Clustering Unstructured Text Documents Using Naïve Bayesian Concept and Shape Pattern Matching

Durga Toshniwal, Rishiraj Saha Roy

Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand 247667, India
durgafec@iitr.ernet.in, rishiraj.saharoy@yahoo.co.in
doi: 10.4156/ijact.vol1.issue1.8

Abstract

Clustering organizes text in an unsupervised fashion. In this paper, we propose an algorithm for clustering unstructured text documents using naïve Bayesian concept and shape-pattern matching. The Vector Space Model is used to represent our dataset as a term-weight matrix. In any natural language, semantically linked terms tend to co-occur in documents. Hence, the co-occurrences of pairs of terms in the term-weight matrix are observed. This information is used to build a term-cluster matrix where each term may belong to multiple clusters. The naïve Bayesian concept is used to uniquely assign each term to a single term-cluster. The documents are assigned to clusters using mean computations. We apply shape pattern-matching to group documents within the broad clusters obtained earlier. The proposed algorithm has been validated using benchmark datasets available on the Internet. Our results show that the proposed scheme has a significantly better running time as compared to traditional algorithms.

Keywords

Text Clustering, Naïve Bayesian Concept and Shape Pattern Matching.

1. Introduction

Nowadays, a substantial portion of the available information is stored in text document databases, which consist of large collections of documents from various sources, such as news articles, research papers, books, digital libraries, and e-mail messages [1]. Text databases are rapidly growing due to the increasing amount of information available in electronic form, such as electronic publications, various kinds of electronic documents, e-mail, and the World Wide Web (which can also be viewed as a huge, interconnected, dynamic text database). Nowadays most of the information in government, industry, business, and other institutions are stored electronically, in the form of text databases. Data stored in most text databases are semistructured data in that they are neither completely unstructured nor completely structured. For example, a document may contain a few structured fields, such as title, authors, and publication date, but it also contains some largely unstructured components, such as the abstract and the contents. There have been a great deal of studies on the modelling and implementation of semi-structured and unstructured data in recent database research. Moreover, information retrieval techniques, such as text indexing methods, have been developed to handle unstructured documents.

This paper is organized as follows. It comprises of a total of seven sections. In Section 1, an introduction to text mining has been given. In Section 2, the literature review performed before doing this work has been discussed. The research gaps thus found and the motivation for the work have also been given. The document model used to represent the text dataset has been explained in Section 3. The proposed design for the complete clustering scheme has been described in detail in Section 4. The experimental results and the relevant discussion have been given in Section 5. The conclusions drawn from the obtained results and the suggestions for future work have been given in Section 6. The references used for writing this paper have been given in Section 7.

2. Literature Review

Clustering is performed to organize text documents in an unsupervised manner. When text documents are represented in the form of vectors, common clustering methods that employ the concepts of distances, hierarchies, and densities among data objects can be applied. But the vector space almost always has a very large number of dimensions, due to the great number of terms present. A projection of the documents into a lower dimensional subspace brings the semantic structure of the document to light. After the operations
of dimension reduction have been performed, traditional clustering algorithms can be applied to obtain meaningful results efficiently. This curse of dimensionality poses a tough challenge for clustering and other text mining operations. Now a few recent text clustering approaches are described. A supervised feature selection method, named CHIR [2], has been proposed which is based on the \( \chi^2 \) statistic and new statistical data that can measure the positive term-cluster dependency. A new text clustering algorithm named TCFS has been proposed, which stands for Text Clustering with Feature Selection. TCFS can incorporate CHIR to identify relevant features (i.e., terms) iteratively, and clustering becomes a learning process. An approach to text clustering has been proposed in [3] which combines the advantages of the \( k \)-means algorithm and the Self-Organizing Map (SOM) techniques. A text-clustering algorithm of Frequent Term Set-Based Clustering (FTSC), which uses frequent term-sets for text clustering, has been proposed [4].

The application of the naïve Bayesian concept in classification will be discussed now. It will be explained in detail as it is one of the fundamental concepts used in the proposed algorithm. Bayesian classifiers are statistical classifiers [5]. They can predict a class membership probability, i.e., the probability that a given tuple belongs to a particular class [6]. Bayesian classification is based on Bayes’ theorem, described below. Studies comparing classification algorithms have found a simple Bayesian classifier known as the naïve Bayesian classifier to be comparable in performance with popular classifiers. Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases.

An architecture of a perception-based decision making system in a time series database domain which integrates perception-based TSDM has been proposed [7], computing with words and perceptions, and expert knowledge. The new tasks that should be solved by the perception-based TSDM methods to enable their integration in such systems have also been discussed. These tasks include the precisiation of perceptions, shape pattern identification, and pattern retranslation. Clipping is the process of transforming a real valued series into a sequence of bits representing whether each data is above or below the average [8]. Shape pattern-based similarity is one of the basic concepts used in this work. An algorithm has been proposed which applies a linguistic variable concept tree to describe the slope feature of time series, and has been named Shape Dynamic Time Warping [9].

Traditional text clustering algorithms attempt to find clusters among the documents directly, based on term weight vectors. So they have to deal with vectors of a very high dimensionality. Very few attempts were made to first cluster the terms on the basis of semantic correlation and then cluster the documents based on these term-clusters. The naïve Bayesian theory had been applied only to classifiers. Shape pattern-based similarity, a highly successful technique in time series data mining, had not yet been explored in the mining of text data, even though representing text documents as sequences has long been in practice.

