pISSN 1976-913X
eISSN 2092-805X

J Inf Process Syst, Vol.8, No.4, December 2012 http://dx.doi.org/10.3745/JIPS.2012.8.4.693

# Machine Learning Based Keyphrase Extraction: Comparing Decision Trees, Naïve Bayes, and Artificial Neural Networks

Kamal Sarkar*, Mita Nasipuri* and Suranjan Ghose*

**Abstract**—The paper presents three machine learning based keyphrase extraction methods that respectively use Decision Trees, Naïve Bayes, and Artificial Neural Networks for keyphrase extraction. We consider keyphrases as being phrases that consist of one or more words and as representing the important concepts in a text document. The three machine learning based keyphrase extraction methods that we use for experimentation have been compared with a publicly available keyphrase extraction system called KEA. The experimental results show that the Neural Network based keyphrase extraction method outperforms two other keyphrase extraction methods that use the Decision Tree and Naïve Bayes. The results also show that the Neural Network based method performs better than KEA.

**Keywords**—Keyphrase Extraction, Decision Tree, Naïve Bayes, Artificial Neural Networks, Machine Learning, WEKA

## 1. INTRODUCTION

The keyphrases help readers rapidly understand, organize, access, and share the information in a document. Keyphrases are the phrases consisting of one or more significant words. They can be incorporated in the search results as subject metadata to facilitate information searches on the web [1]. A set of keyphrases related to a document may be viewed as a concise summary of the document. Although a summary produced by selecting a set of keyphrases lacks the fluency of the normal text, a list of keyphrases associated with a document may serve as an indicative summary or document metadata, which helps readers in searching for relevant information. Keyphrases are meant to serve various goals. For example:

1. When they are printed on the first page of a journal document, the goal is summarization. They enable the reader to quickly determine whether the given article is worth reading in-depth.
2. When they are added to the cumulative index for a journal, the goal is indexing. They enable the reader to quickly find an article that is relevant to a specific need.
3. When a search engine form contains a field labelled *keywords*, the goal is to enable the

Manuscript received July 2, 2012; accepted October 5, 2012.
**Corresponding Author: Kamal Sarkar**
* Dept. of Computer Science and Engineering, Jadavpur University, Kolkata, India ({jukamal2001, mitanasipuri}@yahoo.com, suranjanghose@yahoo.co.uk)

reader to make the search more precise. A search for documents that match a given query term in the *keyword* field will yield a smaller, higher quality list of hits than a search for the same term in the full text of the documents.

4. When the searching is done on limited display area devices such as mobiles, PDAs etc., the concise summary, in the form of keyphrases, provides a way for displaying search results in the smaller display area [2, 3].

Although the research articles published in the journals generally come with several author assigned keyphrases, many documents such as news articles, review articles, etc. may not have author assigned keyphrases at all, or the number of author-assigned keyphrases available with the documents is too limited to represent the topical content of the articles. Also, many documents do not come with author assigned keyphrases. So, an automatic keyphrase extraction process is highly desirable.

## 2. RELATED WORK

In a number of previous works the authors have suggested that document keyphrases can be useful in a variety of applications such as retrieval engines [1, 4], browsing interfaces [5], thesaurus construction [6], and document classification and clustering [7].

An algorithm to choose noun phrases from a document as keyphrase candidates has been proposed in [8]. Phrase length, phrase frequency, and the frequency of head nouns are the features used in this work. Noun phrases are extracted from a text using a base noun phrase skimmer and an off-the-shelf online dictionary.

Chien [9] developed a keyphrase extraction system for Chinese and other Asian languages.

HaCohen-Kerner et al. [10, 11] proposed a model for keyphrase extraction based on supervised machine learning and a combination of the baseline methods. They applied a *decision tree* (a machine learning algorithm) for feature combination.

Hulth et al. [12] proposed a keyphrase extraction algorithm in which the hierarchically organized thesaurus, and the frequency analysis were integrated. Inductive logic programming has been used to combine evidences from frequency analysis and the thesaurus.

A graph based model for keyphrase extraction has been presented in [13]. A document is represented as a graph in which the nodes represent terms, and the edges represent the co-occurrence of terms. Whether a term is a keyword is determined by measuring its contribution to the graph.

A neural network based approach to keyphrase extraction has been presented in [14]. It exploits traditional features such as term frequency, inverted document frequency, and position (binary) features. The neural network has been trained to classify a candidate phrase as either a keyphrase or not a keyphrase. A threshold value of 0.5 has been applied to the output predictions for the crisp classification of the candidate phrase in the categories of: keyphrase or not a keyphrase.

Turney [15] has treated the problem of keyphrase extraction as supervised learning task. Some of the features are the positional information of the phrase in the document and whether or not the phrase is a proper noun. Keyphrases are extracted from candidate phrases based on an examination of their features. Turney's program is called the Extractor.

A keyphrase extraction program called Kea, developed by Frank et al. [16, 17], uses Bayesian learning for keyphrase extraction tasks. A model is developed from the training documents with exemplar keyphrases and corresponds to a specific corpus containing the training documents. Each model consists of a Naive Bayes classifier and two supporting files containing phrase frequencies and stop words (a stop word is an insignificant word, such as "the"), which are commonly used (occur frequently) in a language. The list of such words is called a “stop list” or a “stop word list.” The developed model is used to identify the keyphrases from a document. In both Kea and Extractor, the candidate keyphrases are identified by splitting up the input text according to phrase boundaries (numbers, punctuation marks, dashes, and brackets etc.). Finally, a phrase is defined as a sequence of one, two, or three words that appear consecutively in the text. The phrases beginning or ending with a stop word are not taken into consideration. Kea and Extractor both used supervised machine learning based approaches. Two important features, such as the distance of the phrase's first appearance and TF*IDF (used in the information retrieval setting), are considered during the development of Kea. Here TF corresponds to the frequency of a phrase in a document and IDF is estimated by counting the number of documents in the training corpus that contain the phrase at least once.

Frank et al. [16, 17] has compared the performance of Kea to Turney’s work and showed that the performance of Kea is comparable to the system proposed by Turney.

An n-gram based technique for filtering keyphrases has been presented in [18]. This approach initially computes n-grams such as unigram, bigram, etc. for extracting candidate keyphrases. The n-grams that contain the verb with the form <x+ing> or the stop word “of” at the beginning or at the end are removed. After filtering the candidate keyphrases, the phrases are ranked based on the features such as term frequency and the position of the phrase in a document and also in a sentence.

