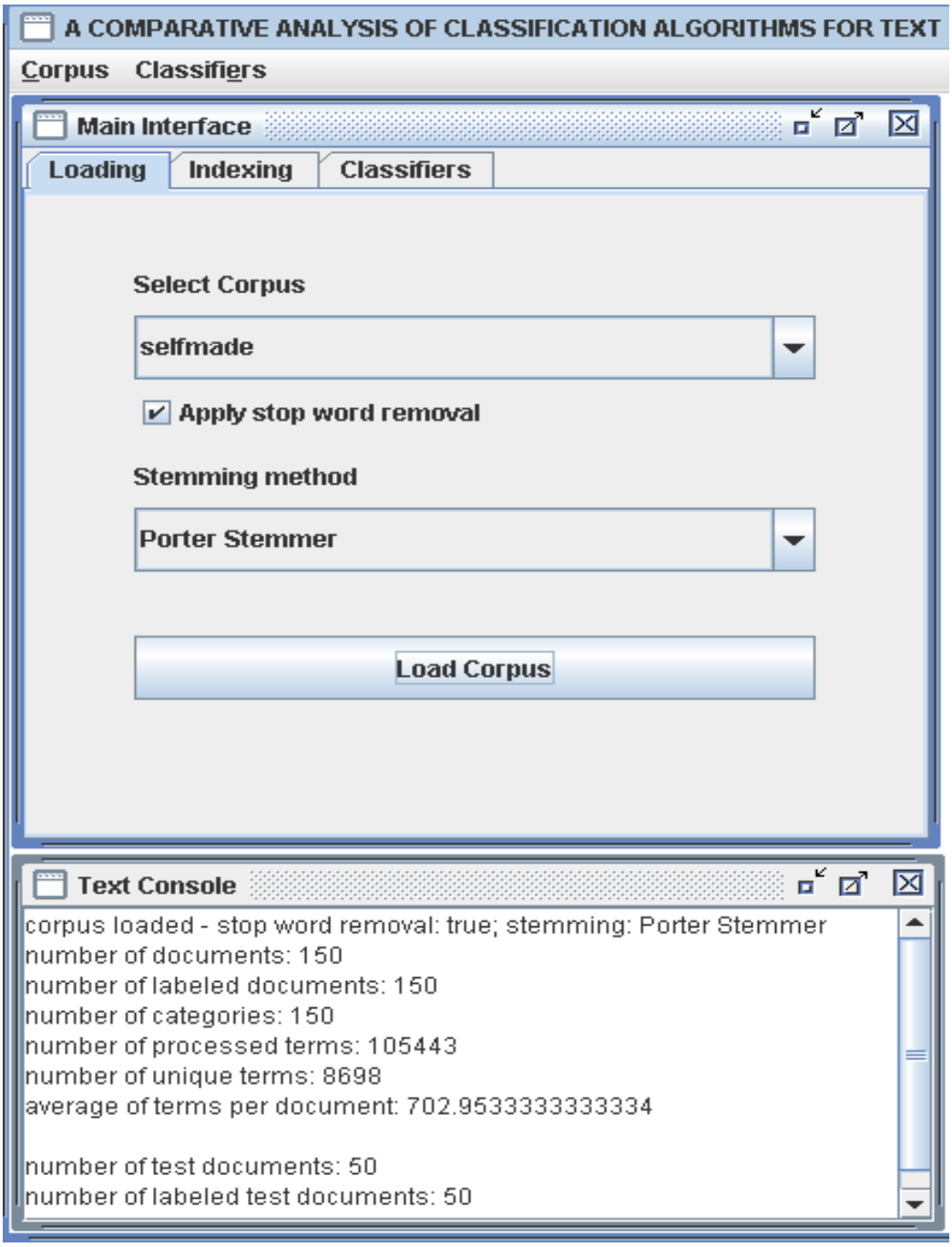


Fig 1.Text Preprocessing

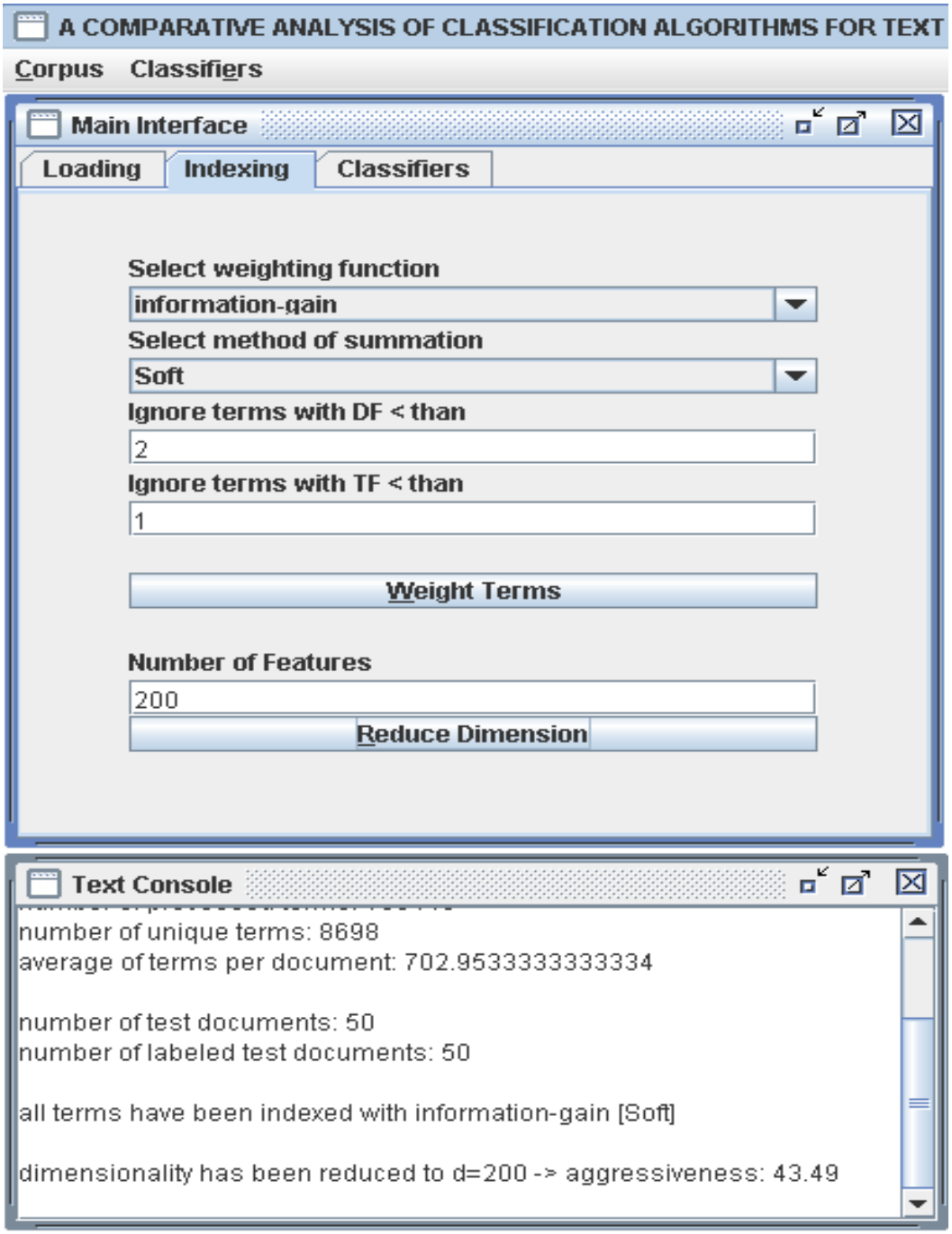


Fig 2.Feature Weighting and Extraction