3. Document Model

Initially the text data have to be preprocessed. This includes stopword removal, word stemming, and dimensionality reduction, preferable using locality preserving indexing scheme [10] if we are going to cluster the text data. Proposals of many models have been made for dealing with text mining problems. One of them is the Vector Space Model [11], the use of which has been made in this work. This model is briefly explained in this section. In the Vector Space Model, initially the digital documents are divided into terms by morphological analysis. A term is sometimes called a word. This is a preprocessing step called tokenization.

Let there be \( m \) documents in the corpus and \( n \) terms in the dictionary. Then each document can be modelled as a vector \( v \) in an \( n \)-dimensional space. This is why this model is called the Vector Space Model. The term frequency of a term \( t \) in a document \( d \) is the number of occurrences of \( t \) in \( d \). Let it be denoted by \( TF(d, t) \). There are ways to normalize this term frequency. For example, in this work, we use the Cornell-SMART system that uses the following formula to compute the normalized term frequency [1]:

\[
TF(d, t) = \begin{cases} 
0 & \text{if } freq(d, t) = 0 \\
1 + \log_{10}(1 + \log_{10}(freq(d, t))) & \text{otherwise}
\end{cases}
\] (1)

There is another important measure called the Inverse Document Frequency (IDF) that represents the scaling factor, or the importance, of a term \( t \). If a term \( t \) occurs frequently in many documents, its importance will be scaled down due to its reduced discriminative power. For example, the term ‘football’ is likely to be less relevant if it occurs in a set of news articles about a football tournament. So we need to scale down its importance accordingly. According to the same Cornell-SMART system, \( IDF(t) \) is defined by the following formula:

\[
IDF(t) = \log_{10}(1 + |d| / |d_t|)
\] (2)
Clustering Unstructured Text Documents Using Naïve Bayesian Concept and Shape Pattern Matching
Durga Toshniwal, Rishiraj Saha Roy

Figure 1. Complete block diagram of proposed clustering scheme

where \(|d|\) is the total number of documents, and \(|d_t|\) is the number of documents containing the term \(t\). Here \(|d_t|\) cannot be zero as then the term \(t\) would not have been included in the dictionary. In a complete vector-space model, the TF and the IDF measures are combined together, which forms the TF-IDF measure used throughout this work:

\[
TF-IDF(d, t) = TF(d, t) \times IDF(t) \quad (3)
\]

In a sample TF-IDF matrix, the \(i^{th}\) row represents a document vector for document \(d_i\), the \(j^{th}\) column represents the TF-IDF values for term \(t_j\), and each entry registers \(TF-IDF(d_i, t_j)\).

4. Proposed Work

An overview of our proposed algorithm has been given in Figure 1. Initially, the TF-IDF matrix is available. The algorithm to build the co-occurrence matrix from the TF-IDF matrix is presented in Figure 2. In this algorithm, the co-occurrences between terms are studied. When two terms co-occur in a document, the minimum of the number of their occurrences is taken as a co-occurrence measure.

\[
Co-occurrence(term i, term j) = Co-occurrence(term j, term i) \quad (4)
\]

As a result, if \(co-occurrence(term i, term j)\) has been computed previously, computation or storage of \(co-occurrence(term j, term i)\) is not necessary. This matrix is built in a row-major fashion, so the resulting co-occurrence matrix is an upper triangular matrix. We can now formally state the method used to build the co-occurrence matrix \(CoOccMat\) mathematically:

\[
CoOccMat(i, j) = \sum_{k=1}^{m} [\text{minimum}(TFIDFMat(k, i), TFIDFMat(k, j))] \\
= \begin{cases} 
-1 & \text{if } i < j \\
0 & \text{if } i = j \\
0 & \text{if } i > j 
\end{cases} 
\]

(5)

where \(m\) is the total number of documents and \(TFIDFMat\) is the input TF-IDF matrix.

Figure 2. Algorithm to obtain co-occurrence matrix
The term cluster matrix is now ready to be built using the co-occurrence matrix. The algorithm for this procedure is given in Figure 3.

**COOCC-TO-TERMCLUS** (CoOccMat, NumFeats)

1. for every term \(1 \) to \( \text{NumFeats} \)
2. do identify which terms it co-occurs with
3. put each such term in cluster of current term
4. terms which do not co-occur with any other term are put in their own clusters
5. for every term \(1 \) to \( \text{NumFeats} \)
6. do identify which clusters it belongs to
7. for every term in such cluster
8. do calculate co-occurrence probability with itself
9. compute products of all such probabilities (application of naïve Bayesian concept)
10. select highest probability
11. assign term finally to cluster with highest probability
12. terms which do not co-occur with any other term remain in their own clusters
13. return term-cluster matrix