Medical literature such as research articles, clinical trial reports, medical case reports, and medical news available on the web are the important sources to help clinicians who are working in patient care. The pervasion of the huge amount of medical information through WWW has created a growing need for the development of techniques for discovering, accessing, and sharing knowledge from medical literature. The straightforward method for keyphrase extraction in the medical domain is to extend the methods used in other domains to the medical domain. But, when the domain changes, the style of the document, keyphrases, and their features may change and domain specific features may be useful in keyphrase extraction tasks. A few studies have been carried out for keyphrase extraction in medical domain. Li et.al. [19], presented a study on extracting concepts from medical documents. In their study, the noun phrases have been considered as the candidate phrases and the noun phrases have been identified using a lexicon database, which integrates the WordNet lexical database [20] and the SPECIALIST Lexicon (http://umlsks.nlm.nih.gov/). They used a domain specific glossary database to determine the domain specificity of the candidate phrases and integrated the domain specific feature and the traditional term frequency feature to rank candidate phrases for keyphrase extraction. Keyphrase extraction using Naïve Bayes in the medical domain has been presented in [28]. This system has been tested on a small set of 25 documents.

## 3. A FRAMEWORK FOR MACHINE LEARNING BASED KEYPHRASE EXTRACTION

This section discusses a framework for keyphrase extraction using a machine learning algorithm. Fig. 1 shows a basic framework for keyphrase extraction using machine learning techniques. A corpus of documents and human assigned keyphrases are required for training the keyphrase extractor. The source documents should be reduced to a set of candidate phrases, which are analysed in terms of the features of interest. The candidate phrases are represented in terms of feature vectors. Then the feature vectors are labelled as "positive" (keyphrase) or "negative" (not a keyphrase). The labelling procedure involves a comparison between the source document phrases and the document's keyphrases created by the authors or a professional or by someone else.

The candidate phrases, which are also found in the list of human created keyphrases, can be analysed for the features that make them worthy of being keyphrases. The labelled feature vec-

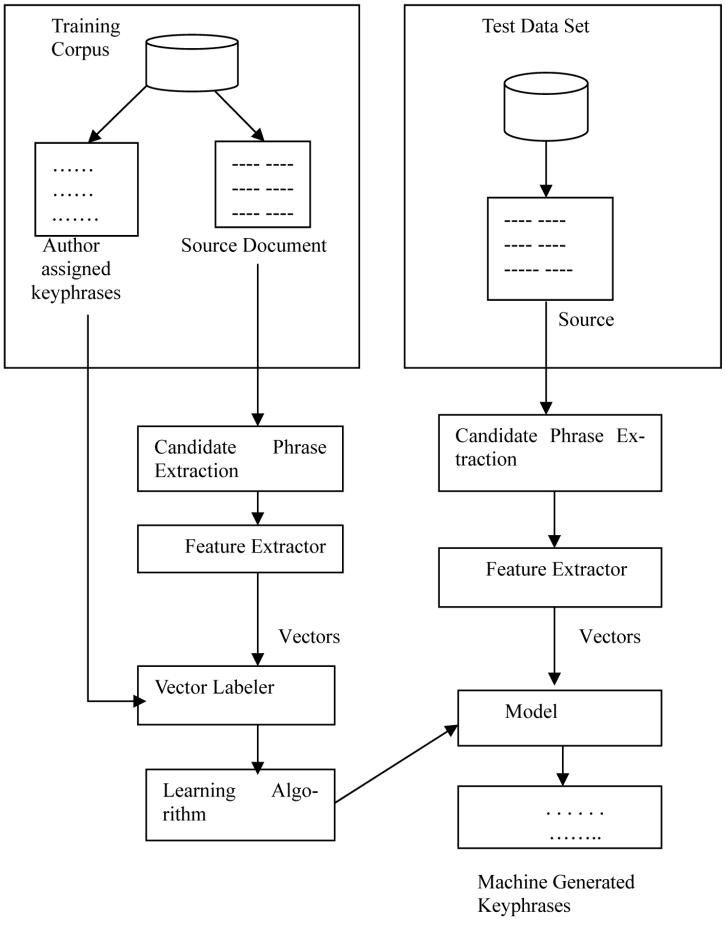



Fig. 1. Machine learning framework for keyphrase extraction

tors corresponding to the candidate phrases are then fed to a learning algorithm, which learns a model that can be used to classify each candidate phrase as a being or not being a keyphrase.

As shown in Fig. 1, a machine learning algorithm is fed with the labelled feature vectors, which are prepared after performing several steps such as candidate phrase extraction and feature extraction on the corpus of source documents.

## 4. CANDIDATE KEYPHRASE EXTRACTION AND FEATURE EXTRACTION

### 4.1 Candidate Keyphrase Extraction

The candidate keyphrase extraction method consists of two primary components: document preprocessing and noun phrase identification. The noun phrases in the document are treated as the candidate keyphrases [1].

*4.1.1 Document Preprocessing*

The preprocessing task includes formatting the document. In general, journal articles are available in PDF format. If the source document is in PDF format, it is converted to a text format (ASCII format) before it is submitted to the keyphrase extractor. Tables and figures are manually removed.

*4.1.2 Noun Phrase Identification*

To identify the noun phrases, documents should be tagged. The articles are passed to a Part-Of-Speech (POS) tagger called MontyTagger [21] to extract the lexical information about the terms. Fig. 2 shows a sample output of the MontyTagger for the following text segment:

"*European nations will either be the sites of religious conflict and violence that set Muslim minorities against secular states and Muslim communities against Christian neighbors, or they could become the birthplace of a liberalized and modernized Islam that could in turn transform the religion worldwide*."

In the Fig. 2, NN,NNS,NNP,JJ,DT,VB,IN,PRP,WDT,MD, etc. are lexical tags assigned by the tagger. In this figure, the tagger's output is shown in a bigger box and the meanings of those tags are shown in an attached small box. This is not the complete tag set. These are some examples of tags in the Penn Treebank tag-set (http://www.cis.upenn.edu/~treebank/) used by the MontyTagger.