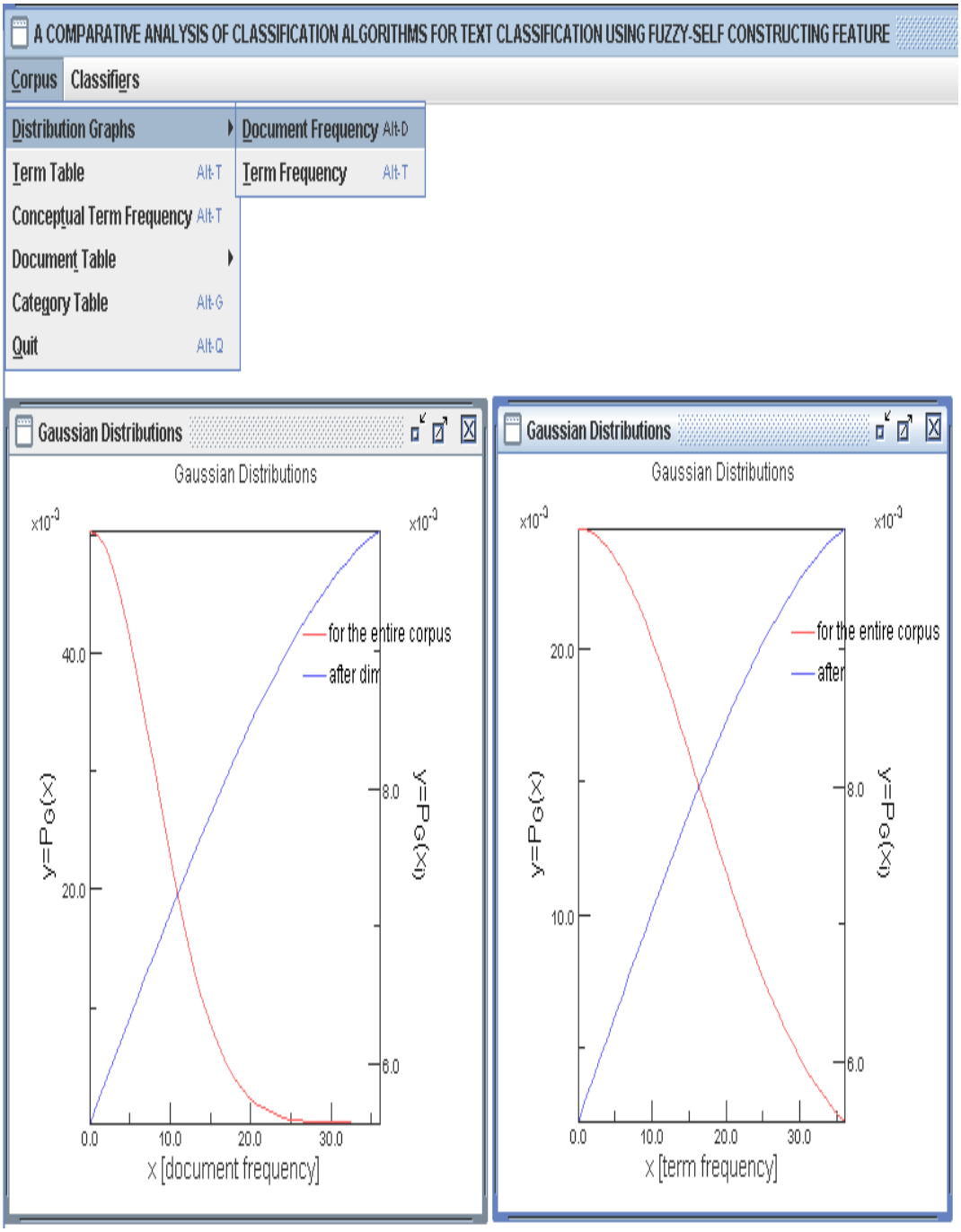
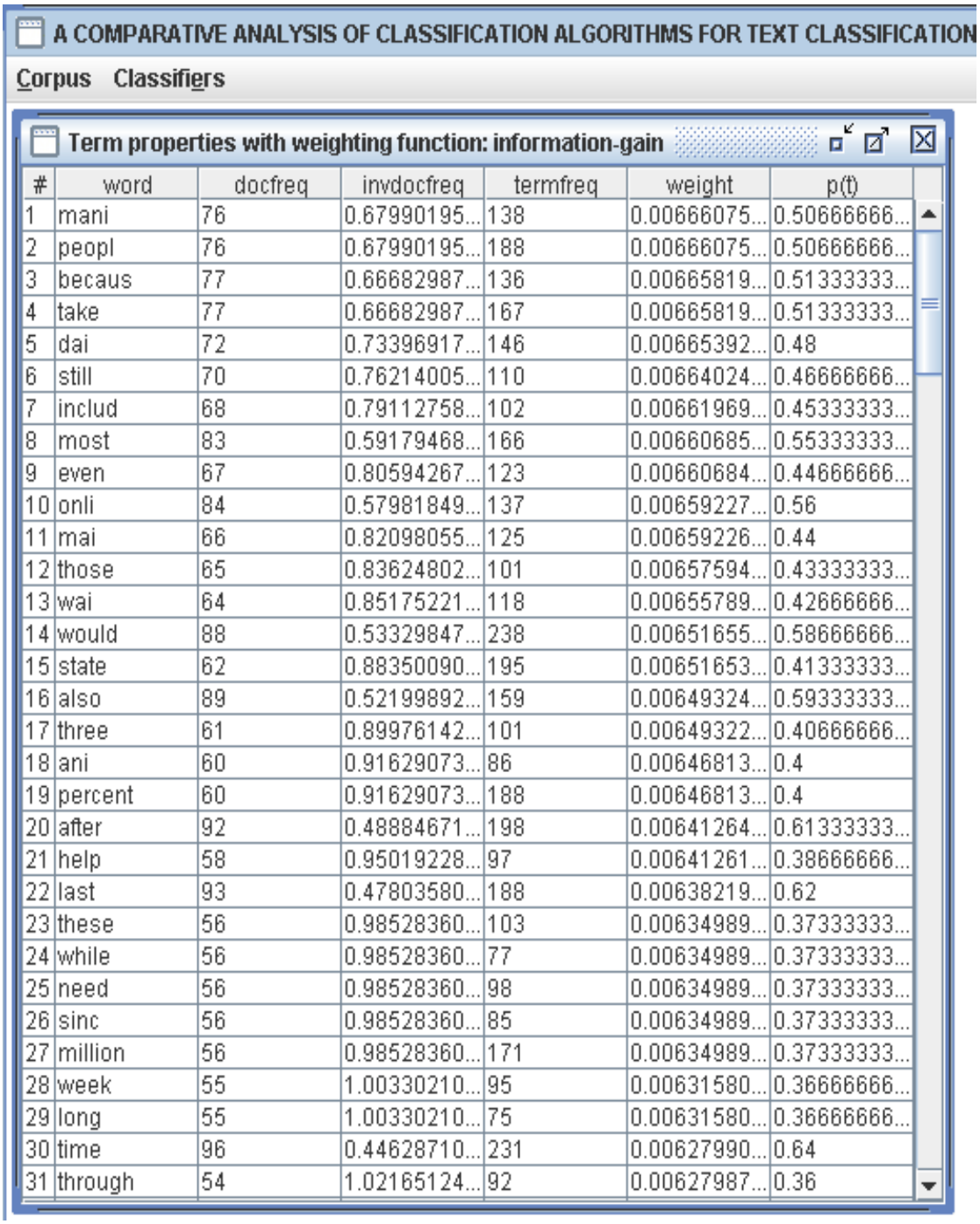
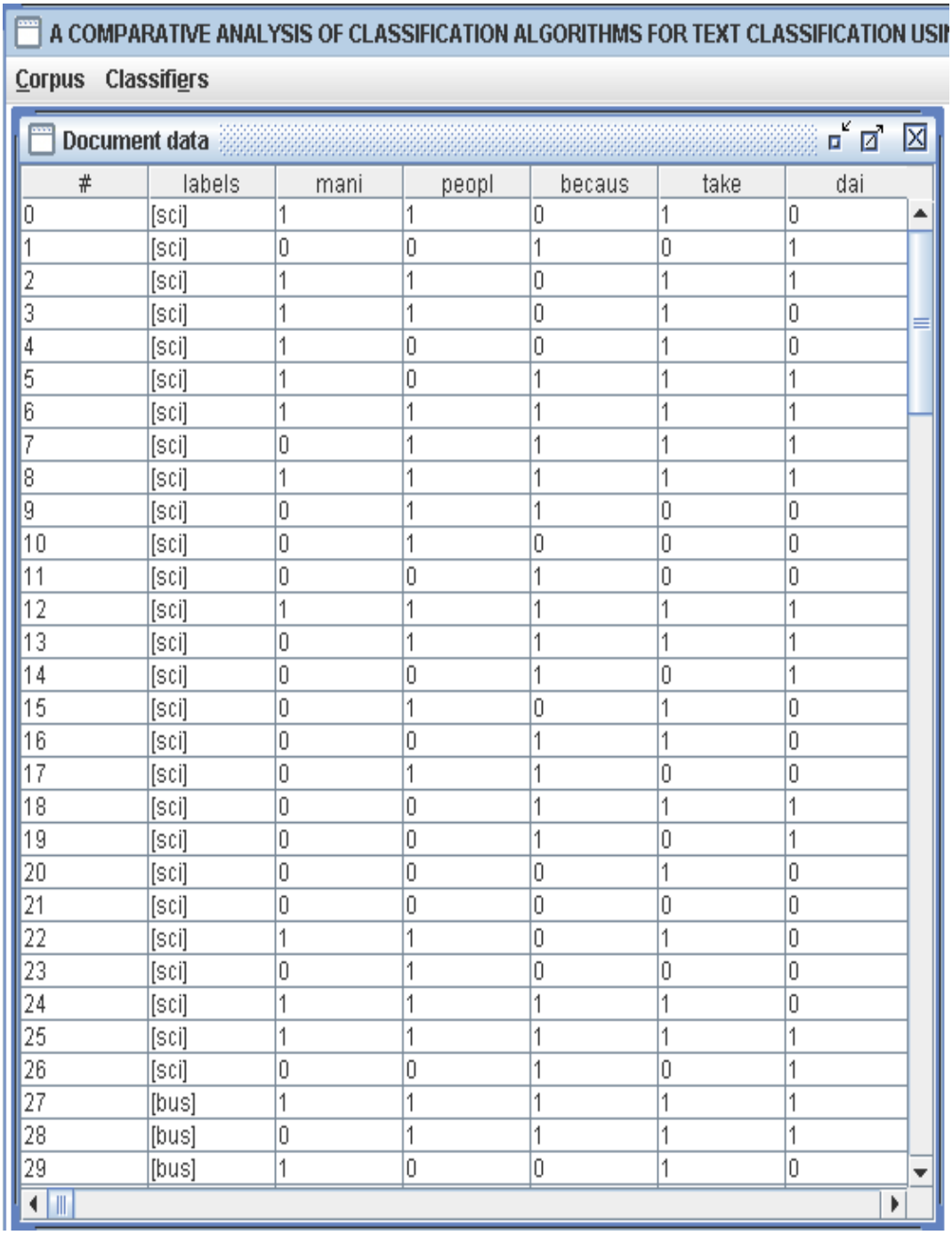


