International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 27355-27373

http://www.ripublication.com

A Survey on Classification and Clustering Algorithms For Uncompressed and Compressed Text

Akshay S[1], Nayana K[2] , Karthika S[3]
Department of Computer Science, Amrita Vishwa Vidyapeetham,Mysore,India.
akshay_rao26@yahoo.com[1], nayanababuk@gmail.com[2]
karthikamallika@gmail.com[3]

Abstract

Internet is a pool of information, which contains billions of text documents which are stored in compressed format. In literature there are many text classification algorithms which work on uncompressed text documents. Since web pages contain text data which are stored in compressed format and the text documents must be taken back to its original format for the purpose of data mining activities. The process of decompression of text documents consumes more computational time. So this work introduces a study on different text classification and clustering algorithms and their comparison in compressed domain. Various methods for representing text in compressed domain are explained and experiments are conducted on LZW method for comparison. Different classification and clustering algorithms are also discussed. A comparative analysis on all these methods is presented.

Keywords: Compressed text, clustering, LZW, K means Clustering.

Introduction

Due to the availability of documents in digital form, content based document management task has gained much importance in the information system field. Text Documents are the most common type of information store house especially with the increased use of the internet. Automatic document management task has gained much of importance in the field of information retrieval for the past several decades. Text Classification consists of many challenges and difficulties. Firstly it is difficult to capture high level semantics and abstracts concepts of natural languages just from few key words. Furthermore semantic analysis which is a major step in designing an information retrieval system is not well understood. High dimensionality and variable length, content and quality are responsible for efficiency and accuracy demands on classification systems.

Text classification is one of the important research issues in the field of text mining, where the documents are classified with supervised knowledge. Text Processing System has three different stages. They are Pre Processing stage, Representation state and Classification stage or Clustering stage.

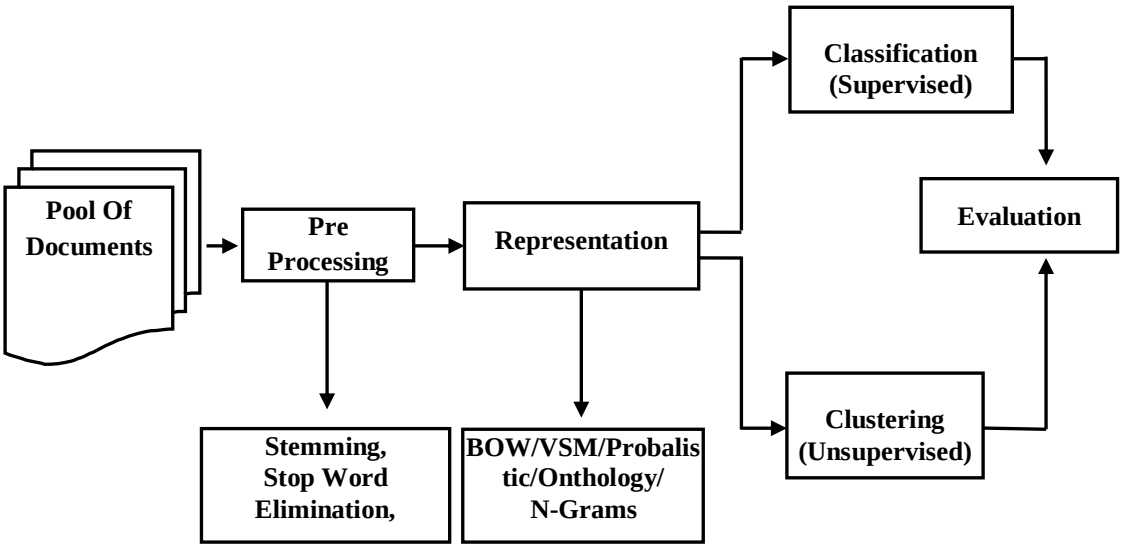

Figure 1: Block diagram of text processing system

A. Pre-processing:

It is one of the text encoding techniques. Text pre-processing means splitting a text document into stream of words by removing all punctuation marks and other non-text characters by replacing them with tabs and single white spaces. Preprocessing can be done using the following methods.

Stemming: A stem is a portion of a word which is left after the removal of its affixes. (i.e., prefixes and suffixes). For example, *connect* is the stem for the variants *connected, connecting, connection* and *connections.* Stemming reduces variants of the same root to a common concept and it also reduces the size of the indexing structure.

Stop Word Elimination: Eliminating stop word is nothing but filtering out words with low discrimination values. For example, *a, the, this, that, where, when… etc.* Stop word elimination reduces indexing structure considerably.

Pruning: It discards words appearing rarely or more frequently in a document.

Detagger*:* It finds the special tags in the documents and filters them.

Tokenizer*:* It identifies nonempty sequence of characters, excluding spaces and punctuation in documents.

Term Weighting: It is a process of assigning weights to each term based on its frequency. Any of the above mentioned preprocessing techniques can be employed to convert an unstructured data into a structured data.

B. Representation:

Representation means the pre-processed documents which are represented in the desired form. Usually it is represented in the form of Bag of Words and Vector Space model.