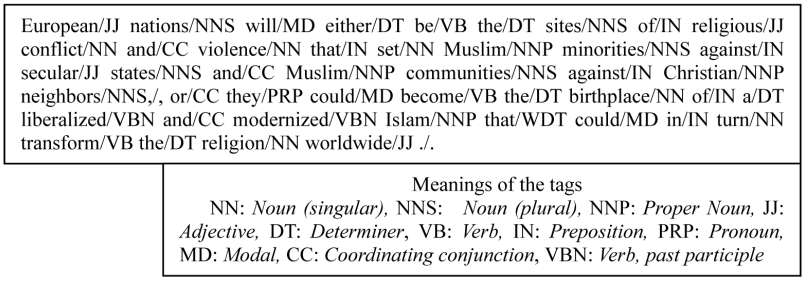
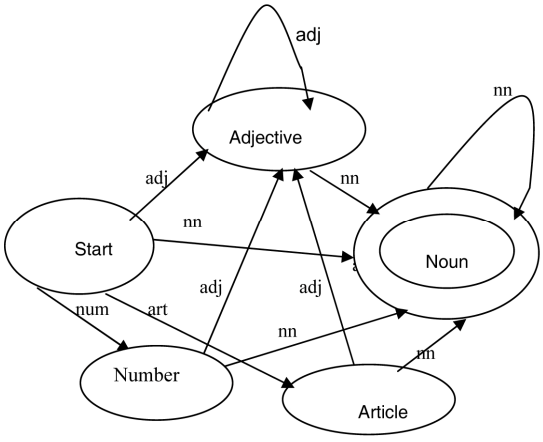



Fig. 2. A sample output of the tagger




Meanings of the labels:
*adj:* indicates any tag assigned to a word by the tagger when the word is an adjective.
*nn:* indicates any tag assigned to a word by the tagger when the word is a noun.
*art:* indicates any tag assigned to an article("a", "an", "the") or a determiner by the tagger.
*num:* indicates any tag assigned to a numeric word by the tagger

Fig. 3. The *finite state machine* for noun phrase identification

In our work, the base noun phrase finder is implemented on the basis of a finite state machine (FSM), as shown in Fig. 3. The input to the noun phrase finder is a tagged sentence. The sentences in a document are tagged by the *part-of-speech* (POS) tagger. Each state in the FSM represents a part-of-speech and an arc is labeled with the tag of a word in the input sentence. For example, the POS tagger assigns the tag, "CD," to a numeric token in a sentence. If the tag of the first word of a sentence is CD, the FSM will transit to the state "Number." The label "num" in the FSM basically refers to the tag "CD." Similarly, the label "adj" refers to one of the tags such as JJ, JJS, JJR, which are used by the tagger to indicate the various types of adjectives.

Table 1 shows a list of noun phrases identified by our noun phrase extractor. In this table, the

Table 1. Sample output of the noun phrase extractor for the sample input shown in Figure 2

| Document number | Sentence Number | Noun Phrase Number | Noun Phrases |
|---|---|---|---|
| 100 | 4 | 1 | European nations |
| 100 | 4 | 2 | sites |
| 100 | 4 | 3 | religious conflict |
| 100 | 4 | 4 | violence |
| 100 | 4 | 5 | sets Muslim minorities |
| 100 | 4 | 6 | secular states |
| 100 | 4 | 7 | Muslim communities |
| 100 | 4 | 8 | Christian neighbors |
| 100 | 4 | 9 | birthplace |
| 100 | 4 | 10 | Islam |
| 100 | 4 | 11 | turn |
| 100 | 4 | 12 | religion |

$10^{th}$ noun phrase is "Islam," but manual inspection suggests that it should be "Modernized Islam."

This discrepancy occurs since the tagger mistakenly assigns the tag, "VBN," to the word "Modernized," even though the word has been used here as an adjective. However, this is an example that shows where the error of a tagger may affect the performance of a noun phrase extractor. With a few exceptions, the proposed noun phrase extractor performs well on the documents in our corpus.

## 4.2 Features Extraction

After identifying the document phrases, a document is reduced to a collection of noun phrases. Since we focus on keyphrase extraction from scientific articles, which are generally very long in size (6 to more than 20 pages), the collection of noun phrases identified in an article may be huge in number. From a huge collection, a small number of phrases (5 to 15 phrases) may be selected as the keyphrases. So, we should design and compute a set of features based on which we can determine whether a candidate phrase is a keyphrase or not. Then a classifier can be designed based on this set of features to classify a candidate phrase as being a keyphrase or as not being a keyphrase.

Discovering good features for a classification task is very much an art. In our work, we use a set of features such as phrase frequency, phrase links to other phrases, Inverse Document Frequency (IDF), positional information, and phrase structural information. The three features of phrase frequency, phrase links to other phrases, and Inverse Document Frequency (IDF) are non-linearly combined to obtain a new variant of thematic features [22]. Similarly, phrase length and the length of the longest word in a phrase are combined into a new single feature.

The feature weighting methods and the feature value normalization methods for the different features used for our keyphrase extraction task are discussed below.

### *4.2.1 Phrase Frequency (PF), Phrase Links to Other Phrases, and Inverse Document Frequency (IDF)*

A noun phrase occurring frequently in a document is assumed to be more important in that document. The number of times a phrase (noun phrase) independently occurs in its entirety in a document is considered to be the Phrase Frequency (PF). A noun phrase may appear in a text either independently or as a part of other noun phrases. These two types of appearances of noun phrases should be distinguished. If the noun phrase P1 appears in full as a part of another noun phrase P2 (that is, P1 is contained in P2), it is considered that P1 has a link to P2. The number of times a noun phrase has links to other phrases is counted and considered to be the Phrase Link Count (PLC). Two features, Phrase Frequency (PF) and Phrase Link Count (PLC) are combined to have a single feature value using the following measure:

$$F_{freq} = \sqrt{(1/2) * PF * PF + PLC}$$

In the above formula, the Phrase Frequency (PF) is squared only to give it more importance than the Phrase Link Count (PLC). 1/2 is used to moderate the value. Here we hypothesize that an independent (full) occurrence of a phrase should get higher importance than its partial occurrence (a sub-phrase of the phrase occurs in another phrase) while deciding whether a phrase is

keyphrase worthy or not.

Inverse Document Frequency (IDF) is a useful measure to determine the commonness of a term(a phrase or a word) in a corpus. The IDF value for any term is computed using the formula: log(N/df), where N is the total number of documents in a corpus and df (document frequency) is the number of documents in the corpus that contain this term. A term with a lower df value means that the term is less frequent (i.e., rare) in the corpus and hence the IDF value (reciprocal of df) becomes higher. Thus the IDF value is a measure for determining the rarity of a term in a corpus. Traditionally, the TF (Term Frequency) value of a term is multiplied by the IDF to compute the importance of a term, where the TF indicates the frequency of a term in a document. The TF*IDF measure favors a relatively rare term that is more frequent in a document. We combine $F_{freq}$ and IDF in the following way to have a variant of the Edmundsonian Thematic feature [22]:

$$F_{thematic} = F_{freq} * IDF$$

The value of this feature is normalized by dividing it with the maximum of $F_{thematic}$ scores obtained by the candidate keyphrases belonging to a document.