Fig 3.Gaussian Distribution Graph



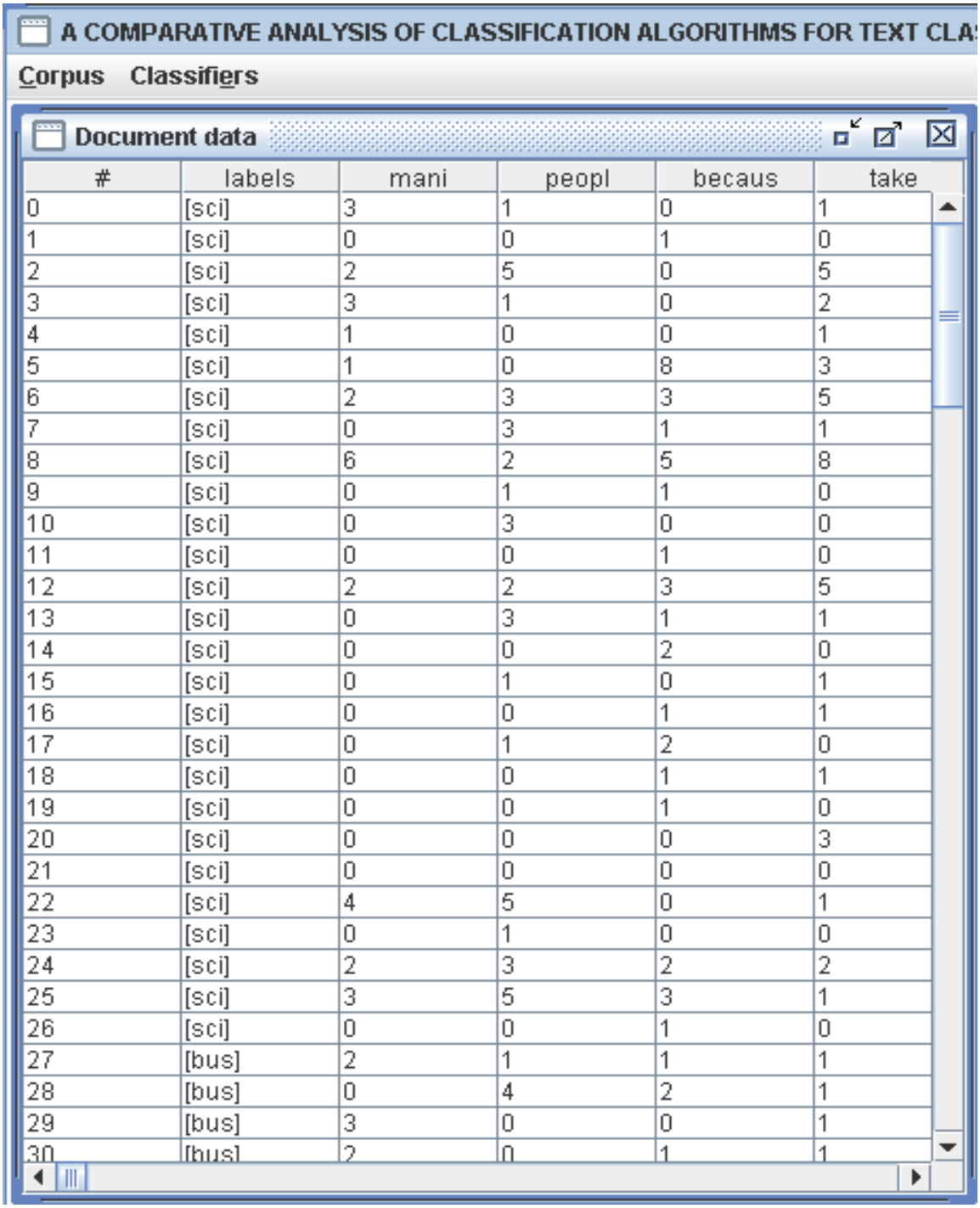
#	word	docfreq	invdocfreq	termfreq	weight	p(t)
1	mani	76	0.67990195...	138	0.00666075...	0.50666666...
2	peopl	76	0.67990195...	188	0.00666075...	0.50666666...
3	becaus	77	0.66682987...	136	0.00665819...	0.51333333...
4	take	77	0.66682987...	167	0.00665819...	0.51333333...
5	dai	72	0.73396917...	146	0.00665392...	0.48
6	still	70	0.76214005...	110	0.00664024...	0.46666666...
7	includ	68	0.79112758...	102	0.00661969...	0.45333333...
8	most	83	0.59179468...	166	0.00660685...	0.55333333...
9	even	67	0.80594267...	123	0.00660684...	0.44666666...
10	onli	84	0.57981849...	137	0.00659227...	0.56
11	mai	66	0.82098055...	125	0.00659226...	0.44
12	those	65	0.83624802...	101	0.00657594...	0.43333333...
13	wai	64	0.85175221...	118	0.00655789...	0.42666666...
14	would	88	0.53329847...	238	0.00651655...	0.58666666...
15	state	62	0.88350090...	195	0.00651653...	0.41333333...
16	also	89	0.52199892...	159	0.00649324...	0.59333333...
17	three	61	0.89976142...	101	0.00649322...	0.40666666...
18	ani	60	0.91629073...	86	0.00646813...	0.4
19	percent	60	0.91629073...	188	0.00646813...	0.4
20	after	92	0.48884671...	198	0.00641264...	0.61333333...
21	help	58	0.95019228...	97	0.00641261...	0.38666666...
22	last	93	0.47803580...	188	0.00638219...	0.62
23	these	56	0.98528360...	103	0.00634989...	0.37333333...
24	while	56	0.98528360...	77	0.00634989...	0.37333333...
25	need	56	0.98528360...	98	0.00634989...	0.37333333...
26	sinc	56	0.98528360...	85	0.00634989...	0.37333333...
27	million	56	0.98528360...	171	0.00634989...	0.37333333...
28	week	55	1.00330210...	95	0.00631580...	0.36666666...
29	long	55	1.00330210...	75	0.00631580...	0.36666666...
30	time	96	0.44628710...	231	0.00627990...	0.64
31	through	54	1.02165124...	92	0.00627987...	0.36

Fig 4.Term Table



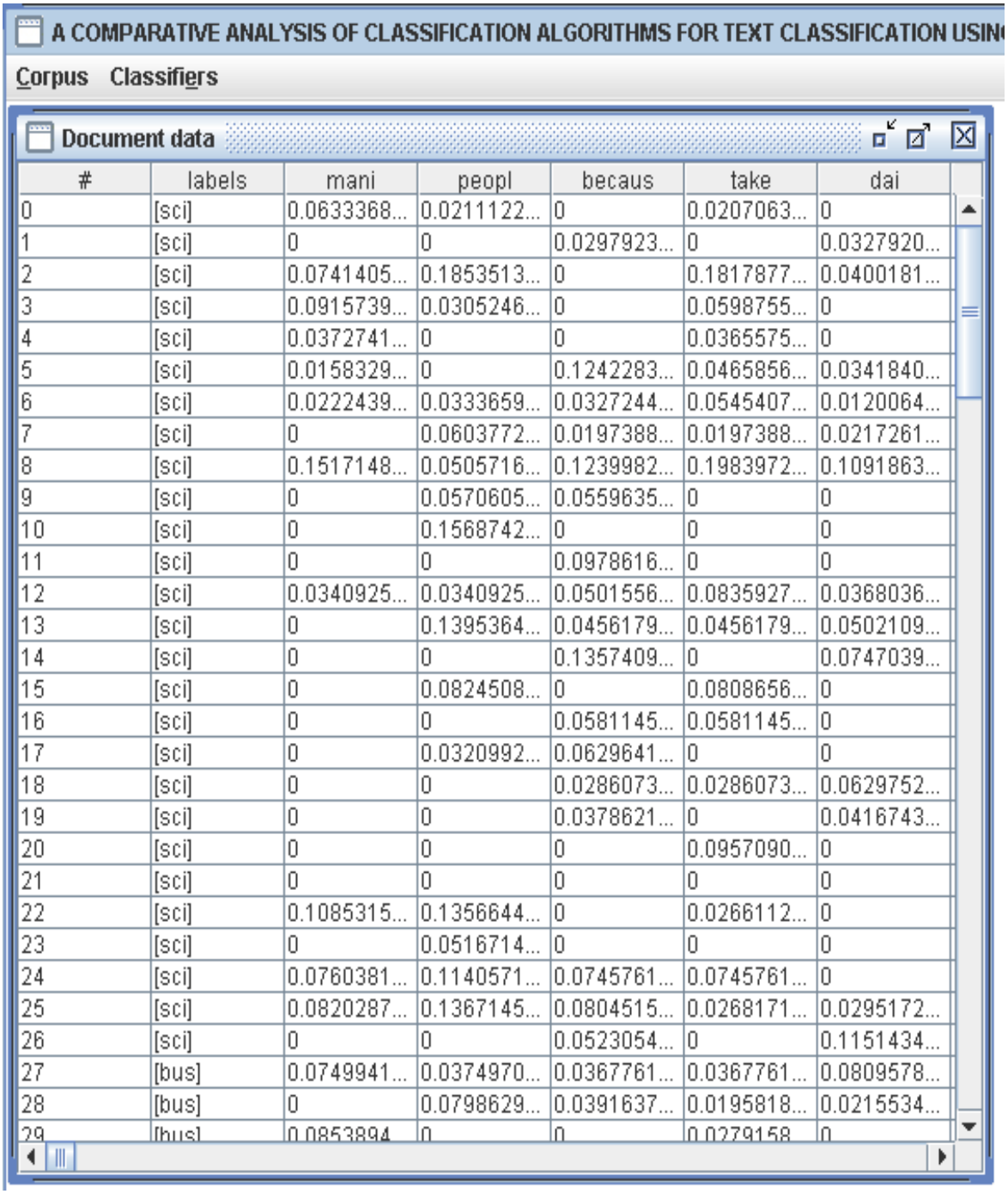
#	labels	mani	peopl	becaus	take	dai
0	[sci]	1	1	0	1	0
1	[sci]	0	0	1	0	1
2	[sci]	1	1	0	1	1
3	[sci]	1	1	0	1	0
4	[sci]	1	0	0	1	0
5	[sci]	1	0	1	1	1
6	[sci]	1	1	1	1	1
7	[sci]	0	1	1	1	1
8	[sci]	1	1	1	1	1
9	[sci]	0	1	1	0	0
10	[sci]	0	1	0	0	0
11	[sci]	0	0	1	0	0
12	[sci]	1	1	1	1	1
13	[sci]	0	1	1	1	1
14	[sci]	0	0	1	0	1
15	[sci]	0	1	0	1	0
16	[sci]	0	0	1	1	0
17	[sci]	0	1	1	0	0
18	[sci]	0	0	1	1	1
19	[sci]	0	0	1	0	1
20	[sci]	0	0	0	1	0
21	[sci]	0	0	0	0	0
22	[sci]	1	1	0	1	0
23	[sci]	0	1	0	0	0
24	[sci]	1	1	1	1	0
25	[sci]	1	1	1	1	1
26	[sci]	0	0	1	0	1
27	[bus]	1	1	1	1	1
28	[bus]	0	1	1	1	1
29	[bus]	1	0	0	1	0