**Figure 3.** Algorithm to obtain term-cluster matrix

In this algorithm, clusters are formed within the term-set. Terms which are linked semantically will be grouped under one cluster. It is assumed that terms which have a high degree of co-occurrence are likely to be linked semantically. In this work, it is also assumed that one term may belong to only cluster. A term is uniquely assigned to a single term-cluster. This is done by the application of conditional probability and the naïve Bayesian concept. The conditional probabilities of a term belonging to each of the possible clusters are calculated. It is assigned to the cluster with the highest probability. Now the algorithm which has been used will be explained. From the co-occurrence matrix obtained, the terms co-occur are known. Initially, each term is treated as a cluster centre and all terms co-occurring with this term are put into the cluster corresponding to this term. Terms which do not co-occur with any other term are the singular terms in their respective clusters. The initial term cluster matrix \( \text{IntlTermClusMat} \) is built from the co-occurrence matrix \( \text{CoOccMat} \) according to the following equation:

\[
\text{IntlTermClusMat}(i, j) = \begin{cases} 1 & \text{if } \text{CoOccMat}(i, j) \neq 0 \\ 0 & \text{otherwise} \end{cases}
\]

In the newly obtained matrix, the rows correspond to initial clusters and the columns to the member terms. The clashes have to be removed now and a term assigned uniquely to a cluster. The naïve Bayesian concept is used now. It is based on the assumption that a term’s probability of belonging to a particular cluster is independent of its probabilities of its belonging to the other clusters. This effectively translates to the fact that a term’s probability of co-occurrence with one term is independent of its probability of co-occurrence with another term. This assumption can be called cluster conditional independence. Just like the corresponding Bayesian classifiers, it is naïve in this regard. The bases of this concept are those of the Bayes’ theorem and conditional probability. Let \( X \) represent one of \( m \) terms and \( C_1, C_2, \ldots, C_m \) the term-clusters. Then, \( P(C_i|X) \) represents the posterior probability of term \( X \) belonging to cluster \( C_i \), given that \( X \) is known. It is also called the a posteriori probability of \( C_i \) conditioned on \( X \). In contrast, \( P(C_i) \) is the prior probability, or \( \text{apriori} \) probability, of \( C_i \). This is the probability of the cluster \( C_i \) being chosen at random from the \( m \) clusters. The posterior probability, \( P(C_i|X) \), is based on more information (i.e. knowledge of term number) than the prior probability, \( P(C_i) \), which is independent of \( X \). Similarly, \( P(X|C_i) \) is the posterior probability of \( X \) conditioned on \( C_i \). It is the probability that given the cluster chosen is \( C_i \), the term chosen is \( X \). \( P(X) \) is the prior probability of \( X \), i.e. the probability of the term \( X \) being chosen at random from the list of all terms. By the Bayes Theorem,

\[
P(C_i|X) = \frac{P(X|C_i) \times P(C_i)}{P(X)} \quad (7)
\]

Given a term \( X \), the clustering scheme will predict that \( X \) belongs to the term-cluster having the highest posterior probability, conditioned on \( X \). So it predicts that term \( X \) belongs to the cluster \( C_i \) if and only if \( P(C_i|X) > P(C_j|X) \) for \( 1 \leq j \leq m, j \neq i \). Thus we maximize \( P(C_i|X) \). The cluster \( C_i \) for which \( P(C_i|X) \) is maximized is called the maximum posteriori probability. \( P(C_i|X) \) is given by Equation (7). As \( P(X) \) is constant for all clusters, only \( P(X|C_i) \times P(C_i) \) needs be maximized. Since the cluster prior probabilities are not known, it is assumed that the clusters are equally likely, that is, \( P(C_i) = P(C_j) = \ldots = P(C_m) \), and we would therefore maximize \( P(X|C_i) \).

By the naïve assumption of cluster conditional independence, \( P(X|C_i) \) can be estimated in the following way:

\[
P(X|C_i) = \prod_{k=1}^{n} P(\text{co-occurrence of } X \text{ and } X_k)
\]

where \( n \) is the number of terms in \( C_i \) \quad (8)

\[
P(X|C_i) = P(\text{co-occurrence of } X \text{ and } X_1) \times P(\text{co-occurrence of } X \text{ and } X_2) \times \ldots \times P(\text{co-occurrence of } X \text{ and } X_n)
\]

where \( X_1, X_2, \ldots, X, \ldots X_n \) are the terms belonging to \( C_i \). \quad (9)
The probability of co-occurrence of terms \( X_i \) and \( X_j \) is defined by

\[
P(\text{co-occurrence of } X_i \text{ and } X_j) = \frac{\text{No. of co-occurrences of } X_i \text{ and } X_j}{\text{No. of co-occurrences of } X_i \text{ and all other terms}}
\]

(10)

The probability of co-occurrence of a term with itself (trivial case) is assumed to be one. There is another important modification to be introduced. In the product of Equation (9), if any of the co-occurrence probabilities is zero, it makes the whole product zero. But a term need not co-occur with every other term in its cluster. But without any modification to the existing calculations, non-co-occurrence with even a single term in a term-cluster would nullify the whole product. Without the zero probability, a high probability might have been possible, suggesting that \( X \) may have belonged to class \( C_i \). A zero probability cancels the effects of all of the other (posteriori) probabilities (on \( C_i \)) involved in the product. There is a simple trick to avoid this problem. It can be assumed that the training database is so large that adding one to each count that does not co-occur) is treated as 1, and similarly the denominator also gets increased by 1 in the probability calculations. Now the algorithm (Figure 4) for calculating the co-occurrence probability between two terms is presented. At the end of this process, the terms which did not co-occur with any other term still remain in their own clusters. After this round of final assignment, a term cluster matrix is obtained in which every term belongs to a single cluster only. So there is exactly a single one entry in a single column. As a row is traversed columnwise (corresponding to an equivalent cluster), the terms which have 1s in their corresponding locations belong to the cluster under consideration. Finally the term cluster matrix is converted into a memory-efficient bag-of-words representation. This means that instead of a row containing 0s in locations of terms not belonging to the cluster and 1s in locations of terms belonging to the cluster, the row directly contains the identifiers of the terms belonging to the cluster. This matrix is the final term cluster matrix and is used for document cluster and sub-cluster determination.