4.2.2 Phrase Position

Keyphrases usually occur early in the document. So, the candidate keyphrases that occurs early in a document should be given higher score. We consider the position of the first occurrence of a phrase in a document as a feature. Unlike the previous approaches [14, 16], which assume the position of a phrase as a binary feature, in our work, the score of a phrase that occurs first in the i-th sentence in a document is computed using the following formula:

$$F_{pos} = \frac{1}{\sqrt{i}}, \text{if } i <= n$$
$$= 0, \text{otherwise}$$

where *n* is the threshold on the sentence position

*4.2.3 Phrase Length and Word Length*

These two features can be considered as the structural features of a phrase. Phrase length becomes an important feature in the keyphrase extraction task. We find that keyphrases consisting of 4 or more words are relatively rare in our corpus.

The length of the words in a phrase can be considered as a feature. According to Zipf's Law [23], shorter words occur more frequently than larger ones. For example, articles occur more frequently in a text. So, the word length can be an indication for the rarity of a word. We consider the length of the longest word in a phrase to be a feature.

If the length of a phrase is PL and the length of the longest word in the phrase is WL, these two feature values are combined to have a single feature, which is called a PL*WL feature, whose value is computed using the following formula:

$$F_{PL*WL} = \sqrt{\log(1+PL) * \log(1+WL)}$$

The value of this feature is normalized by dividing it with the maximum of the PL*WL feature values obtained by the candidate keyphrases belonging to a document.

## 5. EXPERIMENTAL DATASET

The dataset used for the experiments on the keyphrase extraction task described in this paper consists of 210 full journal articles whose sizes range from 6 pages to 30 pages. To build up the corpus, full journal articles were downloaded from the websites of journals into three domains: Economics, Legal (Law), and Medical. Articles on economics were collected from various issues of journals such as *The Journal of Economics* (Springer), *The Journal of Public Economics* (Elsevier), *Economics Letters*, and *The Journal of Policy Modeling*. All of these articles are available in PDF format.

Articles on the law and legal cases were downloaded from various issues of law journals such as *Computer Law and Security Review* (Elsevier), *The International Review of Law and Economics* (Elsevier), *The European Journal of Law and Economics* (Springer), *The Computer Law and Security Report* (Elsevier), and *The AGORA International Journal of Juridical Sciences* (Open access).

Medical articles were downloaded from various issues of medical journals such as *The Indian Journal of Medicine*; *The Indian Journal of Pediatrics*; *The Journal of Psychology and Counseling*; *The African Journal of Traditional, Complementary and Alternative Medicines*; *The Indian Journal of Surgery*; *The Journal of General Internal Medicine; The American Journal of Medicine*; *The International Journal of Cardiology*; and *The Journal of Anxiety Disorders*. A number of articles under each category used in our experiments are shown in Table 2.

For system evaluation, the set of 210 journal articles was divided into multiple folds where each fold consisted of one training set of 140 documents and a test set of 70 documents. The training set was totally disjointed from the test set. The set of author assigned keyphrases available with the articles were manually removed before candidate terms were extracted. For all experiments discussed in this paper, the same fold configurations of trainings and test sets were used. Some useful statistics about our corpus are given below.

Our corpus contained 210 journal articles. The total number of noun phrases extracted from these documents by our noun phrase extractor was 191,572. The average number of author-assigned keyphrases for these documents was 4.87. Some author-provided keyphrases may not occur in the document they are assigned to, because the authors may construct some keyphrases with their own words. We observed in our corpus that the average number of author provided keyphrases that appeared somewhere in the documents was 4.30. This justifies that some author assigned keyphrases for a document may not occur in the document they are assigned to.

Table 2. Source documents used in the experiments

| Type of Source Document | Number of Documents |
|---|---|
| Economics | 82 |
| Law | 53 |
| Medical | 75 |

## 6. EVALUATION MEASURES

There are two usual practices for evaluating the effectiveness of a keyphrase extraction system. One method was to use human judgment. We asked human experts to give scores to the keyphrases generated by the system. Another method, which is less costly, is to measure how well the system-generated keyphrases match the author-assigned keyphrases. We followed the second approach to evaluate the proposed keyphrase extraction system by computing its precision and recall using author-provided keyphrases for the documents. For our experiment, precision and recall have been defined as follows:

*precision* = the proportion of the extracted keyphrases that match the keyphrases assigned by a document's author(s). So, the precision is defined as follows:

$$\text{Precision} = \frac{N}{K}$$

where N = the number of keyphrases matched and K = the number of system generated keyphrases.

*Recall* = the proportion of the keyphrases assigned by a document's author(s) that are extracted by the keyphrase extraction system. So, the recall is defined as follows:

$$\text{Recall} = \frac{N}{M}$$

where N = the number of keyphrases matched and M = the number of author assigned keyhrases.

Previous studies have used those measures and have found that it is an appropriate method to measure the performance of a keyphrase extraction system [7, 8, 11, 19].

## 7. EXPERIMENTS

### 7.1 WEKA

For experimentation, we chose three machine learning algorithms from the machine learning workbench called WEKA.

WEKA (Waikato Environment for Knowledge Analysis) [24] is a popular suite for machine learning tools. It was developed at the University of Waikato. It is software that is freely available under the GNU General Public License.

The WEKA workbench [25] includes a collection of visualization tools and machine learning algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to this functionality.

All of WEKA's techniques assume that the data is available as a single flat file, where each data point is described by a fixed number of attributes (normally, numeric or nominal attributes). There is also a module called the *Experimenter*, which allows for the systematic comparison of the predictive performance of WEKA's machine learning algorithms on a collection of datasets.

The *Explorer* interface has several panels that give access to the various components of the workbench. The *Preprocess* panel includes the options for importing data from a database or a

file and preprocessing this data using different *filtering* algorithms. These filters can be used to transform the data (e.g., turning numeric attributes into discrete ones). The *Classify* panel enables the user to apply classification algorithms to the resulting dataset and to estimate the accuracy of the resulting predictive model.

### 7.2 Training a Learning Algorithm for Keyphrase Identification

We train a learning algorithm chosen from WEKA for classifying the noun phrases into two categories: keyphrase (positive) and not a keyphrase (negative). For training purposes, the training data should be prepared by converting the noun phrases in the source documents to the feature vectors and labeling the vectors with positive or negative labels. Then the labeled training data is used to train a learning algorithm for keyphrase identification.