Fig 5.Document Frequency



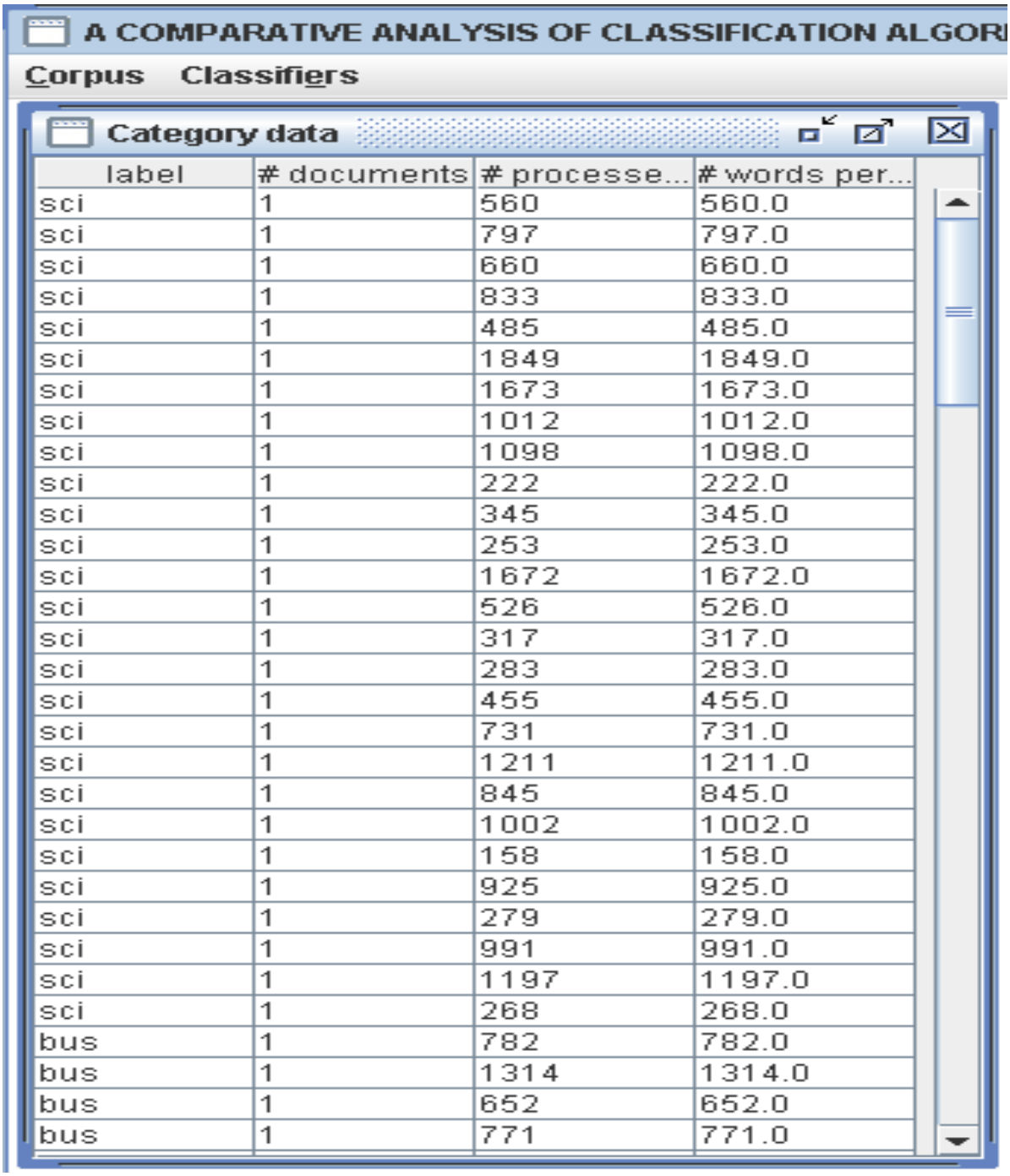
#	labels	mani	peopl	becaus	take
0	[sci]	3	1	0	1
1	[sci]	0	0	1	0
2	[sci]	2	5	0	5
3	[sci]	3	1	0	2
4	[sci]	1	0	0	1
5	[sci]	1	0	8	3
6	[sci]	2	3	3	5
7	[sci]	0	3	1	1
8	[sci]	6	2	5	8
9	[sci]	0	1	1	0
10	[sci]	0	3	0	0
11	[sci]	0	0	1	0
12	[sci]	2	2	3	5
13	[sci]	0	3	1	1
14	[sci]	0	0	2	0
15	[sci]	0	1	0	1
16	[sci]	0	0	1	1
17	[sci]	0	1	2	0
18	[sci]	0	0	1	1
19	[sci]	0	0	1	0
20	[sci]	0	0	0	3
21	[sci]	0	0	0	0
22	[sci]	4	5	0	1
23	[sci]	0	1	0	0
24	[sci]	2	3	2	2
25	[sci]	3	5	3	1
26	[sci]	0	0	1	0
27	[bus]	2	1	1	1
28	[bus]	0	4	2	1
29	[bus]	3	0	0	1
30	[bus]	2	0	1	1

