A Novel Text Classifier Using Supervised and Unsupervised Term Weighting Method

Chandrashekhar P. Bhamare, Dinesh D. Patil

Abstract— Measuring the similarity between documents is an important operation in the text processing field. Text categorization (also known as text classification, or topic spotting) is the task of automatically sorting a set of documents into categories from a predefine set [1]. TEXT categorization (TC) is the task of automatically classifying unlabeled natural language documents into a predefined set of semantic categories [2]. The term weighting methods assign appropriate weights to the terms to improve the performance of text categorization [1]. The traditional term weighting methods borrowed from information retrieval(IR), such as binary, term frequency (tf), tf:idf, and its various variants, belong to the unsupervised term weighting methods as the calculation of these weighting methods do not make use of the information on the category membership of training documents. Generally, the supervised term weighting methods adopt this known information in several ways. Therefore, the fundamental question arise here, "Does the difference between supervised and unsupervised term weighting methods have any relationship with different learning algorithms?", and if we consider normalized term frequency instead of term frequency along with relevant frequency the new method will be ntf.rf but will this new method is effective for text categorization? So we would like to answer these questions by implementing new supervised and unsupervised term weighing method (ntf.rf). The proposed TC method will use a number of experiments on two benchmark text collections 20NewsGroups and Reuters. Proposed system will use term weighting methods with preprocessing, so it will not requires labeled data and with the help of this, automatically results are improved in the form of precession, recall and accuracy. Proposed system improved the accuracy as compared to previous work and in that, supervised classifier is having more accuracy than unsupervised classifier.

Keywords — *Text Categorization, Information Retrieval, Term weighting.*

I. INTRODUCTION

Text Categorization (TC) is the task of automatically classifying unlabelled natural language documents into a predefined set of semantic categories. As the first and a vital step, text representation converts the content of a textual document into a compact format so that the document can be recognized and classified by a computer or a classifier. This problem has received a special and increased attention from researchers in the past few decades due to many reasons like the gigantic amount of digital and online documents that are easily accessible and the increased demand to organize and retrieve these documents efficiently. The fast expansion of the Internet globally also has increased the need for more text categorization systems. Efficient text categorization systems are beneficial for many applications, for example, information retrieval, classification of news stories, text filtering, categorization of incoming e-mail messages and memos, and classification of Web pages.

A. Existing System

A large number of machine learning, knowledge engineering, and probabilistic-based methods have been proposed for TC. The most popular methods include Bayesian probabilistic methods, regression models, example-based classification, decision trees, decision rules, Rocchio method, neural networks, support vector machines (SVM), and association rules mining. In the vector space model (VSM), the content of a document is represented as a vector in the term space. Terms can be at various levels, such as syllables, words, phrases, or any other complicated semantic and/or syntactic indexing units used to identify the contents of a text. Different terms have different importance in a text, thus an important indicator wi (usually between 0 and 1) represents how much the term ti contributes to the semantics of document d. The term weighting method is such an important step to improve the effectiveness of TC by assigning appropriate weights to terms. Although TC has been intensively studied for several decades, the term weighting methods for TC are usually borrowed from the traditional information retrieval (IR) field, for example, the simplest binary representation, the most famous tf:idf, and its various variants. Recently, the study of term weighting methods for TC has gained increasing attention. In contrast to IR, TC is a supervised learning task as it makes use of prior information on the membership of training documents in predefined categories. This known information is effective and has been widely used for the feature selection [1] and the construction of text classifier to improve the performance of the system. In this study, we group the term weighting methods into two categories according to whether the method involves this prior information, i.e., supervised term weighting method (if it uses this known membership

Manuscript received June, 2015.