Before converting the source documents into a collection of the noun phrases, author assigned keyphrases (if any) were removed and stored in different files with a document identification number. For each noun phrase in a given document we extracted the feature values from the document using the measures discussed in Subsection 4.2. We performed the following steps to label the feature vectors in the training set:

- If a noun phrase under consideration is found in a list of author assigned keyphrases corresponding to a document, the phrase is labeled as a "Positive" example; otherwise, the phrase is labeled as a "Negative" example.
- The feature vector for each noun phrase looks like {<$a_1$, $a_2$, $a_3$,.., $a_n$>} and {<$a_1$, $a_2$, $a_3$,…, $a_n$>, <label>} where the first form(unlabeled form) of a vector is used at the time of testing and the second form (labeled form) of a vector becomes a training instance (example) for a learning algorithm. $a_1$, $a_2$, … ,$a_n$, indicates the feature values for a noun phrase and <label> indicates the category label for the feature vector. For example, a positive instance and a negative instance for our keyphrase extraction task are respectively {< 1.000, 0.8149, 0.7155 >, Y} and {<0.3535, 0.1398, 0.6733 >, N}. A training set consisting of a set of instances of the above form is built up by running a computer program on the training documents selected from the corpus.
- After the training dataset has been prepared, the learning algorithm is trained on the training set to classify the candidate phrases as one of two categories—"Positive" or "Negative." The positive category indicates that the candidate phrase is a keyphrase and the negative category indicates that it is not.
- For training a learning tool under WEKA, the training data is stored in a file with the prescribed format. A classifier is then trained on a training dataset that is supplied to the WEKA tool as the file [25]. Then, the trained classifier is applied to the feature vectors corresponding to the candidate phrases identified from the test documents.

To build up a model using a learning algorithm under WEKA (www.cs.waikato.ac.nz/ml/weka), we have used WEKA's Simple CLI utility, which provides a simple command-line interface that allows for a direct execution of WEKA commands. As already mentioned, we chose the following three machine learning tools from WEKA: *Naïve Bayes*, *decision tree*, and *multilayer perceptron*. Each of these learning algorithms is trained to classify the candidate phrases into the two categories of positive (keyphrase) and negative (not keyphrase). After

training, each learning algorithm produces a separate model and each model is then tested on the test dataset.

### 7.3 Testing a Learning Algorithm for Keyphrase Identification

The model produced by a learning algorithm through training is tested on the test documents where each test document's candidate keyphrases are also presented to the model in the form of feature vectors and the candidate keyphrases are classified by the model into positive and negative categories. During testing, we used the -p option at the command line interface along with the WEKA commands. With this option, we can generate a probability estimate for the class of each vector along with the classification output. This is required when the number of the candidate keyphrases classified as positive by the classifier is less than the desired number of the keyphrases. The output produced by a classifier for each test document is saved in a separate file.

### 7.4 Keyphrase Extraction

Since a learning algorithm is trained to classify the candidate keyphrases into the two categories of positive (Y) and negative (N), the candidate keyphrases from the test documents are also classified by the learner into positive and negative categories.

Sometimes this cannot give the desired number of keyphrases specified by the users. When the number of the candidate keyphrases classified as positive by the classifier is less than the desired number of the keyphrases, we use the class probability estimates for ranking the classified candidate keyphrases using the algorithm, as shown in Fig. 4.

After ranking the candidate keyphrases using the algorithm in Fig. 4, the top K phrases are selected as the keyphrases for a test document, where the value for K is given by a user as input.

Input:

A file containing candidate phrases identified from a test document, their classifications (positive or negative) and the corresponding probability estimates of belonging to the classes (positive or negative).

Steps:

i. Select the noun phrases, which have been classified as positive by the classifier and reorder these selected noun phrases in decreasing order of the probability estimates of being in Class 1 (positive). Save the ordered phrases in an output file and delete them from the input file.

ii. For the rest of the noun phrases, which are classified by the classifier as "Negative," we order the phrases in the increasing order of the probability estimates of being in Class 0 (negative). In effect, the phrase for which the probability estimate of being in Class 0 is minimum comes at the top. Append the ordered phrases to the output file.

iii. Save the output file.

Fig. 4. Candidate keyphrase ranking based on a classifier's decisions

## 8. RESULTS AND DISCUSSION

In the previous section, we have discussed how a machine learning algorithm chosen from WEKA can be trained and tested for keyphrase extraction. The following are the configurations and the other details of the three machine learning algorithms that we used for our proposed

keyphrase extraction task.

## 8.1 Learning Algorithms

The details of the dataset are already discussed in Section 5. For all the learning algorithms used in our experiments on keyphrase extraction, the experimental dataset remained the same. The performance of the learning algorithms on keyphrase extraction task was measured by precision and recall, which have also been defined in Section 6.

*8.1.1 Learner-1: Naïve Bayes*

The training procedure for a learning algorithm has already been discussed in Subsection 7.2. The same is also applied for training the Naïve Bayesian learning algorithm for keyphrase extraction. The Naïve Bayesian learning algorithm is chosen from the WEKA machine learning toolkit.

Before training the Naïve Bayes classifier, the feature values, which are continuous, should be discretized. This is necessary for improving the performance of the Naive Bayes classifier. Discretization quantizes a continuous attribute into ranges so that the resulting new attribute can be treated as a nominal one, where each value represents a range of values of the original numeric attribute. Table 3 shows the ranges of feature values after a discretization scheme was applied to the continuous features, which are used for our keyphrase extraction task. Discretization can be done separately before training, using the WEKA *preprocess* panel, or the Naïve Bayes classifier can be run with the -D option during the training and the testing phases. The option -D is for discretizing continuous feature values while training and testing the Naïve Bayes classifier.

Fayyad and Irani's [26] discretization scheme, which is based on the Minimum Description Length (MDL) principle is used for this discretization task.

Table 4 shows author assigned keyphrases for journal article number 12 in the test corpus. Table 5 shows the top 5 keyphrases extracted by the proposed Naïve Bayesian keyphrase extractor. Table 4 and Table 5 show that out of 5 keyphrases extracted by the Bayesian approach, 2

Table 3. Discretization table

| Features | Discretization ranges | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| $F_{pos}$ | [-inf, 0.3434] | [0.3434, 0.6698] | [0.6698, 0.9082] | [0.9082, inf] |
| $F_{thematic}$ | [-inf, 0.1003] | [0.1003, 0.2776] | [0.2776, 0.6976] | [0.6976, inf] |
| $F_{PL*WL}$ | [-inf, 0.6913] | [0.6913, inf] | | |

Table 4. Author assigned keyphrases for journal article number 12 in the test corpus

| **Doc. No.** | **Author Key** |
|---|---|
| 12 | adult immunization |
| 12 | barriers |
| 12 | consumer |
| 12 | provider survey |

Table 5. Top 5 keyphrases extracted by the proposed Naïve Bayesian keyphrase extractor

| Doc. No. | Extracted Keyphrases |
|---|---|
| 12 | immunization |
| 12 | Adults |
| 12 | Barriers |
| 12 | adult immunization |
| 12 | Vaccine |

keyphrases matched with the author assigned keyphrases. The overall performance of the proposed Naïve Bayesian keyphrase extractor is shown in Table 8.