CALC-COOCC-PROBAB(term1, term2, CoOccMat, NumFeats)
1 numerator ← number of co-occurrences between term1 and term2 obtained from CoOccMat
2 MinVal is the minimum of the two terms
3 MaxVal is the maximum of the two terms
4 numerator ← CoOccMat[MinVal][MaxVal]
5 2 denominator ← number of co-occurrences between term1 and all other terms
6 3 for every element in term1-th row and term1-th column in CoOccMat
7 4 do increment denominator by corresponding value
8 5 if numerator = 0 ► Laplacian correction
9 6 then numerator ← numerator + 1
10 7 denominator ← denominator + 1
11 8 if denominator ≠ 0
12 9 then probability ← numerator / denominator
13 else probability ← 0
14 return probability

Figure 4. Algorithm to calculate co-occurrence probability

document clusters. This is done by computing the arithmetic mean of the TF-IDF values corresponding to the terms of every cluster, sequentially. The document will be assigned to the cluster yielding the highest mean. The main implication of this is that the number of document clusters is equal to the number of term-clusters. It does not vary with the number of documents, provided the number of terms remains fixed. This is very helpful as the number of documents \( D \) is generally much larger than the number of terms \( N \) \( (D >> N) \). As a result, the number of term-clusters is also much lower than \( D \). This helps us divide a large document set into a manageable number of clusters. Mathematically, the cluster number of document number \( i \) is given by

\[
\text{Cluster}(i) = \max\{\frac{\sum_{j=1}^{n} TFIDFMat(i, FinalTermClusMat(p, i))}{n}\}
\]

(11)

where \( TFIDFMat \) is the TF-IDF matrix \( FinalTermClusMat \) is the final term-cluster matrix

and, the maximization is performed over all \( p \), i.e. all term-clusters;

\( n \) is the number of terms in each cluster; so \( n \) may vary from cluster to cluster

The algorithm for this procedure is given in Figure 5. The document clustering results are stored in a document cluster matrix which has three columns and a number of rows equal to the number of documents. The first column stores the document identifier, the second column stores the document cluster identifier,
and third column is allocated to store the document sub-cluster identifier. After this first level of

CLUS-BY-MEAN(TFIDFMat, FinalTermClusMat, NumDocs, NumFeats)
1 for every document from 1 to NumDocs
2 do for every term cluster
3 do compute arithmetic mean of values in
document vector of current document in
TFIDFMat corresponding to terms in
current term-cluster
4 assign document to term cluster with highest mean
5 return final document cluster matrix with cluster information but
without sub-cluster information

**Figure 5.** Algorithm for document clustering

clustering is performed, this matrix is returned but the third column, as expected, is still empty. It is filled in only after the sub-clustering procedure is completed.

The document clusters provides a broad grouping of the documents. Often a finer level of clustering is required which is provided by the sub-clustering procedure. Here the representation of text documents as sequences in the form document vectors is of fundamental importance. Here the concept of shape pattern-based similarity is applied. A logical graph is assumed consisting of the points in the TF-IDF matrix corresponding to the cluster of the document. The TF-IDF values (equivalently term weights) (y-axis) are observed against the terms (x-axis). Here the word ‘observed’ is used and not ‘plotted’ because though shapes and graphs are conceptually being dealt with, explicit plotting and a manual study of the graphs are not necessary. The shape of this plot gives the inherent pattern associated with a document. Computations on the document vectors help in performing the equivalent operations. The graphical representations, as provided in the figures later, give an easy illustration of the concept. The algorithm for the sub-clustering procedure is presented in Figure 6. The sub-clustering procedure is also fully unsupervised and based on the notion of the relative importance of the various terms in the term-cluster in the document under consideration. This is reflected by the changes that the TF-IDF values go through corresponding to the terms in the term-cluster of the document. Let there be \( k \) terms in the term-cluster of the document under consideration. This corresponds to \( k \) points on the x-axis. Corresponding to the \( k \) points in a term-cluster, there are \((k - 1)\) transition points of importance in the graph. The differences in the TF-IDF values over consecutive points are of interest and help in determining the shape pattern present in the plot. These differences help in determining the gradient of the graph as it moves across these transition points. A \((k - 1)\)-character array for every document is maintained