Fig 6.Term Frequency



#	labels	mani	peopl	becaus	take	dai
0	[sci]	0.0633368...	0.0211122...	0	0.0207063...	0
1	[sci]	0	0	0.0297923...	0	0.0327920...
2	[sci]	0.0741405...	0.1853513...	0	0.1817877...	0.0400181...
3	[sci]	0.0915739...	0.0305246...	0	0.0598755...	0
4	[sci]	0.0372741...	0	0	0.0365575...	0
5	[sci]	0.0158329...	0	0.1242283...	0.0465856...	0.0341840...
6	[sci]	0.0222439...	0.0333659...	0.0327244...	0.0545407...	0.0120064...
7	[sci]	0	0.0603772...	0.0197388...	0.0197388...	0.0217261...
8	[sci]	0.1517148...	0.0505716...	0.1239982...	0.1983972...	0.1091863...
9	[sci]	0	0.0570605...	0.0559635...	0	0
10	[sci]	0	0.1568742...	0	0	0
11	[sci]	0	0	0.0978616...	0	0
12	[sci]	0.0340925...	0.0340925...	0.0501556...	0.0835927...	0.0368036...
13	[sci]	0	0.1395364...	0.0456179...	0.0456179...	0.0502109...
14	[sci]	0	0	0.1357409...	0	0.0747039...
15	[sci]	0	0.0824508...	0	0.0808656...	0
16	[sci]	0	0	0.0581145...	0.0581145...	0
17	[sci]	0	0.0320992...	0.0629641...	0	0
18	[sci]	0	0	0.0286073...	0.0286073...	0.0629752...
19	[sci]	0	0	0.0378621...	0	0.0416743...
20	[sci]	0	0	0	0.0957090...	0
21	[sci]	0	0	0	0	0
22	[sci]	0.1085315...	0.1356644...	0	0.0266112...	0
23	[sci]	0	0.0516714...	0	0	0
24	[sci]	0.0760381...	0.1140571...	0.0745761...	0.0745761...	0
25	[sci]	0.0820287...	0.1367145...	0.0804515...	0.0268171...	0.0295172...
26	[sci]	0	0	0.0523054...	0	0.1151434...
27	[bus]	0.0749941...	0.0374970...	0.0367761...	0.0367761...	0.0809578...
28	[bus]	0	0.0798629...	0.0391637...	0.0195818...	0.0215534...
29	[bus]	0.0853894...	0	0	0.0279158...	0

Fig 7.Term frequency Inverse Document frequency (TFIDF)



label	# documents	# processe...	# words per...
sci	1	560	560.0
sci	1	797	797.0
sci	1	660	660.0
sci	1	833	833.0
sci	1	485	485.0
sci	1	1849	1849.0
sci	1	1673	1673.0
sci	1	1012	1012.0
sci	1	1098	1098.0
sci	1	222	222.0
sci	1	345	345.0
sci	1	253	253.0
sci	1	1672	1672.0
sci	1	526	526.0
sci	1	317	317.0
sci	1	283	283.0
sci	1	455	455.0
sci	1	731	731.0
sci	1	1211	1211.0
sci	1	845	845.0
sci	1	1002	1002.0
sci	1	158	158.0
sci	1	925	925.0
sci	1	279	279.0
sci	1	991	991.0
sci	1	1197	1197.0
sci	1	268	268.0
bus	1	782	782.0
bus	1	1314	1314.0
bus	1	652	652.0
bus	1	771	771.0