*8.1.2 Learner-2: Multilayer Perceptron (MLP)*

The *multilayer perceptron neural network* is included in WEKA under the panel named *classifier/ functions*. For our experiment, the classifier MLP of the WEKA is trained with the training dataset using the following values of the parameters:

- Number of layers: 3 (one input layer, one hidden layer and one output layer)
- Number of hidden nodes (neurons): 5
- Learning rate: 0.3
- Momentum: 0.2
- Training iteration: 5,000

Table 6 shows the top 5 keyphrases extracted by the proposed MLP based keyphrase extractor for journal article number 12 in the test corpus. The author assigned keyphrases for that article are shown in Table 4.

Table 4 and Table 6 show that out of 5 keyphrases extracted by the MLP based approach, 3 keyphrases match with the author assigned keyphrases. The overall performance of the proposed MLP based keyphrase extractor is shown in Table 9.

Table 6. Top 5 keyphrases extracted by the proposed MLP based keyphrase extractor

| Doc No. | Extracted Keyphrases |
|---|---|
| 12 | Immunization |
| 12 | adult immunization |
| 12 | healthcare providers |
| 12 | Consumers |
| 12 | Barriers |

*8.1.3 Learner-3: Decision Tree*

J48 is included in WEKA under the panel named *classifiers/trees*. J48 is basically the C4.5 version [27] of the *decision tree*. C4.5 can itself handle the continuous attributes efficiently, so no separate discretization algorithm is applied for this purpose. For our experiment, J48 is trained with the default values of the configuration parameters. The default values for the parameters, confidence factor, and MinNumObj are 0.25 and 2 respectively. The confidence

Table 7. Top 5 keyphrases extracted by the proposed decision tree based keyphrase extractor

| Doc No. | NP |
|---|---|
| 12 | immunization |
| 12 | adult immunization |
| 12 | sample |
| 12 | shortage |
| 12 | immunization practices |

Table 8. Overall performance of the proposed Naïve Bayesian keyphrase extraction system

| Number of Keyphrases | Proposed Naïve Bayesian Keyphrase Extraction System | |
|---|---|---|
| | Average Precision | Average Recall |
| 5 | 0.33 | 0.35 |
| 10 | 0.21 | 0.46 |
| 15 | 0.16 | 0.51 |

Table 9. Overall performance of the proposed MLP based keyphrase extraction system

| Number of Keyphrases | Proposed MLP Based Extraction System | |
|---|---|---|
| | Average Precision | Average Recall |
| 5 | 0.34 | 0.36 |
| 10 | 0.23 | 0.48 |
| 15 | 0.17 | 0.53 |

Table 10. Overall performance of the proposed decision tree based keyphrase extraction system

| Number of Keyphrases | Proposed Decision Tree Keyphrase Extraction System | |
|---|---|---|
| | Average Precision | Average Recall |
| 5 | 0.25 | 0.27 |
| 10 | 0.18 | 0.40 |
| 15 | 0.14 | 0.44 |

factor is used for pruning and its smaller value incurs more pruning. The value for the parameter, MinNumObj, specifies the minimum number of instances per leaf.

Table 7 shows the top 5 keyphrases extracted by the proposed decision tree based keyphrase extractor. A comparison of Table 4 and Table 7 shows that, out of the 5 keyphrases extracted by the decision tree based approach, 1 matches with the author assigned keyphrases. The overall performance of the proposed Decision Tree based keyphrase extractor is shown in Table 10.

## 8.2 System Performance

To measure the overall performance of the proposed machine learning based keyphrase extraction systems, the experimental dataset (210 journal articles**)** was divided into 3 folds for 3-fold cross validation. A separate model was built for each fold to collect 3 test results, which were averaged to obtain the final results for a system. For comparisons among the various keyphrase extraction systems presented in this paper, the number of keyphrases to be extracted (value for K) was set to 5, 10, and 15, respectively.

The overall performance of each of the learning based methods viz. the Naïve Bayesian keyphrase extractor, the MLP based keyphrase extractor, and the decision tree based keyphrase extractor is shown in Table 8, Table 9, and Table 10 respectively. Table 11 and Table 12 show the comparisons between the performances of the three learning algorithms in the keyphrase extraction task.

Table 11 and Table 12 show that the MLP based keyphrase extraction system performs the best and that the decision tree based keyphrase extraction system performs the worst. We can also see from the tables that the performance of the Naïve Bayes keyphrase extraction with a proper feature discretization algorithm is better than the decision tree based system and slightly worse than the MLP based system.

To interpret the results reported in this paper, we should analyze the upper bounds of the precision and recall of a keyphrase extraction system on our dataset. The upper bounds of precision and recall of a keyphrase extraction system on our dataset can be computed in the following two ways:

1) Some author provided keyphrases might not occur in the document that they were assigned to. According to our corpus, about 88% of author provided keyphrases appear somewhere in the source documents. After extracting candidate keyphrases using our candidate phrase extraction algorithm, we find that only 71% of author provided keyphrases appear somewhere in the list of candidate keyphrases extracted from all of the source documents. So, keeping our candidate keyphrase extraction algorithm fixed, if a system is designed with the best possible features or a system is allowed to extract all of the candidate keyphrases in each document as the keyphrases, the highest possible average recall for a system could only be (4.87 * 0.71) /4.87 = 0.71. In our experiments, the average number of author-provided keyphrases for all of the documents is only 4.87, so the precision would not be high even when the number of extracted keyphrases is large. For example, when the number of keyphrases to be extracted for each document is set to 10, the highest possible average precision is around (4.87 * 0.71) / 10 = 0.346. If the number of keyphrases to be

Table 11. Performance comparisons of the proposed MLP based keyphrase extraction system and the proposed Naïve Bayesian keyphrase extraction system

| Number of keyphrases | Average Precision | | Average Recall | |
|---|---|---|---|---|
| | MLP | Naïve Bayes | MLP | Naïve Bayes |
| 5 | 0.34 | 0.33 | 0.36 | 0.35 |
| 10 | 0.23 | 0.21 | 0.48 | 0.46 |
| 15 | 0.17 | 0.16 | 0.53 | 0.51 |

Table 12. Performance comparisons of the proposed MLP based keyphrase extraction system and the proposed decision tree based keyphrase extraction system

| Number of Keyphrases | Average Precision | | Average Recall | |
|---|---|---|---|---|
| | MLP | Decision Tree | MLP | Decision Tree |
| 5 | 0.34 | 0.25 | 0.36 | 0.27 |
| 10 | 0.23 | 0.18 | 0.48 | 0.40 |
| 15 | 0.17 | 0.14 | 0.53 | 0.44 |

extracted is set to 15, the highest possible average precision will again decrease.