Bag of Word Representation: The BoW model allows a dictionary-based modeling, and each document looks like a "bag" (thus the order is not considered), which contains some words from the dictionary. For example,

A : John likes to watch movies. Mary likes too.

B: John also likes to watch football games.

Based on these two text documents, a dictionary is constructed as:

Dictionary ={1:"John", 2:"likes", 3:"to", 4:"watch", 5:"movies", 6:"also", 7:"football", 8:"games", 9:"Mary", 10:"too"},

This has 10 distinct words. And using the indexes of the dictionary, each document is represented by a 10-entry vector:

A : [1, 2, 1, 1, 1, 0, 0, 0, 1, 1] B: [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]

where each entry of the vectors refers to count of the corresponding entry in the dictionary.

Vector Space Model*:* The process of representing a text document in the form of vectors of weighted words is called Vector Space Model. The most popular representation model used in information retrieval and text mining is a matrix representation. This model represents documents as a vector in *m-dimensional space*.

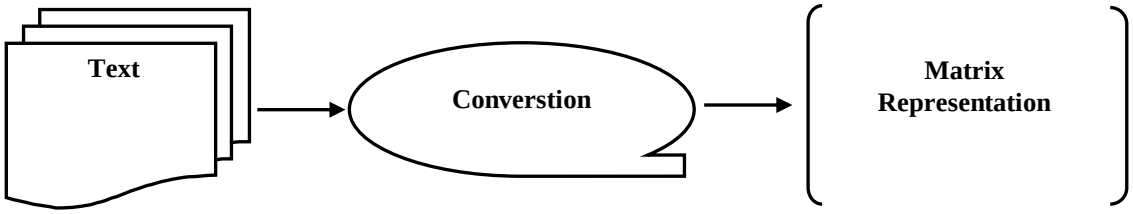

Figure 2: Vector Space Model

Let D be the set of documents (d_1,d_2,d_3…….d_n), a text document is represented as a vector of terms (t_1,t_2,t_3……t_n). The size of VSM is $d_i \times t_j$, where i = 1…..n and j = 1……m. Each cell in matrix represents the term frequency of the word in the particular document.

Dimensionality reduction technique: Feature selection and feature extraction are the major techniques of dimensionality reduction. We use feature extraction for our model. Feature extraction transforms the data from the high-dimensional space to a space of fewer dimensions. The data transformation may be linear, as in Principal Components Analysis, but many non-linear techniques also exist. We reduced the term document matrices of source and plagiarized documents using principle component analysis.

The main linear technique for dimensionality reduction, Principal Components Analysis (PCA), performs a linear mapping of the data to a lower dimensional space in such a way, that the variance of the data in the low-dimensional representation is

maximized. Eigen space is calculated by identifying the eigenvectors of the covariance matrix derived from a set of source documents. The eigenvectors corresponding to non-zero eigen values of the covariance matrix form an ortho normal basis that rotates and/or reflects the text document in the N-dimensional space. Specifically, each document is stored in a vector of size N.

$$x^i = [x_1^i, x_2^i, ..., x_N^i]^T$$

The documents are mean centered by subtracting the mean document from each document vector.

$$\bar{x}^i = x^i - m,$$

where

$$m = \frac{1}{p}\sum_{i=1}^{p} x^i$$

These vectors are combined, side-by-side, to create a data matrix of size NxP (where P is the number of documents).

$$\overline{X} = [\bar{x}^1 | \bar{x}^2 | \ldots | \bar{x}^p]$$

The data matrix X is multiplied by its transpose to calculate the covariance matrix.

$$\Omega = \overline{XX}^T$$

This covariance matrix has up to P eigenvectors associated with non-zero eigenvalues, assuming P<N. Here V is the set of eigenvectors associated with the eigenvalues**.**

Order the eigenvectors $v_i \in V$ according to their corresponding eigenvalues $\lambda_i \in \Lambda$ from high to low. The eigenvector associated with the largest eigenvalue is the eigenvector that finds the greatest variance in the documents. The eigenvector associated with the second largest eigenvalue is the eigenvector that finds the second most variance in the documents. This trend continues until the smallest eigenvalue is associated with the eigenvector that finds the least variance in the documents. This matrix of eigenvectors is the eigenspace V **,** where each column of V is an eigenvector.

$$V = [v_1 | v_2 | \ldots | v_p]$$

Each of the centered source document ($\bar{x}^i$) is projected into the eigenspace. To project the document into the eigenspace, calculate the dot product of the image with each of the ordered eigenvectors.

$$\tilde{x}^i = V^T \bar{x}^i$$

Therefore, the dot product of the document matrix and the first eigenvector will be the first value in the new vector. The new vector of the projected document matrix

will contain as many values as eigenvectors. Thus, the eigenspace is created for source documents. Next, the source documents are projected into the eigenspace.

C. Classification:

Classification is also termed as supervised learning. Classification aims at assigning pre-defined labels to text documents. Its main objective is to develop a model or profile or labels for each class. Classification methods can be categorized as follows:

- Decision Tree.
- Naive – Bayes Classifier.
- Nearest Neighbour Classifier.
- Neural Networks.