SUB-CLUS-BY-SHAPE(TFIDFMat, FinalTermClusMat, DCM, NumDocs, NumFeats)
1 declare ShapeList to store list of unique patterns
2 initially ShapeList contains only end-marker
3 for every document from 1 to NumDocs
4 declare and initialize string to store associated shape pattern
5 do fetch TF-IDF values in document vector corresponding to terms of term-cluster
6 for every pair of consecutive terms in term-cluster
7 do observe difference between corresponding TF-IDF values
8 if TF-IDF value corresponding to second term higher
9 then add U to current pattern as graph moves Up
   Here graph refers to plot of TF-IDF values versus corresponding terms
10 else if TF-IDF value corresponding to second term lower
11 then add D to current pattern as graph moves Down
12 else (TF-IDF values equal)
13 add L to current shape pattern as graph remains Level
14 compare shape pattern with every pattern in ShapeList sequentially
15 if match is found
16 then associate document with current shape identifier
17 else
18 add new shape to ShapeList
19 push end marker by one position
20 associate document with new shape identifier
21 Sort in ascending order of shape indices within clusters
22 Assign first document to first sub-cluster
23 for \( i \leftarrow 1 \) to NumDocs
24 do if shapes match and clusters match for consecutive documents
25 then assign documents to previous sub-cluster
26 else if shapes do not match or clusters do not match
27 then create new sub-cluster and assign document to it
28 return final document cluster matrix with sub-cluster information

**Figure 6.** Algorithm for document sub-clustering

which stores the alphabets ‘U’, ‘D’, or ‘L’ according as the graph moves up, down, or remains level (three possibilities) across a transition point, in sequence, i.e. this array stores the description of the shape pattern present in the document’s graph. As a result, there will be a total of \( 3^{k-1} \) possible shapes inherent in the document vectors, a number which may become quite large for a large \( k \). But even for large real datasets, only a much reduced set of shape patterns appear (the number of patterns discovered are only of significance within a sub-category, and not across them; as explained later). This has been shown experimentally in Section 5.
Whenever a new document is encountered, the shape array for this document is compared to the arrays of the existing shapes, which are maintained separately in a text file. If the pattern matches with an existing one, the index number for this shape (shape identifier) is assigned to the document. If it is a new shape, the next unique serial number is assigned to the shape and the document, and the pattern is added to the list of existing shapes. This numbering is done on a global basis, i.e. two different shapes always have different serial numbers, even if they appear in different sub-categories only. This simplifies the indexing procedure without increasing any time or space requirement. The set of all the indices of the obtained shape patterns forms the shape alphabet. Shape identifier 0 (null) is reserved for documents with clusters where the number of terms is one, i.e. a case when no pattern can be formed. Documents within a particular cluster with the same shape pattern (or equivalently sharing the same shape identifier) form sub-clusters. This completes the clustering procedure within the clusters based on shape patterns. Let us take an example. Say, a document belongs to a cluster with five terms in it. So there are \((k - 1)\), i.e. 4 transition points. Let the corresponding TF-IDF values be \(\{9, 16, 16, 21, 6\}\). Then the associated graph can be said to move up, remain level, again move up, and finally move down. As a result, the associated shape pattern will be \(\{U, L, U, D\}\) (Figure 7). It is to be noted that only the shape of the pattern (and not the magnitude of a rise or a fall) is sufficient to reflect the importance of the respective terms within the document, which is the basis for our sub-clustering procedure. The sub-cluster identifiers are copied back into the third column of the document cluster matrix. A vector from this matrix may be represented as

\[
\{\text{document}_\text{id}, \text{cluster}_\text{id}, \text{sub-cluster}_\text{id}\}
\]

**Figure 7.** Shape pattern \(\{U, L, U, D\}\)

### 5. Results and Discussion

We have used a variety of benchmark datasets [12] available on the internet to validate our algorithm. The details of these datasets are given below, in increasing order of complexity.

**Case 1:** The TF-IDF matrix corresponds to a set of five thousand documents and fifty terms. The term set consists of groups of co-occurring terms, with no co-occurrence between terms of different groups.

**Case 2:** The TF-IDF matrix corresponds to a set of five thousand documents and fifty terms. The term set consists of groups of co-occurring terms, but with co-occurrence between terms of different groups.

**Case 3:** The TF-IDF matrix corresponds to a set of five thousand documents and two hundred terms. The term set consists of groups of co-occurring terms, with co-occurrence between terms of different groups.

**Case 4:** We deal with two special cases in the last two datasets. The first one is named ADA has marketing applications. The task of ADA is to discover high revenue people from census data, presented in the form of a two-class classification problem. The raw data from the census bureau is known as the Adult database in the UCI machine-learning repository. The fourteen original attributes (features) represented age, workclass, education, marital status, occupation, and native country. They included continuous, binary and categorical attributes. They were finally aggregated to form a data matrix corresponding to forty six thousand and thirty three text documents, with forty eight terms, each term representing an attribute. We have used the first five thousand rows and all the forty eight columns for our work.

**Case 5:** The last dataset is named SYLVA, an ecology application. The task of SYLVA is to classify forest cover types. The forest cover type for \(30 \times 30\) metre cells was obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. The problem dealt with the study of Ponderosa pine versus everything else. The input matrix consisted of one lakh, forty five thousand, two hundred and fifty two rows (documents) (out of which we have used the first five thousand to maintain uniformity among the datasets) and two hundred and sixteen input variables (terms) (all have been considered).

For each of the five datasets listed, we give the number of documents, terms, clusters and sub-clusters, the average number of sub-clusters per cluster, and the average number of documents per cluster and sub-cluster. This summary is given in Table I (legend is at the bottom of the table). Due to space constraints, we had to use the abbreviated forms in the column headers. We gradually vary the size of the dataset and observe the change in our metrics, keeping the number of terms constant. We also record how our metrics vary with the number of terms when we vary the number of terms, keeping the number of documents fixed. For all
the five cases (case 3 has two parts as shown in Table 1), we plot graphs for the results (Figs. 8 through 13) and then explain our findings.