***Chandrashekhar P. Bhamare** , Department of Computer Science and Engineering, SSGB College of Engineering and Technology., Bhusawal, India*

***Second Author name**, Department of Computer Science and Engineering, SSGB College of Engineering and Technology., Bhusawal, India*

ISSN: 2278 – 1323

information) and unsupervised term weighting method (if it does not use this information). Generally, the supervised term weighting methods adopt this known information in several ways. One approach is to weight terms by using feature selection metrics they are naturally thought to be of great help to assign appropriate weights to terms in TC. Another approach is based on statistical confidence intervals [4], which rely on the prior knowledge of the statistical information in the labeled training data. Nevertheless, there is another approach that combines the term weighting method with a text classifier [5]. Similar to the idea of using feature selection scores, the scores used by the text classifier aim to distinguish the documents. Since these supervised term weighting methods take the document distribution into consideration, they are naturally expected to be superior to the unsupervised (traditional) term weighting methods. However, not much work has been done on their comprehensive comparison with unsupervised term weighting methods. Although there are partial comparisons in [3] and [4], these supervised term weighting methods have been shown to have mixed results. .

There are various works done in text categorization till date, as text categorization can be done in both ways supervised and unsupervised there are various ways such as Racchio, decision trees, Naïve Bayes, SVM, etc. but according to results of all these methods it is proven that SVM is one the best method used for Text Categorization by using bag-of-words.

Decision Tree as one of the method uses information gain factor for text categorization which categorizes documents on the basis of combination of word occurrences for which it uses the tree based methods such as ID-3 and C4.5. but in this particular way as it is using combination of words most of words in English language are used in different ways according to the need of statement, hence in such cases it may give the proper results word-wise but fails to work in various situations also so we cannot state this as an effective one. These methods generate classifiers by inductive learning rule [4]. Decision tree induction classifier are biased towards frequent classes [7]

Naïve Bayes is again one of the method of text categorization in which user can get very reasonable performance in an attractive frameworks [5]. In this method document is considered as a binary feature vector which is the representation of whether term is present or not. And is also known as *multivariate Bernoulli naïve Bayes.* but in this method there are two problems first is its rough parameter estimation and calculations are done by taking all positive documents into consideration and second is in handling categories of rare terms or insufficient data where it cannot work well[6].

Olex again one of the novel method for text categorization specially in case of automatic induction of rule based text classifier. which needs documents to be classified into positive and negative literals. This rule allows prediction about belongingness about the terms in document, and also is an optimization problem. Non informative words are removed from the documents in order to increase the time efficiency; this uses chi^2 and information gain as one of the key method for calculating the efficiency and term goodness.

On one hand it is proved to be both effective and efficient and on other hand its local one-term-at-a-time greedy search strategy prevents it to cope with term interaction, as no two or more terms are calculated and evaluated at a time as a whole. Another problem is inconvenience with rule generation stems from the way how greedy heuristics works [7].

kNN which are the instance-based classifier do not relay on statistical distribution of training data, they cannot good positive examples. In this two different documents may be nearest neighbor even they are of different category and a vice-versa can also occur.

Racchio method also performs well after sorting the data into positive and negative categories but does not give that much efficiency this can also be proven by Table. 1 with the help of a labeling heuristic, called PNLH (Positive examples and Negative examples Labeling Heuristic) which is an extension of preliminary work in [8]. There is one more method introduced by Hisham Al- Mubaid and Syed A. umair, Lsquare using distributional clustering and learning logic. This is mainly focused on word feature and feature clustering, generates different separating and nested sets. This gives the results as good as the SVM with the differentiation of very few points [4].

Table I: Key Finding of Various Methods Used for Text Classification

Author and Year	Method used	Key Findings
Hisham Al-Mubaid and Syed A. Umair Year: 2003	SVM	This method is same as ours but they hav tested only few categories from the database [9].
Yiming yang, Jan O. Pedersen Year: 2005	Term weighting method, CHI^2	This method is quite expensive and i better for classifier such as neural network [2].
Padraig Cunningham and Sarah Jane Delany Year: 2007	K-NN	k-NN is useful if analysis of neighbour i important for classification [7].
Man Lan, Chew Lim Tan, Senior Member, IEEE, Jian Su, and Yue Lu, Member, IEEE Year: 2009	Tf.rf	According to him relevant frequenc approach is the best way. But this method can b applied only for two categories and two keyword from each category [1].
C. Deisy, M. Gowri, S. Baskar, S.M.A. Kalaiarasi, N. Ramaraj Year: 2010	Support Vector Machine	They have implemented modified invers document frequency and have implemented classifie for radial basis function [9].
Pascal Soucy, Guy W. Mineau 2014	Conf Weight	This can replace tf.idf method an performs well for both with and without featur selection [4].