D. Clustering:

It is also termed as unsupervised learning. It can be used in order to find group of documents with similar contents. It is a collection of data objects, which are similar to one another with in the same cluster but dissimilar to objects in another cluster. Here there is no pre-defined classes are assigned. Methods of clustering are mainly,

- Partitional Clustering.
- Hierarchical Clustering.
- K-Means Clustering.
- Density based Clustering.

Classifiers For Uncompressed Documents

The task of text classification is to assign a Boolean value to each pair $(d_j, k_i) \in D_o \times K,$ where D_o is the domain of document and K is a set of predefined categories. The task is to approximate the true function $\phi : D_o \times K \rightarrow 1,0$ by means of a function $\hat{\phi} : D_o \times K \rightarrow \{1,0\}$ such that ϕ *and* $\hat{\phi}$ coincide as much as possible. The function $\hat{\phi}$ is called a classifier. A classifier can be built by training it systematically using a set of training documents. (B.S. Harish. et. al., 2010)

A. Nearest Neighbor Classifier.

Nearest neighbors can be used to determine the class label of the test example. A nearest neighbor classifier represents each example as a document in a d-dimensional space , where d is the number of attributes(2 attributes i.e., number of documents and features). Given a test example, we compute its proximity to the rest of the documents in the training set, using one of the proximity measures. Nearest neighbor classification is part of a more general technique known as instance based learning, which uses specific training instance to make predictions without having to maintain an abstraction (or model) derived from data. Instance based learning algorithms require a proximity measure to determine the similarity or distance between instances and a classification function that returns the predicted class of a test instance based on its proximity to other instances.

The *nearest neighbor rule* is attractive because no prior knowledge of the distributions of the documents is required. These rules rely instead on the training set of documents with known query document provided by the user to make decisions on the class it belongs of unknown documents.

Nearest Neighbor Classifier works as follows:

Step 1: Documents are collected which are of different categories.

Step 2: In these documents the common words such as the, this, was etc which doesn't give any meaning are removed which helps in the classification process which is called preprocessing step.

Step 3: The relevant words which give the appropriate meaning is collected from all the classes and made it as dictionary.

Step 4: Vector representation is created using dictionary and classes which contains 0's and 1's. 0 represents the word is absent in the class which is there in dictionary and 1 represents that word is present.

Step 5: A query document is given by the user to analyze for which class it belongs.

Similarity between the query document and the vector representation of the documents is noted.

Step 6: The values obtained are sorted in ascending order and the corresponding index is taken of the first value.

Step 7: Query document's class is obtained by the ratio of the first value to the number of documents per class.

Step 8: That is the nearest class that query document belongs.

In Nearest neighbor, classifying a test sample can be quite expensive because we need to compute the proximity values individually between the test and training examples. The decision boundaries of nearest neighbor classifier also have high variability because they depend on the composition of training documents. Increasing the number of nearest neighbor may reduce such variability. Nearest neighbor classifier can produce wrong predictions unless the appropriate proximity measure and data preprocessing steps are taken. The nearest neighbor algorithm is easy to implement and executes quickly, but it can sometimes miss shorter routes which are easily noticed with human insight, due to its "greedy" nature.

B. K - Nearest Neighbor Classifier.

K-Nearest Neighbor is one of the most popular algorithms for text categorization. KNN is part of supervised learning that has been used in many applications in the field of data mining, statistical pattern recognition, image processing and many others. Despite the simplicity of the algorithm, it performs very well and is an important benchmark method.

The idea behind k-Nearest Neighbor algorithm is quite straightforward. To classify a new document, the system finds the k nearest neighbors among the training documents, and uses this to weight the new document. The intuition is that two instances of documents far apart in the training set defined by the appropriate distance function are less likely than two closely situated instances to belong to the same class.

The k-nearest-neighbor classifier is commonly based on the Euclidean distance between the training set and the specified query document. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbors.

The neighbors are taken from a set of objects for which the correct classification is known (preprocessed documents). This can be thought of as the training set for the algorithm, though no explicit training step is required. In order to identify neighbors, the objects are represented by position vectors in a multidimensional feature space.

In the traditional KNN algorithm, the value of k is fixed beforehand. If k is too large, big classes will overwhelm small ones. On the other hand, if k is too small, the advantage of KNN algorithm, which could make use of many experts, will not be exhibited. If $k = 1$, then the object is simply assigned to the class of its nearest neighbor. In practice, the value of k is usually optimized by many trials on the training and validation sets.

K-Nearest Neighbor is also known as lazy learner that is, when given a training tuple, KNN simply stores it and waits until it is given a test tuple. Only when is sees the test tuple does it perform generalization in order to classify the tuple based on its similarity to the stored training tuples. A major drawback to using this technique to classify a new vector to a class is that the classes with the more frequent examples tend to dominate the prediction of the new vector, as they tend to come up in the K nearest neighbors when the neighbors are computed due to their large number. One of the ways to overcome this problem is to take into account the distance of each K nearest neighbors with the new vector that is to be classified and predict the class of the new vector based on these distances.

C. Centroid Classifier.

Centroid Classifier has been shown to be a simple and yet effective method for text categorization. A centroid classifier is a modified version of *k*-NN classifier. Instead of comparing the test document with all training documents, Centroid calculates a centroid (a vector) for all training documents in each class and compares the test document with these centroids to find the most probable (similar) class.