Now that we have presented the graphs, we will explain the findings. To begin with, we observe that for each of the datasets 1, 2, and 3 (Part I), the number of clusters does not vary with the number of documents (resulting in the steady increase of ANDPC). This is due to the fact that the document clustering is a two stage process: the first being clustering of the terms, and the second being the assignment of the document to the term-cluster with the highest corresponding TF-IDF mean. So, if the number of terms is kept constant, the number of document clusters will not vary with the number of the documents. This has the great advantage of managing the large corpus with a reasonable number of clusters (since number of terms << number of documents). It is also a logical conclusion of the fact that for a reasonably large document database, unless the dictionary is expanded, the number of document categories will not change. The number of clusters detected in Cases 1 and 2 strongly prove that our naïve Bayesian assumption works well. The difference between these two datasets was that there were terms overlapping with more than one well-formed cluster, strongly with one and weakly with the others. There were also stray distracting terms which did not form a cluster of their own but tried to destabilize the structure of well-formed clusters. Otherwise, the well-formed term-clusters were the same in both these datasets. Our scheme has been successful in nullifying the effect of the stray terms (also evident in Cases 4 and 5 analyzed later) and also in uniquely assigning overlapping terms to the cluster with whose terms which it had the strongest co-occurrence. To demonstrate the effect of a change in the number of terms, we have varied the number of terms from forty through two hundred keeping the number of documents fixed at five thousand for the dataset of Case 3. The results then display a change in the number of clusters initially, but later become almost constant. But simply the number of clusters does not reveal the full picture here. We observe that the numbers of clusters are 25, 58, 58, 58, and 57 when the numbers of features are 40, 60, 80, 120, and 160 respectively. But initially the 58 clusters all contained only one or two terms each. We had mostly single-term clusters of trivial real-world use. As the number of terms grew, the clusters became meaningful, and began to contain reasonable numbers of terms like three to six. For space constraints, we are not able to provide the number of terms in each term-cluster formed or the number of documents in each document cluster; otherwise this behavior would have been apparent. With the increase in the number of terms, the number of associated shape patterns within term-clusters also increase, increasing steadily the sub-cluster count. But since the number of documents is kept constant, the average number of documents per sub-cluster decreases monotonically, though at a very slow rate. This is because the rate of increase in the number of sub-clusters (due to the appearance of new shape patterns) is less than the rate at which new terms are added. But this step is done only as a demonstration, as increasing the number of terms while keeping the number of documents constant does not have much significance in real life, whereas the reverse is the case in most text clustering applications like organizing documents for a news agency or for a research conference. Adding new documents incrementally (keeping the number of terms constant) results in the appearance of new shape patterns within the existing clusters. As a result, we observe the trend of an increasing NSC, ANSCPC, and ANDPSC with an increase in the number of documents for each of the datasets 1 through 3 (Part I). Coming to the special datasets, we observe that although there were minor deviations, the average number of clusters detected for Case 4 data was two. This confirms our prior knowledge about the dataset. Case 5 data was found to have only one cluster, again, as known earlier. This confirms that our algorithm is capable of detecting true clusters from large datasets even when a large number of the terms are distractors (having stray non-zero values) and the actual number of clusters is as low as one or two. For both cases, as the number of clusters is low, ANDPC is very high. For Case 5, since the number of terms in the special clusters is much higher than normal, the associated number of shape patterns that it may give rise to is also very high (3\text{Number of terms in term-cluster}). As a result, we have a very high NSC and very low ANDPSC. But the notion of sub-clusters does not have much significance for these two cases.

All traditional text clustering algorithms (k-means, EM, farthest-first, and density-based) require the number of desired clusters as user input. But our clustering scheme does not require any user input or domain knowledge. It determines the inherent clusters present within the documents based on semantically linked terms. There is also no sub-clustering feature available in standard algorithms. As a result, we have adopted running time to be the main performance metric between our scheme (level I) and the standard algorithms (available in WEKA [13]). Both of the systems have been run on the same Java platform (with the number of clusters detected by our system as the input to the standard algorithms). These results are tabulated in Table 2 (legend at the bottom).

From Table 2 (especially the shaded regions), we can easily see that our algorithm’s average running time is
significantly better than the standard algorithms for the same number of clusters detected. This is because all the standard algorithms tend to find clusters on a global basis, treating the entire document vector as a unit entity. As a result, they have to constantly deal with vectors of a very high dimensionality. Our algorithm tries to find local entities (term-clusters) within the term-set first and then clusters the documents on the basis of these local entities. Thus we look at local entities preserving the global structure of the document vector.

Let the number of documents and terms be \( m \) and \( n \) respectively. The approximate total running of our algorithm in the document clustering level, \( T(m, n) \), is \( O(mn^2) \), in the best, the worst, and the average cases. So the proposed algorithm in this level has a running time which varies linearly with the number of documents and quadratically with the number of terms, for all the three cases. For the sub-clustering level, the analysis is performed relative to a single cluster as it is a process associated with each cluster independently. Let there be \( p \) documents and \( q \) terms in the cluster. Let the total approximate running time be denoted by \( T(p, q) \). Then \( T(p, q) \) is \( O(p) \) in the best case and \( O(3^p) \) in the worst and the average cases. So the proposed algorithm in the document-sub-clustering level has a running time which varies linearly with the number of documents \( p \) in the cluster in the best case, and with the order of \( 3^p \) in the worst and the average cases.