B. Problems in Existing System

Even though there are various methods for text categorization which works differently according to the method parameters, but this is also true that these values changes or works according to the text hence it is necessary to check the properties of text. Some of which are listed below

1. Requires Labeled Database
2. Requires Cleaned dataset
3. Requires Linear reparability in dataset
4. Document vectors are sparse
5. High dimensional input space
6. Irrelevant features

ISSN: 2278 – 1323

C. Improvement in Existing System

1. Requires Labeled Database: Existing Systems requires labeling input for text categorization. This can be overcome by machine learning approach which will use training and testing phase, doesn't requires labeled dataset.
2. Requires Cleaned dataset: Existing Systems requires cleaned dataset. This can be overcome by incorporating preprocessing in proposed system, which consist filtering, tokenization, steaming and pruning.
3. Requires Linear reparability in dataset: most of categories of Ohsumed database are linearly separable hence classifier need to find it out first and so are many of the Reuters tasks also where the idea of existing is to find such linear separators.
4. Document vectors are sparse: for each document there are some entries which are not zero. Kivinen, Warmuth and Auer [8] gives both the theoretical and empirical evidences for the mistakes bound model that additive algorithms, which have similar inductive bias like proposed system are well suited for problems like dense concepts and sparse instances.
5. High dimensional input space: When we actually categorize the text we come across many features and proposed system will give over fitting protection which does not depends on the number of features. They have potential to handle large number of feature spaces.
6. Irrelevant features: To avoid the above stated problem one way is this and will do through feature selection. Through text categorization we get very few relevant features according to their information gain factor also many time even word occurring very few times gives the more relevant information. So the good classifier must combine many features and this aggressive selection may lead to loss of information and proposed system will give many parameters for feature selection, which may avoid this up to great extent.

Another two advantages of proposed system are one, it is based on simple ideas and provides clear intuition of what we are exactly learning from those examples. Second is it performs very well in practical applications and complex algorithm of feature extraction [17].

II. PROPOSED SYSTEM

A. Term Weighting Methods: A Brief Review

In text representation terms are words or phrases or any indexing term and each is represented by a value whose measure gives the importance of that term. All documents are categorized to get the features and those features acts as a keyword. The frequency of keyword that is the number of times the particular term occurs in the document is denoted by tf (term frequency), likewise there are various terms which gives frequency count of keywords which are given in table 2 as given below.

Table II: Term Frequency Factor [1]

Term frequency factor	Denoted by	Description
1.0	Binary	Binary weight =1 for terms present in a vector
Tf alone	Tf	Row term frequency(no of times term occurs in a document)
Log(1+tf)	Log tf	Logarithm of a term frequency
1-(1/(1+tf))	ITF	Inverse term frequency usually tf-1
ntf	Ntf	Normalized term frequency

In Table II four are commonly used term frequency factor in which binary term gives only the presence of term by 0 and 1 but it does not give importance of term hence we cannot use this in feature generation this is used in Naïve Bayes and Decision Trees also. Next is most popular term frequency representation which adopts raw term frequency however different variants of this also gives log(1+tf) which is nearly as same as log(tf) this is used to scale the unfavorably high term frequency in the documents[1], this is again reduced to certain extend by formula 1-(1/(1+tf)) known as inverse term frequency but this frequency factor is not as effective as an term frequency, no doubt it reduces the value of term frequency when it is high but it does not support to the new input document of classifier to categorize if it is not exactly the keyword but very close to the keyword. In that case we need to use the term frequency but if we do so again we need to minimize the unfavorable high value so the another solution that we are proposing in this is normalized term frequency factor denoted as ntf. Which is given by the equation (1) where i is the keyword that we want to search for and j is the document in which it occurs, while k_i is the maximum times occurring keyword in that document which may be or may not be same as i.