Similarity measure used by the centroid-based scheme allows it to classify a new document based on how closely its behavior, as measured by the average similarity between the documents, matches the behavior of the documents belonging to different classes. This matching allows it to dynamically adjust for classes with different densities.

The similarity measure of the centroid-based scheme accounts for dependencies between the terms in the different classes. This feature is the reason why it consistently out-performs other classifiers, which cannot take these dependencies into account.

Centroid-based methods combine documents represented using a vector-space model, to find a representation for a “prototype” document that summarizes all the known documents for a given class, which is called the centroid. There are several ways to calculate this centroid during the training phase and several proposals for centroid-based methods are available in the literature. The average formula is one

such method where each centroid, is represented by the average of all the vectors for the positive training examples for a class.

Centroid Classifier works as follows:

Step 1: Documents are collected which are of different categories such as business, health and sports.

Step 2: In these documents the common words such as the, this, was etc which doesn't give any meaning are removed which helps in the classification process which is called preprocessing step.

Step 3: The relevant words which give the appropriate meaning is collected from all the classes and made it as dictionary.

Step 4: Vector representation is created using dictionary and classes which contains 0's and 1's. 0 represents the word is absent in the class which is there in dictionary and 1 represents that word is present.

Step 5: In that representation we divide into classes each having certain documents.

Step 6: In each class, we take half of the documents as Training Set and last half as the Testing set.

Step 7: Mean is taken in each column in the Training Set and represent that Mean in a table.

Step 8: A Query Document is taken randomly from Testing set.

Step 9: Similarity between Query document and the Mean table is calculated.

Sort the similarities in ascending order.

Step 10: Query document belongs to the class of the minimum similarity value, which is the first value in the sorted list.

Classifiers for compressed documents

A. Variants of LZWW approches

A practical adaptive compression algorithm based on LZSS is presented by Klein, S.T. and Shapira, D. 2000 which is especially constructed to solve the compressed pattern matching problem, i.e., pattern matching directly in a compressed text without decompressing.

Krishnavenyet, al 2008 improvement rate in microprocessor speed by far exceeds the improvement in DRAM memory. This method incorporates compression to main memory in order to fully utilize its resources and improve performance of data access.

Peter Fenwick 1995, they described a novel text compressor which combines Ziv-Lempel compression and arithmetic coding with a form of vector quantisation. The resulting compressor resembles an LZ-77 compressor, but with no explicit phrase lengths or coding for literals.

AnanthKamath et al 2009, Compression algorithms reduce the redundancy in data representation to decrease the storage required for that data. Lossless compression researchers have developed highly sophisticated approaches, such as Huffman encoding, arithmetic encoding, the Lempel-Ziv (LZ) family, Dynamic Markov Compression (DMC), Prediction by Partial Matching (PPM), and Burrows-Wheeler Transform (BWT) based algorithms.

Charles Bloom 1996, A new compression algorithm is presented which may either be considered an improvement to dictionary coding or an improvement to context coding. This algorithm works by reducing the set of available window position, for an LZ77 encoder to match from.

B. Huffman compression schemes

Justin Zobel and Alistair Moffat 1995 described the implementation of a data compression scheme as an integral and transparent layer within a full-text retrieval system. Using a semi-static word-based compression model, the space needed to store the text is under 30 per cent of the original requirement. The model is used in conjunction with canonical Huffman coding and together these two paradigms provide fast decompression.

C. Arithmetic compression schemes

John g. Cleary and Ian h. 1984. The recently developed technique of arithmetic coding, in conjunction with a Markov model of the source, is a powerful method of data compression in situations where a linear treatment is inappropriate. Adaptive coding allows the model to be constructed dynamically by both encoder and decoder during the course of the transmission, and has been shown to incur a smaller coding overhead than explicit transmission of the model's statistics. But there is a basic conflict between the desire to use high-order Markov models and the need to have them formed quickly as the initial part of the message is sent.

D. Index based compression schemes

N. Jesper Larsson 1996, A practical scheme for maintaining an index for a sliding window in optimal time and space, by use of a suffix tree, is presented. The index supports location of the longest matching substring in time proportional to the length of the match. The total time for build and update operations is proportional to the size of the input.

Compression Techniques

When we speak of a compression technique or compression algorithm, we actually discuss about two algorithms. These is the compression algorithm that takes as input *X* and generates a compressed representation *Xc*, and there is a reconstruction algorithm that operate on compressed representation *Xc* to generate *X* again. These operations are shown in following fig.