<table>
<thead>
<tr>
<th>Case</th>
<th>ND</th>
<th>NT</th>
<th>NC</th>
<th>NSC</th>
<th>ANSCPC</th>
<th>ANDPC</th>
<th>ANDPC</th>
<th>ANDPSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>50</td>
<td>16</td>
<td>51</td>
<td>3.189</td>
<td>62.500</td>
<td>19.608</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2000</td>
<td>50</td>
<td>16</td>
<td>77</td>
<td>4.813</td>
<td>125.000</td>
<td>25.974</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3000</td>
<td>50</td>
<td>16</td>
<td>80</td>
<td>5.000</td>
<td>187.500</td>
<td>37.500</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4000</td>
<td>50</td>
<td>16</td>
<td>89</td>
<td>5.563</td>
<td>250.000</td>
<td>44.944</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5000</td>
<td>50</td>
<td>16</td>
<td>90</td>
<td>5.625</td>
<td>312.500</td>
<td>55.556</td>
<td></td>
</tr>
<tr>
<td>Avg.</td>
<td>3000</td>
<td>50</td>
<td>16.0</td>
<td>77.4</td>
<td>4.838</td>
<td>187.500</td>
<td>36.716</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1. Summary of cluster information for all datasets**

ND = Number of documents, NT = Number of terms, NC = Number of clusters, NSC = Number of sub-clusters, ANSCPC = Average number of sub-clusters per cluster, ANDPC = Average number of documents per cluster, ANDPSC = Average number of documents per sub-cluster, Avg. = Average
Table 2. Comparison of running times (in milliseconds)

<table>
<thead>
<tr>
<th>Case</th>
<th>ND</th>
<th>NT</th>
<th>NC</th>
<th>Proposed scheme</th>
<th>Simple k-means</th>
<th>EM</th>
<th>Density-based</th>
<th>Farthest-first</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>50</td>
<td>16</td>
<td>927</td>
<td>1016</td>
<td>952</td>
<td>971</td>
<td>1007</td>
</tr>
<tr>
<td>1</td>
<td>2000</td>
<td>50</td>
<td>16</td>
<td>2177</td>
<td>2691</td>
<td>2291</td>
<td>2347</td>
<td>2347</td>
</tr>
<tr>
<td>1</td>
<td>3000</td>
<td>50</td>
<td>16</td>
<td>3298</td>
<td>3849</td>
<td>3458</td>
<td>3511</td>
<td>3801</td>
</tr>
<tr>
<td>1</td>
<td>4000</td>
<td>50</td>
<td>16</td>
<td>4472</td>
<td>5055</td>
<td>4498</td>
<td>4527</td>
<td>4888</td>
</tr>
<tr>
<td>1</td>
<td>5000</td>
<td>50</td>
<td>16</td>
<td>5653</td>
<td>6316</td>
<td>5723</td>
<td>5911</td>
<td>6089</td>
</tr>
<tr>
<td>Avg.</td>
<td>3000</td>
<td>50</td>
<td>16</td>
<td>3305.4</td>
<td>3785.4</td>
<td>3844.4</td>
<td>3453.4</td>
<td>3666.0</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>50</td>
<td>16</td>
<td>1135</td>
<td>1475</td>
<td>1913</td>
<td>1698</td>
<td>3959</td>
</tr>
<tr>
<td>2</td>
<td>2000</td>
<td>50</td>
<td>16</td>
<td>2261</td>
<td>3016</td>
<td>2453</td>
<td>2782</td>
<td>2946</td>
</tr>
<tr>
<td>2</td>
<td>3000</td>
<td>50</td>
<td>16</td>
<td>3362</td>
<td>3875</td>
<td>4092</td>
<td>3045</td>
<td>3631</td>
</tr>
<tr>
<td>2</td>
<td>4000</td>
<td>50</td>
<td>16</td>
<td>4682</td>
<td>4790</td>
<td>4699</td>
<td>4850</td>
<td>4751</td>
</tr>
<tr>
<td>2</td>
<td>5000</td>
<td>50</td>
<td>16</td>
<td>5775</td>
<td>5905</td>
<td>5604</td>
<td>5811</td>
<td>5764</td>
</tr>
<tr>
<td>Avg.</td>
<td>3000</td>
<td>50</td>
<td>16</td>
<td>3443.6</td>
<td>3812.2</td>
<td>3752.2</td>
<td>3637.2</td>
<td>4210.2</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>200</td>
<td>57</td>
<td>12645</td>
<td>13789</td>
<td>12680</td>
<td>13003</td>
<td>14804</td>
</tr>
<tr>
<td>3</td>
<td>2000</td>
<td>200</td>
<td>57</td>
<td>27341</td>
<td>30067</td>
<td>28394</td>
<td>29561</td>
<td>29872</td>
</tr>
<tr>
<td>3</td>
<td>3000</td>
<td>200</td>
<td>57</td>
<td>42091</td>
<td>49007</td>
<td>44509</td>
<td>46712</td>
<td>47222</td>
</tr>
<tr>
<td>3</td>
<td>4000</td>
<td>200</td>
<td>57</td>
<td>57802</td>
<td>69691</td>
<td>59012</td>
<td>63423</td>
<td>61571</td>
</tr>
<tr>
<td>3</td>
<td>5000</td>
<td>200</td>
<td>57</td>
<td>72524</td>
<td>89880</td>
<td>74789</td>
<td>77820</td>
<td>71453</td>
</tr>
<tr>
<td>Avg.</td>
<td>3000</td>
<td>200</td>
<td>57</td>
<td>42480.6</td>
<td>50486.8</td>
<td>43876.8</td>
<td>46103.8</td>
<td>44984.4</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>48</td>
<td>2</td>
<td>1115</td>
<td>17</td>
<td>16</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>2000</td>
<td>48</td>
<td>2</td>
<td>1877</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>3000</td>
<td>48</td>
<td>2</td>
<td>3036</td>
<td>16</td>
<td>16</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>4000</td>
<td>48</td>
<td>2</td>
<td>4061</td>
<td>17</td>
<td>16</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>5000</td>
<td>48</td>
<td>2</td>
<td>5723</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>Avg.</td>
<td>3000</td>
<td>48</td>
<td>2</td>
<td>3162.4</td>
<td>16.0</td>
<td>16.0</td>
<td>17.0</td>
<td>16.6</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>216</td>
<td>1</td>
<td>20313</td>
<td>16</td>
<td>18</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>2000</td>
<td>216</td>
<td>1</td>
<td>39360</td>
<td>17</td>
<td>16</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>3000</td>
<td>216</td>
<td>1</td>
<td>58224</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>4000</td>
<td>216</td>
<td>1</td>
<td>77844</td>
<td>16</td>
<td>18</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>5000</td>
<td>216</td>
<td>1</td>
<td>95383</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>Avg.</td>
<td>3000</td>
<td>216</td>
<td>1</td>
<td>58224.8</td>
<td>16.2</td>
<td>16.8</td>
<td>16.8</td>
<td>17.2</td>
</tr>
</tbody>
</table>