Fig 8.Category Table

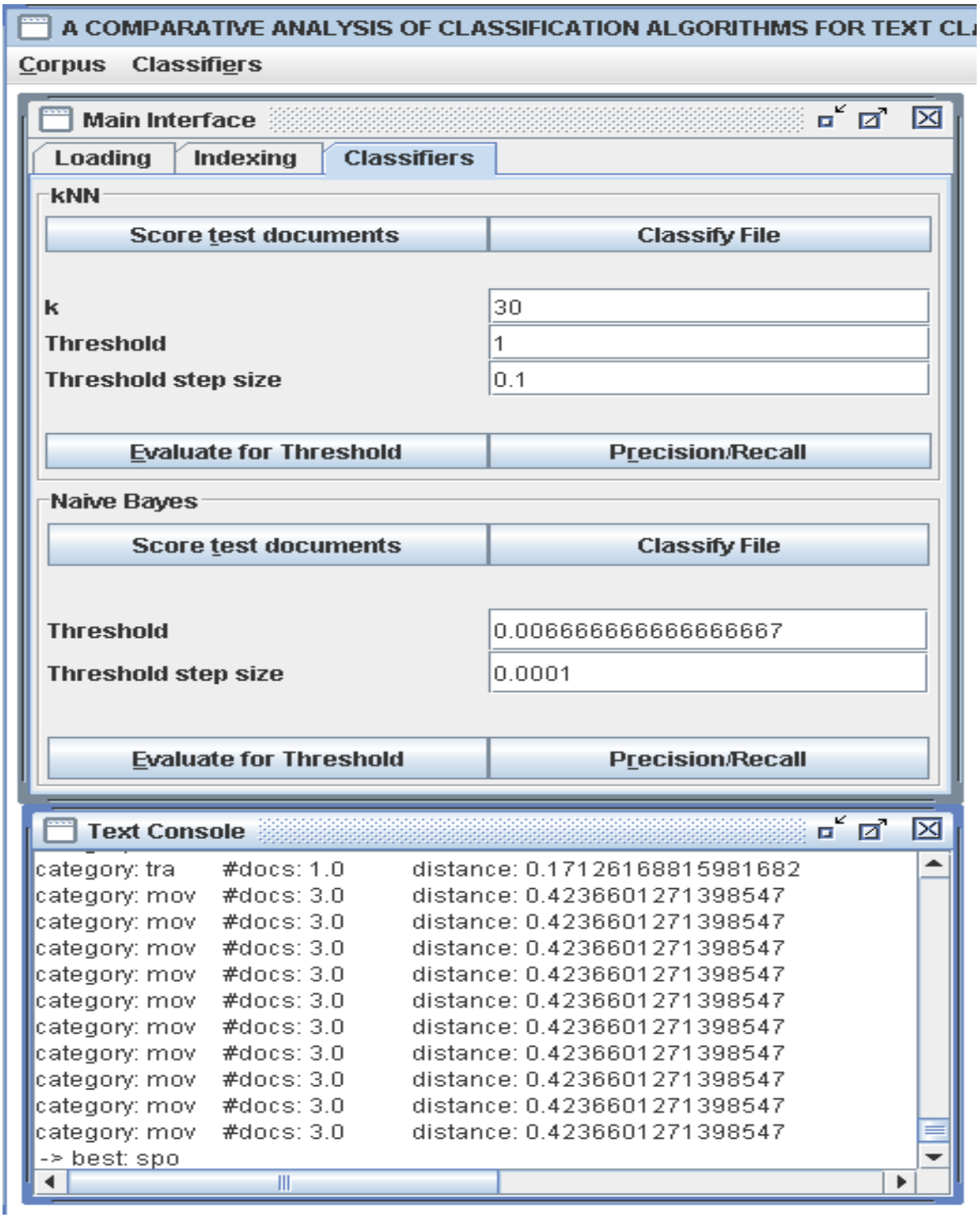


Fig 9.KNN Classification

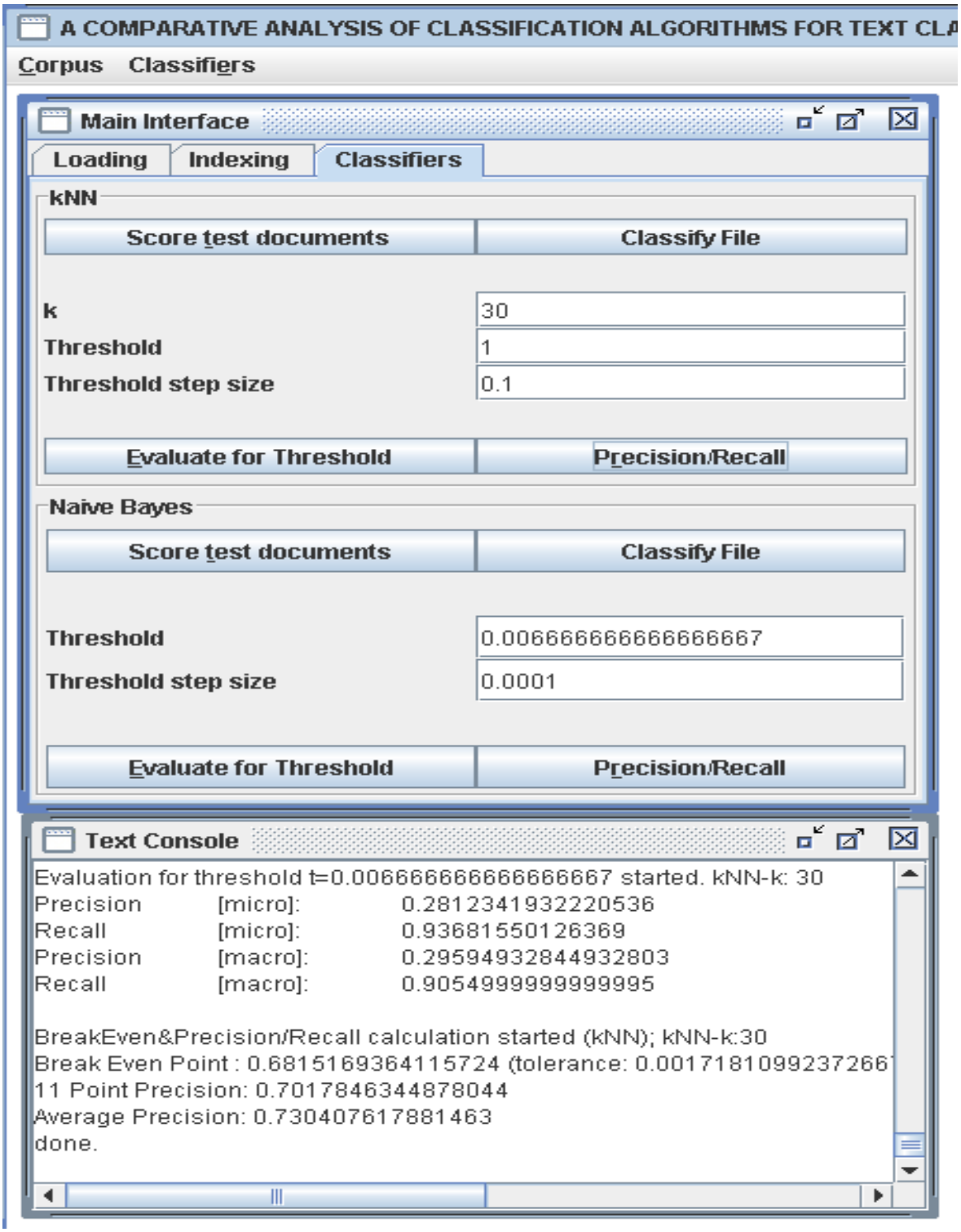