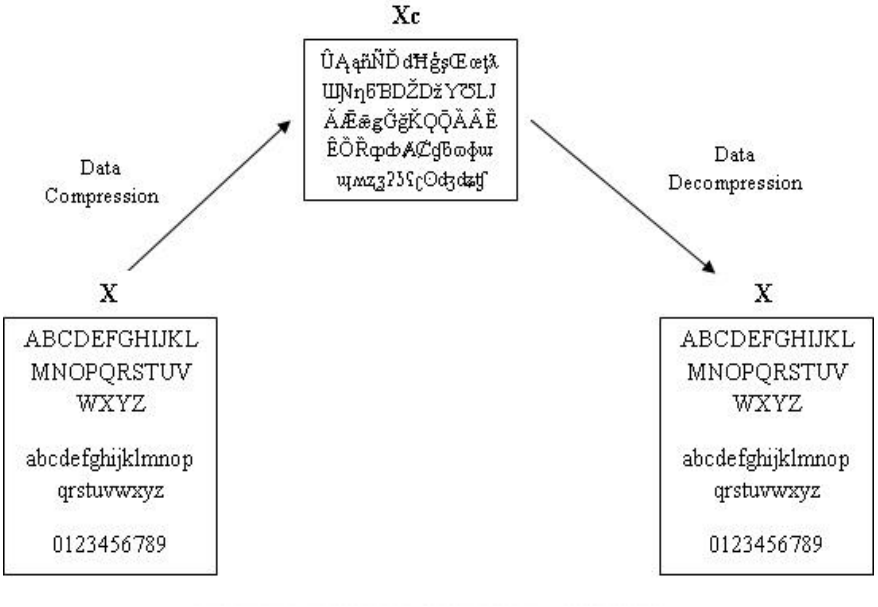

Figure 2.1: Block diagram of text compression and decompression system

Based on the requirements of the reconstruction algorithm, data compression techniques are divided into two broad classed: Lossless Data Compression and lossy Data Compression.

A. Lossless Data Compression

Lossless compression algorithms, as their name implies, involve no loss of information. If data have been losslessly compressed, the original data can be recovered exactly from the compressed data. Lossless data compression algorithm works by replacing repeated pattern in the document. Lossless data compression techniques are extensively used in the applications that cannot tolerate any difference between the original and the reconstructed data.

Text compression is an important area for lossless compression. It is very important the reconstruction is identical to the original text, as a very small change will result in statement will give different meaning.

Run Length Encoding.

Run Length Encoding (RLE) is a simple and popular data compression algorithm. It is based on the idea to replace a long sequence of the same symbol by a shorter sequence. Run-length encoding performs lossless data compression and is well suited to palette-based iconic images. It does not work well at all on continuous running text, because continuous text will not contain the same repeated sequence.

Consider the following example: AAAAABBBCDDDDEE

Using RLE, the above sentence can be represented as 5A3B1C4D2E.

Run Length Encoding, is best when the input data contain very long sequence of a unique symbol. Means for example AAAAAAAAAA, this can be effectively

represented as 10A. In worst case, means if the data contain no repeated symbol, then the RLE representation is twice of original data, means consider one example: ABCDEF will be represented as 1A1B1C1D1E1F.

Huffman Compression Technique:

The Huffman encoding algorithm is an optimal compression algorithm. This is a lossless compression technique, where the reconstructed data is identical to the original data. Huffman compression technique works on frequency of individual symbol. The idea behind the algorithm is that if you have some letters that are more frequent than others, it makes sense to use fewer bits to encode those letters than to encode the less frequent letters. The Huffman algorithm is a so-called "greedy" approach to solving this problem in the sense that at each step, the algorithm chooses the best available option.

It is a statistical compression method that converts characters into variable length bit strings. Most-frequently occurring characters are converted to shortest bit strings, least frequently occurring characters are converted to the longest bit strings. Bit assignment is determined by making a tree structure data first. Each node from top to bottom will contain the data from the most frequent to the least frequent data. The left child will be assigned by 0 and the right child will be assigned by 1. Compression takes two passes. The first pass analyzes a block of data and creates a tree model based on its contents. The second pass compresses the data via the model. The algorithm starts with building the tree in bottom-up fashion. First, every letter starts off as part of its own tree and the trees are ordered by the frequency of the letters in the original string. Then the two least-frequently used letters are combined into a single tree, and the frequency of that tree is set to be the combined frequency of the two trees that it links together.

Shannon Compression Techniques

The Shannon encoding algorithm is a good compression algorithm. This is a lossless compression technique, where the decompressed data is identical to the original data. Shannon compression technique works on frequency of individual data. The idea behind this compression algorithm is that if you have some letters that are more frequent than others, it makes sense to use fewer bits to encode those letters than to encode the less frequent letters. The major difference between the Shannon compression algorithm and Huffman compression algorithm is that the Shannon is top to bottom approach, whereas the Huffman compression is bottom up approach. In Shannon method, the construction of tree starts leaf to the root means in bottom up fashion.

LZW Approach

Lempel–Ziv–Welch (LZW) is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch. It was first published by Welch in 1984 as an improved implementation of the LZ78 algorithm published by Lempel and Ziv in 1978. The LZW algorithm organized around a translation table or string table, that maps input characters into the fixed length codes. The LZW string table has the

prefix property in that for every string in the table its prefix string is also in the table. That is if string ωK, compressed of some string ω and some single character K, is in the table. Then $\omega$is in the table. K is called the extension character on the prefix string ω. The LZW string table contains strings that have been encountered previously in the message being compressed. It consists of a running sample of strings in the message. LZW uses the "Greedy" parsing algorithm, where the input string is examined character serially in one pass, and the longest recognized input strings is parsed off each time. A recognized string is one that exists in the string table. The strings added to the string table are determined by this parsing. (Terry A. Welch, 1984)

Table 1: LZW CODE GENERATIONTABLE

SYMBOL	**CODE**		**SYMBOL**	**CODE**
A	1		N	14
B	2		O	15
C	3		P	16
D	4		Q	17
E	5		R	18
F	6		S	19
G	7		T	20
H	8		U	21
I	9		V	22
J	10		W	23
K	11		X	24
L	12		Y	25
M	13		Z	26

The Basic Encoding Algorithm

Based on the discussion above, encoding input consists of the following steps:

Step 1: Initialize dictionary to contain one entry for each byte. Initialize the encoded string with the first byte of the input stream.