ND = Number of documents, NT = Number of terms, NC = Number of clusters, EM = Expectation Maximization, Avg. = Average
6. Conclusion

In this work, we have proposed a novel two-level text clustering method based on the naïve Bayesian concept and shape pattern matching. In the first level, clusters are detected in the document set. Unlike traditional clustering algorithms, we first proceed to cluster the term-set based on their co-occurrence in the dataset. When a term is found to co-occur non-trivially with terms of more than one cluster, we use the naïve Bayesian concept of conditional independence to assign the term uniquely to one of the clusters. The basis of this term-clustering operation is to bring out the underlying semantic linkages between the terms. The clustering of the documents is then performed on the basis of these term-clusters using simple arithmetic mean computations on the TF-IDF values corresponding to the various clusters. Knowledge of semantic relationships within the terms helps in producing better clusters qualitatively. The sparse matrix representation is used wherever possible to reduce memory usage, as most of the data matrices used for stepwise computational purposes are not densely populated. The document clusters provide us with a broad grouping of the documents. In the second level, we exploit shape pattern-based similarity to find sub-clusters within the document clusters. Shape patterns inherent in the document vectors reflect the relative importance of the terms present within the document. They are used as a discriminatory measure to group documents within a cluster such that documents within a sub-cluster have the same relative importance attached to their terms. We performed an exhaustive comparison between the running times of our scheme and the traditional clustering algorithms available in WEKA. Our results show that the running time of our algorithm is significantly better than the...
others. This is because all the standard algorithms tend to find clusters on a global basis, treating the entire document vector as a unit entity. As a result, they have to constantly deal with vectors of a very high dimensionality. Our algorithm tries to find local entities (term-clusters) within the term-set first and then clusters the documents on the basis of these local entities. Thus we look at local entities preserving the global structure of the document vector. It also detects the major clusters successfully in large datasets when a major number of the terms are of trivial importance, their stray non-zero values acting as distractors trying to destabilize the structure of well-formed clusters. Moreover, our clustering scheme does not require any user input or domain knowledge. It detects the inherent clusters present within the dataset based on semantically linked terms. The number of document clusters does not vary with the dataset size, as long as the term-set is kept fixed. This has a big advantage of managing a large corpus with a reasonable number of clusters (since number of terms << number of documents). This is demonstrated by our results. It is also a logical conclusion from the fact that if our initial dataset size is reasonably large, then if the dictionary is not expanded by adding new terms, new clusters whose documents are semantically linked are also less likely to be produced. Our algorithm will be computationally expensive and will not work well when there is a large degree of co-occurrence between the terms, causing terms to be candidates for almost every initial term-cluster. But in these situations, the structures of the clusters are not well-defined; and as such any clustering algorithm would produce poor results.

As future work, we may devise efficient indexing methods that would allow us to store the cluster information and retrieve details relevant to a few clusters only (which we may want to work with). This will highly contribute to the saving of computational space required, and subsequently in the scalability of the overall process for dealing with large document sets. This is a potential area of improvement. The concept of shape pattern-based similarity may be applied to other text mining operations. We may also introduce more precision if we analyze a single shape pattern further by its gradients. For example, the transition ‘up’ can be made more specific by introducing ‘increasing’, ‘slowly increasing’, and ‘quickly increasing’. This would simply mean introducing gradient thresholds before the determination of the shapes.

7. References

[12] National Science Foundation (NSF), http://www.models.ei.tohu.ac.jp/