Fig 10.KNN Performance

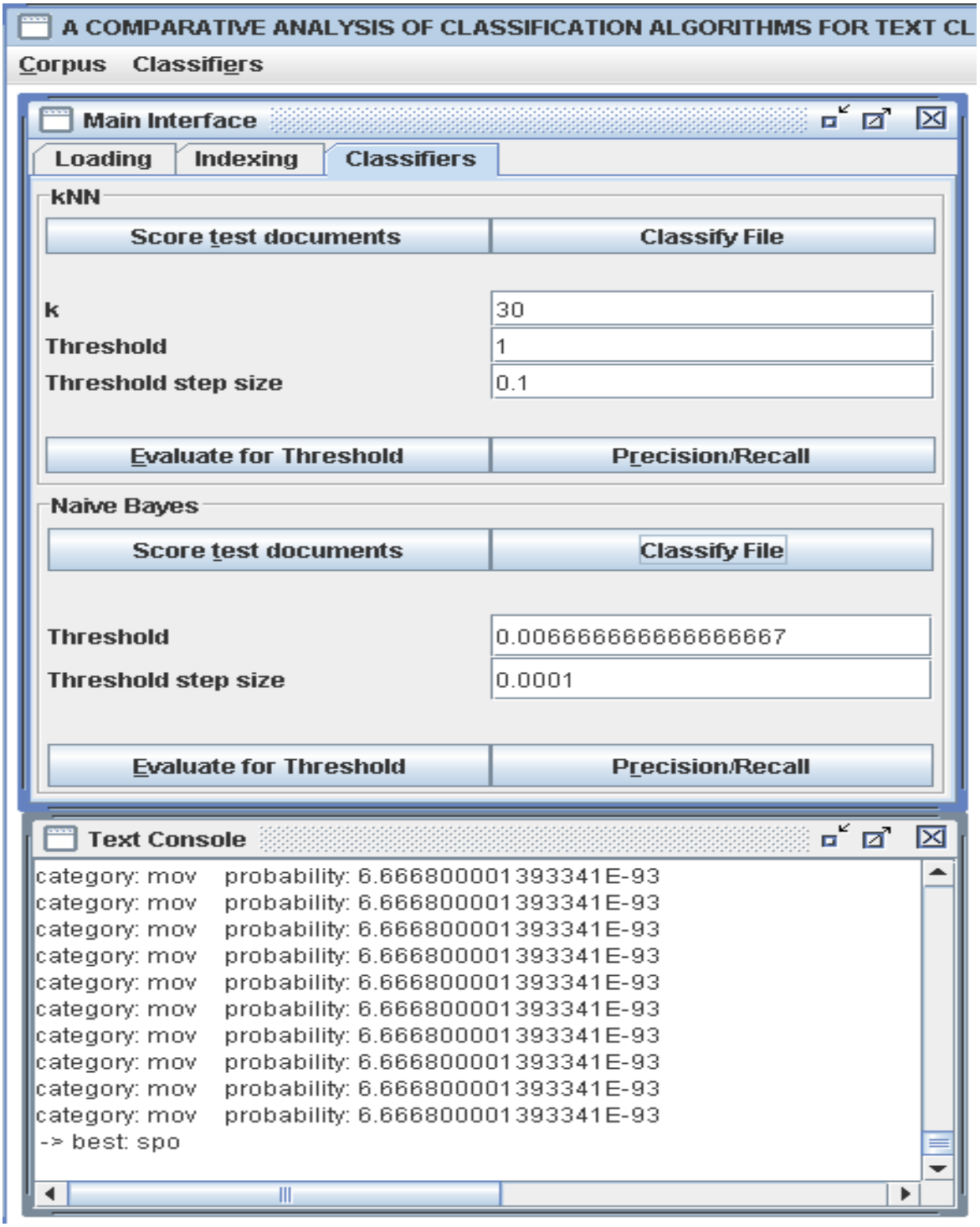


Fig 11. Naïve-Bayesian Classification

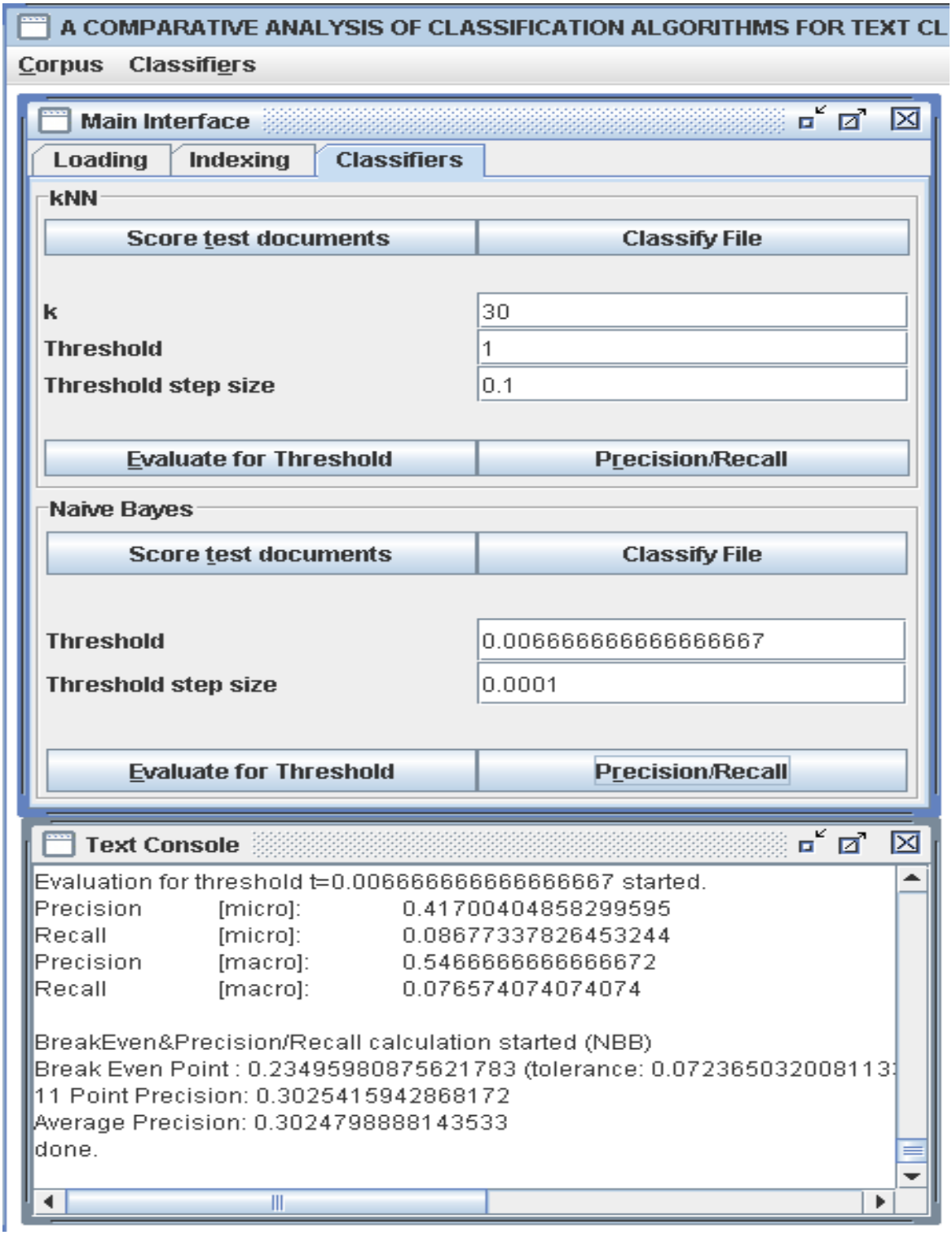