Step 2: Read the next byte from the input stream.

Step 3: If the byte is an EOF go to step 6.

Step 4: If concatenating the byte to the encoded string produces a string that is in the dictionary:

Concatenate the byte to the encoded string.

Go to step 2.

Step 5: If concatenating the byte to the encoded string produces a string that is not in the dictionary:

Add the new sting to the dictionary.

Write the code for the encoded string to the output stream.

Set the encoded string equal to the new byte.

Go to step 2.

Step 6: Write out code for encoded string and exit.

Table 2: LZW Compression Table				
DATA INPUT : TOBEORNOTTOBEORTOBEORNOT#				
CURRENT SEQUENCE	**NEXT CHAR**	**OUTPUT**	**EXTENDED DICTIONARY**	
NULL	T	-	-	-
T	O	20	TO	27
O	B	15	OB	28
B	E	2	BE	29
E	O	5	EO	30
O	R	15	OR	31
R	N	18	RN	32
N	O	14	NO	33
O	T	15	OT	34
T	T	20	TT	35
TO	B	27	TOB	36
BE	O	29	BEO	37
OR	T	31	ORT	38
TOB	E	36	TOBE	39
EO	R	30	EOR	40
RN	O	32	RNO	41
OT	#	34		

The Basic Decoding Algorithm

Based on the discussion above, decoding input consists of the following steps:

Step 1: Initialize dictionary to contain one entry for each byte.

Step 2: Read the first code word from the input stream and write out the byte it encodes.

Step 3: Read the next code word from the input stream.

Step 4: If the code word is an EOF exit.

Step 5: Write out the string encoded by the code word.

Step 6: Concatenate the first character in the new code word to the string produced by the previous code word and add the resulting string to the dictionary.

Step 7: Go to step 3.

Table 3: LZW Decompression Table			
INPUT CODE	**OUTPUT SEQUENCE**	**DICTIONARY**	
20	T	TO	27
15	O	OB	28
2	B	BE	29
5	E	EO	30
15	O	OR	31
18	R	RN	32
14	N	NO	33

15	O	OT	34
20	T	TT	35
27	TO	TOB	36
29	BE	BEO	37
31	OR	ORT	38
36	TOB	TOBE	39
30	EO	EOR	40
32	RN	RNO	41
34	OT	TO	27
DECOMPRESSED DATA :TOBEORNOTTOBEORTOBEORNOT#			

B. Lossy Data Compression:

Lossy compression techniques involve some loss of information and data that have been compressed using lossy techniques which generally cannot be recovered or reconstructed exactly. In return for accepting this distortion in the reconstruction, we can generally obtain much higher compression ratios than is possible with lossless compression.

In many applications, this lack of exact reconstruction is not a problem. For example, when storing or transmitting of speech signals, the exact value of each sample of speech is not necessary. Depending on the quality required of the reconstructed speech, different lossy compression techniques are used.

Dataset

A. Google News Group dataset

A Usenet newsgroup *(http://en.wikipedia.org/wiki/Usenet_newsgroup)* is a repository created usually within the Usenet system, for messages posted from many users in different locations. The articles that the users post to Usenet are organized into topical categories called newsgroups, which are themselves logically organized into hierarchies of subjects. For example, sci.math and sci.physics are within the sci hierarchy for science. In most newsgroups, the majority of the articles are responses to some other article. Most of the responses can be categorized into major subject areas like: news, rec (recreation), soc (society), sci (science), comp (computers), and so forth (there are many more). There are currently over 100,000 Usenet newsgroups, but only 20,000 are active. There are seven original hierarchies of newsgroups.

B. Vehicles Wikipedia Dataset

Vehicles Wikipedia dataset is a small dataset used by Isa et al., (2008) to evaluate a prototype system used for the evaluation of classification performance. In Wikipedia dataset, each class contains 110 documents with a total of 440 documents in the entire dataset. Table 2.9 shows the categories in vehicles Wikipedia dataset and the number of text that each category contains. There is no fixed way to split the Wikipedia dataset into training and testing sets. The content of the documents are free running descriptions of the above said vehicles which are stored in .txt files.

C. 20 Class Dataset

The 20 mini newsgroups *(http:// kdd.ics.uci.edu/databases/20newsgroups.html)* dataset is a publically available dataset consisting collection of approximately 2,000 newsgroup documents, partitioned evenly across 20 different classes. It is a subset of 20 large news group dataset. In 20 mini newsgroup, there are 20 classes each with 100 documents which are randomly picked from the original dataset. In this dataset also, some of the newsgroups are very closely related to each other, while others are highly unrelated. We used only the subject and the body of each message for our experimentation.