2) In the second case, we assume that the candidate keyphrase extraction procedure is perfect. That is, the list of candidate keyphrases contains all author provided keyphrases appearing in the source documents. In that case, 88% of the author provided keyphrases appear somewhere in the list of candidate keyphrases because, on average, 88% of the author provided keyphrases may appear somewhere in the source documents of our corpus. Thus, if a system is allowed to extract all of the candidate terms in each document as the keyphrases, the highest possible average recall for a system can be 0.88 and when the number of keyphrases to be extracted for each document is set to 10, the highest possible average precision is around (4.87 * 0.88) / 10 = 0.429.

## 9. COMPARISON WITH EXISTING SYSTEMS

The proposed keyphrase extraction algorithms are compared with an existing system called Kea [16, 17], which is now a publicly available keyphrase extraction system. Kea uses Bayesian learning for keyphrase extraction task. It produces a model based on the training documents with exemplar keyphrases. Kea uses the supervised machine learning based approach. For keyphrase extraction, Kea considers only two features—the distance of the phrase's first appearance into the document and TF*IDF.

Version 5.0 of Kea[1] was downloaded and installed on a personal computer under the Linux platform. It was then trained and tested on 3 folds generated from the experimental dataset and the final results were obtained by averaging the 3 test results for 3 folds. Each fold consisted of 140 training documents and 70 test documents. For each fold, during the training phase of Kea, a model was built based on Naïve Bayes. This pre-built model was used to extract keyphrases from the test documents. The option of -K was set while training Kea. This option is used to incorporate keyphrase frequency statistics. From the Kea Manual it is found that using this option improves accuracy of Kea. The number of keyphrases to be extracted was set to 5, 10, and 15. Table 13, Table 14, and Table 15 respectively show the performance comparisons between Kea and each of the proposed keyphrase extractors.

The results in the tables above show that the proposed Naïve Bayesian keyphrase extraction system and MLP based system perform significantly better than Kea and the performance of the Decision tree based keyphrase extraction system is comparable to Kea. Kea is considered to be the baseline keyphrase extraction system.

Table 13. Performance comparisons of the proposed MLP based keyphrase extraction system and Kea

| Number of Keyphrases | Average Precision | | Average Recall | |
|---|---|---|---|---|
| | MLP | Kea | MLP | Kea |
| 5 | 0.34 | 0.27 | 0.36 | 0.29 |
| 10 | 0.23 | 0.19 | 0.48 | 0.40 |
| 15 | 0.17 | 0.15 | 0.53 | 0.48 |

1 http://www.nzdl.org/Kea/

Table 14. Performance comparisons of the proposed Naïve Bayesian keyphrase extraction system and Kea

| Number of Keyphrases | Average Precision | | Average Recall | |
|---|---|---|---|---|
| | Naïve Bayes | Kea | Naïve Bayes | Kea |
| 5 | 0.33 | 0.27 | 0.35 | 0.29 |
| 10 | 0.21 | 0.19 | 0.46 | 0.40 |
| 15 | 0.16 | 0.15 | 0.51 | 0.48 |

Table 15. Performance comparisons of the proposed decision tree based keyphrase extraction system and Kea

| Number of Keyphrases | Average Precision | | Average Recall | |
|---|---|---|---|---|
| | Decision Tree | Kea | Decision Tree | Kea |
| 5 | 0.25 | 0.27 | 0.27 | 0.29 |
| 10 | 0.18 | 0.19 | 0.40 | 0.40 |
| 15 | 0.14 | 0.15 | 0.44 | 0.48 |

## 10. CONCLUSION

In this paper, the performances of some machine learning algorithms on keyphrase extraction task were presented. In this context, the three learning algorithms of the *decision tree*, *Naïve Bayes*, and the *Multilayer Perceptron* (MLP) neural network were compared. The experimental results suggest that MLP based keyphrase extraction system outperforms the other two systems that use decision tree and Naïve Bayes for keyphrase extraction. One interesting finding was that for keyphrase extraction, the performance of the Naïve Bayes classifier with a suitable discretization algorithm is comparable to that of the MLP based system. The proposed systems are compared to the publicly available keyphrase extraction system called Kea. Kea is considered to be a baseline system. The Naïve Bayesian keyphrase extraction system and the MLP based keyphrase extraction system outperformed Kea. The performance of the decision tree was comparable to Kea. From these observations, Kea and the decision tree based system can be considered as two different baseline keyphrase extraction systems.

In the future, the proposed systems can be improved by: (1) improving the candidate phrase extraction module of the system and (2) incorporating new features such as document structural features, lexical features, etc.

A manual inspection in the list of author provided keyphrases associated with the documents in the training corpus reveals that there are some documents where the associated author provided keyphrases are not present in the document in the exact form. That is, these types of keyphrases have the following two forms:

1) Component words of the phrase are present in the document, but the phrase itself is not present in the document.
2) Few component words of the phrase are present in the document, but the rest are either not present in the document in exact form (synonym may be present) or are not present at all.

Generating these types of phrases with proper care may improve keyphrase extraction performance, but how it can be achieved with a proper control on false positives has yet to be investigated.

## REFERENCES


[1] Y. B. Wu, Q. Li, "Document keyphrases as subject metadata: incorporating document key concepts in search results". *Journal of Information Retrieval*, Volume 11, Number 3, 2008, pp.229-249.

[2] O. Buyukkokten, H. Garcia-Molina and A. Paepcke, "*Seeking the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices", In Proceedings of the World Wide Web Conference (WWW10)*, Hong Kong, May, 2001.

[3] O. Buyukkokten, O. Kaljuvee, H. Garcia-Molina, A. Paepcke and T. Winograd, "Efficient Web Browsing on Handheld Devices Using Page and Form Summarization", *ACM Transactions on Information Systems (TOIS)*, 20(1), 2002, pp.82-115

[4] S. Jones, M. Staveley, *"Phrasier: A system for interactive document retrieval using keyphrases", In: proceedings of SIGIR'99,* Berkeley, CA, 1999.

[5] C. Gutwin, G.W. Paynter, I.H. Witten, C.G. Nevill-Manning and E. Frank, "Improving browsing in digital libraries with keyphrase indexes." *Journal of Decision Support Systems*, Vol.27, no 1-2, 1999, pp.81-104.

[6] B. Kosovac, D. J. Vanier, T. M. Froese, "Use of keyphrase extraction software for creation of an AEC/FM thesaurus", *Journal of Information Technology in Construction*, 5, 2000, pp.25-36.