Fig 12. Naïve-Bayesian Performance

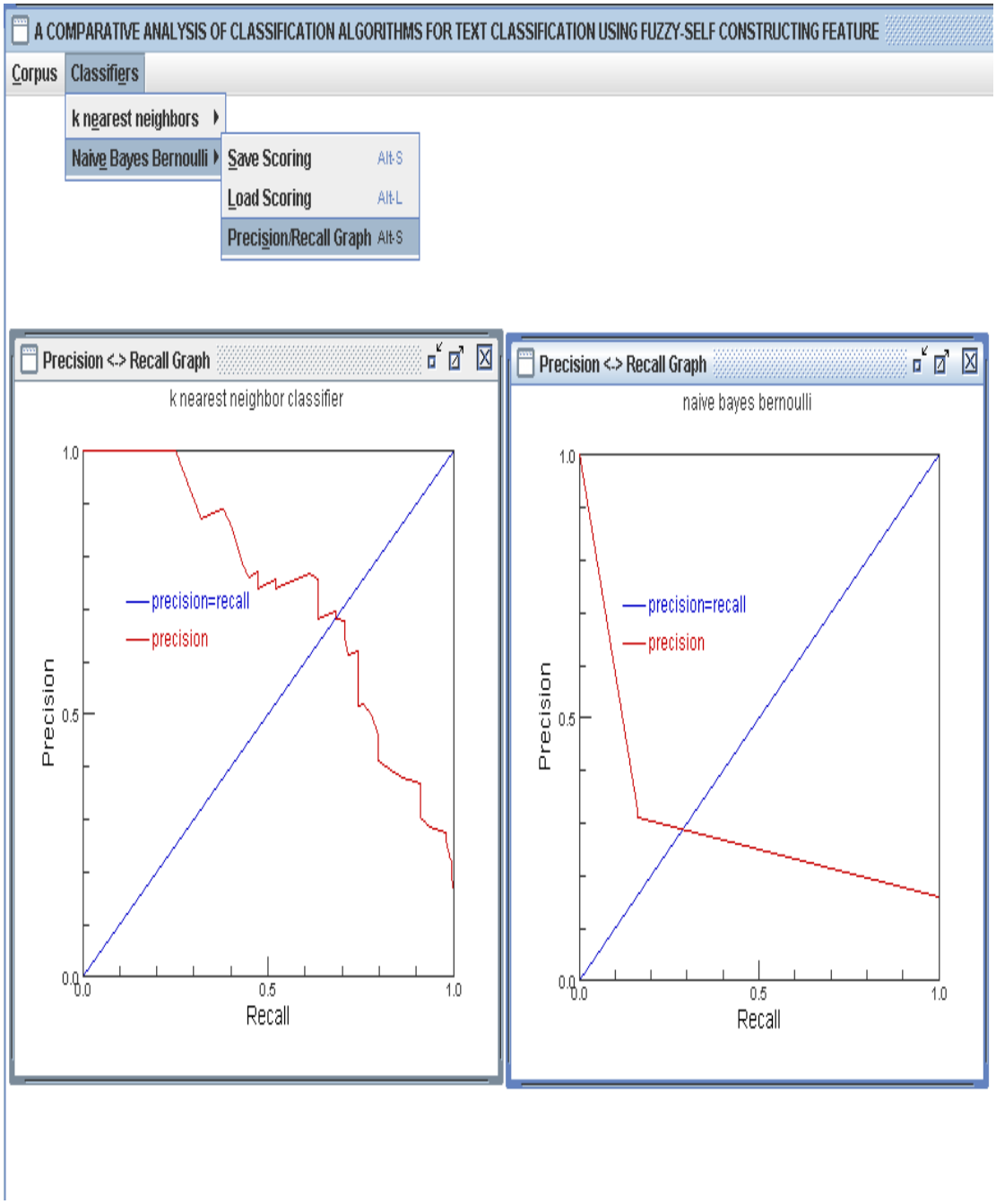


Fig 13.Precision/Recall Graph