Text Classification At Compressed Level

A. LZW Compression Algorithm

For compression, we learnt various compression techniques i.e., Run Length encoding (RLE), Huffman encoding, Shannon encoding and LZW compression techniques. RLE seems to be easy to implement than other technique, it fails in preserving semanticity of the document after decompression. So keeping this in mind we used LZW compression technique for compression the text documents. (Terry A. Welch, 1984) According to LZW compression scheme, a dictionary is created and classification is performed on the newly created dictionary. The block diagram of the proposed method is as shown in the figure. LZW compression algorithm replaces strings of characters with single codes. It does not do any analysis of the incoming text. Instead, it just adds every new string of characters it sees to a table of strings. Compression occurs when a single code is output instead of a string of characters. The first 256 codes (when using eight bit characters) are by default assigned to the standard character set. The remaining codes are assigned to strings as the algorithm proceeds. (Terry A. Welch, 1984). We compress the data using LZW compression technique. It consists of two stages namely training and testing. After compressing the text documents a new compressed dictionary will be created. A new testing document is also compressed using same LZW compression technique. New similarity is used as a feature. Similarity between uncompressed documents is calculated. Like uncompressed text, the similarity between same classes of compressed documents will be high, and similarity between different classes of compressed documents will be low. Using this similarity as a feature we applied nearest neighbour classifier algorithm.

B. Algorithm

Algorithm: Algorithm for classification of compressed text documents.

Start:

Step1 : Input collection of testing documents.

Step 2: Apply preprocessing and LZW compression technique.

Step 3: Develop a database of compressed text documents (dictionary).

Step 4: Input a testing document.

Step 5: Apply preprocessing and LZW compression technique.

Step 6: Find similarity between testing document to training documents.

Step 7: Apply nearest neighbor classifier method.
Step 8: Give a class label to the testing document.
Algorithm ends.

C. Experimentation Results

To evaluate any system we use some of the metrics. In pattern recognition system we usually make use of Precision, Recall and F- Measure as metrics. Precision and Recall are the two widely used metrics for assessing the correctness of a pattern recognition system. Where,

$$Precision = \frac{Correct}{Correct + False} \quad (1)$$

$$Recall = \frac{Correct}{Correct + Missed} \quad (2)$$

$$F - Measure = \frac{2\,PR}{P+R} \quad (3)$$

In order to check the robustness and to study the behavior of the method on large text data, we have conducted experiments on all datasets viz Google news group, vehicle Wikipedia dataset and 20 mini news group dataset. We analyzed the results obtained from different benchmark datasets. The method works well on large dataset, if they possess overlapping criteria. In 40% training and 60% testing trail we got highest f-measure for 20 mini news group dataset. It shows that the method works well on small dataset as well large dataset. In 60 % training and 40 % testing trail we got highest f-measure for dataset 2. The two sets of experiments show the efficiency of the method.

Table 4: Maximum F – Measure table

Datasets	**Max F-Measure**	
	Training / Testing Ratio	
	40 : 60	**60 : 40**
Google news group	77.6	77.3
Vehicles Wikipedia	78.2	78.5
20 mini news groups	80	78.2

Text Clustering At Compressed Level

Clustering has been used in the literature of text classification as an alternative representation scheme for text documents. Several approaches of clustering have been proposed. Clustering is assigning set of observations into subsets (called *clusters*) so that observations in the same cluster are similar in some sense. Clustering is a method of unsupervised learning.

A. Proximity measure

Proximity is the basic quality which identifies and characterizes groups of objects in various domains and contexts. When objects are compared to a set of chosen prototype examples, proximity can be used as a natural component to build a numerical representation. Pattern classes may be learned from such proximity representations by the traditional nearest neighbor rule, as well as by other alternative strategies. Similarity and dissimilarity as proximity measures are considered to study the relationship between compressed and uncompressed document. We calculated similarity between two uncompressed documents and also between compressed documents. Then we performed Eigen value analysis between these similarity measures. The results obtained from eigen value analysis shows that, similarity between uncompressed and compressed documents are same. For compression, we considered various compression techniques i.e., Run Length encoding (RLE), Huffman encoding, Shannon encoding and LZW compression techniques. RLE seems to be easy to implement than other technique, it fails in preserving semantic city of the document after decompression. So keeping this in mind we used LZW compression technique for compression the text documents. (Terry A. Welch, 1984). LZW compression algorithm replaces strings of characters with single codes. It does not do any analysis of the incoming text. Instead, it just adds every new string of characters it sees to a table of strings. This table of strings is referred as dictionary. Similarity as a proximity measure is applied on these newly constructed dictionary. After the proximity calculation, single linkage clustering algorithm is applied.

B. Algorithmic model

Algorithm: Algorithm for clustering of compressed text documents.

Start :

Step1 : Input collection of text documents from dataset.

Step 2: Apply preprocessing and LZW compression technique.

Step 3: Develop a database of compressed text documents (dictionary).

Step 4: Calculate similarity between compressed documents as a proximity measure.