[7] S. Jonse, M. Mahoui, "*Hierarchical document clustering using automatically extracted keyphrase*" , *In proceedings of the third international Asian conference on digital libraries*, Seoul, Korea, 2000, p. 113-20.

[8] K. Barker, N. Cornacchia, "*Using Noun Phrase Heads to Extract Document Keyphrases*", *In: H. Hamilton, Q. Yang (eds.): Canadian AI 2000. Lecture Notes in Artificial Intelligence*, Vol.1822, Springer-Verlag, Berlin Heidelberg, 2000, pp.40-52.

[9] L. F. Chien, "PAT-tree-based Adaptive Keyphrase Extraction for Intelligent Chinese Information Retrieval", *Information Processing and Management*", 35, 1999, pp.501-521.

[10] Y. HaCohen-Kerner, "*Automatic Extraction of Keywords from Abstracts*", *In: V. Palade, R. J. Howlett, L. C. Jain (eds.): KES 2003. Lecture Notes in Artificial Intelligence*, Vol.2773, Springer-Verlag, Berlin Heidelberg, 2003, pp.843-849.

[11] Y. HaCohen-Kerner, Z. Gross, A. Masa, "*Automatic Extraction and Learning of Keyphrases from Scientific Articles*. *In: A. Gelbukh (ed.): CICLing 2005. Lecture Notes in Computer Science*, Vol.3406, Springer-Verlag, Berlin Heidelberg, 2005, pp.657-669.

[12] A. Hulth, J. Karlgren, A. Jonsson, H. Boström, "*Automatic Keyword Extraction Using Domain Knowledge*" , *In: A. Gelbukh (ed.): CICLing 2001. Lecture Notes in Computer Science*, Vol.2004, Springer-Verlag, Berlin Heidelberg, 2001, pp.472-482.

[13] Y. Matsuo, Y. Ohsawa, M. Ishizuka, "*KeyWorld: Extracting Keywords from a Document as a Small World", In: K. P. Jantke, A. shinohara (eds.): DS 2001. Lecture Notes in Computer Science*, Vol.2226, Springer-Verlag, Berlin Heidelberg 2001, pp.271-281.

[14] J. Wang, H. Peng, J-S. Hu, "*Automatic Keyphrase Extraction from Document Using Neural Network*", *ICMLC 2005*, 2005, pp.633-641.

[15] P. D. Turney, "Learning algorithm for keyphrase extraction", *Journal of Information Retrieval*, 2(4), 2000, pp.303-36.

[16] E. Frank, G. Paynter, I. H. Witten, C. Gutwin and C. Nevill-Manning, " *Domain-specific keyphrase extraction*", *In: proceeding of the sixteenth international joint conference on artificial intelligence*, San Mateo, CA,1999.

[17] I.H. Witten, G.W. Paynter, E. Frank, et al, "*KEA: Practical Automatic Keyphrase Extraction*", *In: E. A. Fox, N. Rowe (eds.): Proceedings of Digital Libraries'99: The Fourth ACM Conference on Digital Libraries*. ACM Press, Berkeley, CA, 1999, pp.254-255.

[18] N. Kumar, K. Srinathan, "*Automatic keyphrase extraction from scientific documents using N-gram filtration technique*", *In Proceeding of the eighth ACM symposium on Document engineering*, Sao Paulo, Brazil, 2008.

[19] Q. Li, Y. B. Wu, "Identifying important concepts from medical documents", *Journal of Biomedical Informatics*, 2006, pp.668-679.

[20] C. Fellbaum, "*WordNet: An electronic lexical database*", Cambridge: MIT Press, 1998.

[21] H. Liu, "*MontyLingua: An end-to-end natural language processor with common sense'*, 2004, retrieved in 2005 from http://www.web.media.mit.edu/~hugo/montylingua
[22] H. P. Edmundson, "New methods in automatic extracting". *Journal of the Association for Computing Machinery*, 16(2), 1969, 264-285.
[23] G.K. Zipf, "*The psycho-biology of language*", Cambridge MA: MIT press, 1935 (reprinted 1965).
[24] G. Holmes, A. Donkin and I. H. Witten, "*Weka: A machine learning workbench*". *In Proc Second Australia and New Zealand Conference on Intelligent Information Systems*, Brisbane, Australia, 1994.
[25] I. H. Witten, E. Frank, "*Data Mining: Practical machine learning tools and techniques",* 2nd Edition, Morgan Kaufmann, San Francisco, 2005. http://www.cs.waikato.ac.nz/~ml/weka/book.html, Retrieved in 2007.
[26] U.M. Fayyad, K.B. Irani, "*Multi-interval discretization of continuous-valued attributes for classification learning*", *In proceedings of 13th International Joint Conference on Artificial Intelligence*, San Francisco, CA, Morgan Kaufmann, pp.1022-1027, 1993.
[27] J. R. Quinlan, "C4.5: Programs for Machine Learning", *Morgan Kaufmann Publishers*, 1993.
[28] K. Sarkar, "*Automatic Keyphrase Extraction from Medical Documents", Proceedings of the 3rd International Conference on Pattern Recognition and Machine Intelligence (PReMI 09),* 2009, pp.273-278

**Kamal Sarkar**

He received his B.E degree in Computer Science and Engineering from the Faculty of Engineering, Jadavpur University in 1996. He received the M.E degree and Ph.D. (Engg) in Computer Science and Engineering from the same University in 1999 and 2011 respectively. In 2001, he joined as a lecturer in the Department of Computer Science & Engineering, Jadavpur University, Kolkata, where he is currently an associate professor. His research interest includes text summarization, natural language processing, machine learning, web mining, knowledge discovery from text data.

**Mita Nasipuri**

She received her B.E.Tel.E., M.E.Tel.E., and Ph.D. (Engg.) degrees from Jadavpur University, in 1979, 1981 and 1990, respectively. Prof. Nasipuri has been a faculty member of Jadavpur University since 1987. Her cur rent research interest includes image processing, pattern recognition, and multimedia systems. She is a senior member of the IEEE, U.S.A., Fellow of I.E (India) and W.B.A.S.T, Kolkata, India.

**Suranjan Ghose**

He received his B.E. degree in Electronics & Telecommunication Engineering and M.E.and Ph.D. degrees in Computer Science & Engineering from Jadavpur University, Calcutta, India in 1978, 1981 and 1988 respectively. From 1982 to 1987 he was a faculty member at the Indian Statistical Institute, Calcutta. In 1988 he joined the Department of Computer Science & Engineering, Jadavpur University, Calcutta, where he is currently a professor. His research interests include parallel algorithms & architectures, graphics algorithms, computer networks and fault-tolerant architectures.