Step 5: Apply clustering algorithms (K means method)

Algorithm ends.

C. Experimentation results

The details of the experiments conducted to represent the effectiveness of the proposed method on all the seven datasets viz., Google news group dataset, vehicle Wikipedia dataset and 20 mini news group dataset. During experimentation, we applied single linkage, complete linkage, average linkage, wards method and k-mean clustering algorithms on all dataset.

	Method	**F-Measure**
Google news group	**k Means**	51.28
Vehicles Wikipedia	**k Means**	55.61
20 mini news groups	**k Means**	58.47

Conclusion

A brief introduction to various compression techniques, a study of classification of uncompressed documents, classification and clustering of compressed documents are addressed. A method for classification of compressed document is deviced. In addition to this,considering similarity as a proximity measure LZW method is used to cluster the compressed document. To check the efficiency and the robustness of the proposed models, an extensive experiment is carried out on all the Three benchmark dataset, the details of the results are presented. The result evaluations of all the experiments are carried out by considering precision, recall and f-measure as metrics.

Though our work is appropriate in classifying and clustering compressed text documents the methods further extensions can be considered on the methods. One can work with combinations of compression techniques to increase the classification accuracy. A new representation scheme for a compressed text document can be developed which suits all types of data mining activity. Different proximity measures which increase clustering effecncy can be designed.New representation schemes which preserve semantics at compressed level can be found.

References

[1] Antonia Kyriakopoulou, and Theodore Kalamboukis. “Using clustering to enhance text classification”. In Proceedings of SIGIR 2007. pp.805-806

[2] Ananth Kamath, Ankit Kant, Aravind Srivatsa and Harisha J.A. Dynamic Decompression for Text Files. World Academy of Science, Engineering and Technology 59 2009

[3] Andrej Bratko, Gordon V. Cormack, Bogdan Filipi˘c, Thomas R. Lynam and Bla˘z Zupan “Spam Filtering Using Statistical Data Compression Models” Journal of Machine Learning Research 7 (2006) 2673-2698

[4] B.S. Harish, D.S. Guru, and S. Manjunath. “Representation and Classification of Text Documents: A Brief Review”. IJCA Special Issue on “Recent Trends in Image Processing and Pattern Recognition” RTIPPR, 2010.

[5] B.S. Harish, D.S. Guru, S. Manjunath, and R. Dinesh. “Cluster Based Symbolic Representation and Feature Selection for Text Classification”. ADMA, pp. 158-166, 2010.

[6] Charles Bloom, “A New Data Compression Algorithm” Data Compression Conference, 1996. DCC '96. Proceedings. PP 425

[7] David D. Lewis and Marc Ringuette, “A Comparison of two Learning Algorithms for text Categorization,” Symposium on Document Analysis and IR, ISRI, Las Vegas, April 1994.

[8] Deng Cai, Chiyuan Zhang, and Xiaofie He, “Unsupervised Feature Selection for Multi-Cluster Data”. 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'10), July 2010.

[9] D.S. Guru, B.S. Harish, and S. Manjunath, "Symbolic Representation of Text Documents" Proceedings of the International ACM Conference, Compute 2010, Bangalore, India.

[10] Dinesh, R., Harish, B. S., Guru, D.S., and Manjunath, S. 2009. Concept of Status Matrix in Text Classification. In the Proceedings of Indian International Conference on Artificial Intelligence, Tumkur, India, pp. 2071 – 2079.

[11] Krishnaveny Nadarajan and Zuriati Ahmad Zukarnain, "Analysis of String Matching Compression Algorithms" Journal of Computer Science 4 (3): 205-210, 2008 ISSN 1549-3636

[12] Klaus Berberich, Srikanta Bedathur, and Gerhard Weikum, "TunableWord-Level Index Compression for Versioned Corpora"

[13] Ko, Y. J., Park, J., and Seo, J. 2004. Improving text categorization using the importance of sentences. An International Journal Information Processing and Management, Vol. 40, pp. 65 – 79.

[14] Lan, M., Tan, C. L., Su. J., and Lu, Y.2009. Supervised and Traditional Term Weighting Methods for Automatic Text Categorization. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume: 31 (4), pp. 721 – 735.

[15] Liang, C. Y., Guo, L., Xia, Z. H., Nie, F. G., Li, X. X., Su, L., and Yang, Z. Y. 2006. Dictionary-based text categorization of chemical web pages. An International Journal Information Processing and Management, Vol 42, pp. 1017 – 1029.

[16] N. Jesper Larsson, "Extended Application of Suffix Trees to Data Compression" In Data Compression Conference 1996, PP 190- 1999

[17] Nguyen Cao Truong Hai, Kyung-Im Kim and Hyuk-Ro Park, "SVD-LDA: A Combined Model for Text Classification," Journal of Information Processing Systems, Vol.5, No.1, March 2009.

[18] Tan S., 2008. An improved centroid classifier for text categorization. Journal of Expert System with Applications, vol. 35, no. 2, pp. 279 – 285.

[19] William J.Teahan and David J.Harper, "Using compression-based language models for text categorization". School of Computer and Mathematical Sciences. Scotland, UK.