$$ntf = \frac{freq(i,j)}{\max_{k \in T} freq(k_i, j)} \qquad \textbf{(1)}$$

This gives the normalized term frequency of the document. There are various term weighting methods used for text categorization before this which are listed in table 4, if we go according to that we need to calculated few parameters such as information gain (ig), odds ratio (OR), chi square(x^2), relevant frequency (rf) and inverse document frequency (idf) this calculation is done by dividing the documents in to positive and negative categories and all the calculations are done on the term basis. Suppose there are six terms t1, t2, t3, t4, t5 and t6 as shown in Figure 2 Given one chosen positive category on data collection. Each column represents document

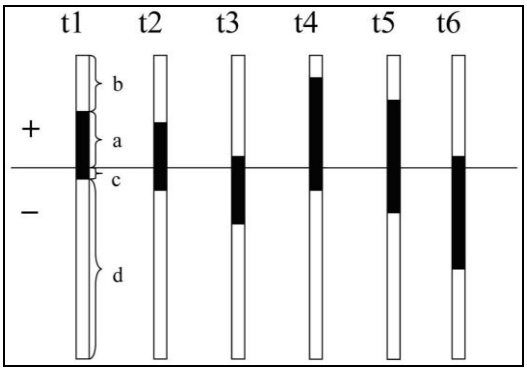

Fig 1: Examples of different distributions of documents that contain six terms in the whole collection [1]

Distribution in the corpus for each term and height is number of documents. The horizontal line divides these documents into two categories the positive (above) and negative(below) the heights of the column above and below the horizontal line denote the number of documents in the positive and negative categories, respectively. The height of

ISSN: 2278 – 1323

the shaded part is the number of documents that contains this term we use a, b, c and d to denote the number of different document as listed below

- a to denote the number of documents in positive category that contains this term
- b to denote the number of documents in positive category that do not contains this term
- c to denote the number of documents in negative category that contains this term
- d to denote the number of documents in negative category that do not contains this term[1]

Table III : Summary of Nine Different Term Weighting Methods[1]

Methods	Denoted by	Description
Unsupervised Term Weighting	Binary	1 for presence, 0 for absence
	Tf	Term frequency alone
	tf.idf	Classic tf and idf
Supervised Term Weighting	tf.rf	Term and relevant frequency
	Rf	relevant frequency alone
	$tf.x^2$	Tf and chi square
	tf.ig	Tf and information gain
	tf.logOR	Tf and logarithm of odds ratio
	ntf.rf	**Proposed method**

Now after getting normalized term frequency you need to find for relevant frequency which need two main parameters one that the number of documents in positive category that contains this term that is a, and the number of documents in negative category that contains this term that is c, based on these two parameters relevant frequency is calculated as given in equation (7)

$$rf = \log\left(2 + \frac{a}{c}\right) \quad (2)$$

By looking at above equation in worst case it may happen that there are no such documents in which the given term is not occurring at all in that case the denominator will become zero and will lead to divide by zero error hence in that case the another option is chose one instead of c, and equation(7) will be modified as equation(8) as given below

$$rf = \log\left(2 + \frac{a}{\max.(1,c)}\right) \quad (3)$$

As the term weighting methods described in table 4 we can go to the calculation of our proposed term weighting method that calculating ntf and rf and multiplying the both which will generate the new term weighting method. When we make term frequency as an normalized one it gives the frequency in range of 0 to 1 which we can call as an normalized one and restrict to unfavorable term count values that is why chosen normalized frequency rather that term frequency when we combine that with idf we get the weights of the terms which generate the vector space model.

B. Block Diagram of Proposed System

This input data is then filtered i.e. the special characters and special symbols such as @, <, >, $, ^, etc. are removed. Tokenization During this phase, all remaining text is parsed, lowercased and all punctuation removed. Stemming techniques are used to find out the root/stem of a word. Stemming converts words to their stems, which incorporates a great deal of language-dependent linguistic knowledge. Pruning also counts the number of times a particular term is occurring in the document which is also called as term frequency. In this way the documents will be preprocessed and in pruning it will measure the number of times of the term occurrences in the particular document as shown below. On the basis of this count, term frequency is calculated as the term frequency is nothing but the number of times occurrence of term in the document. Now the data is ready for further processing which is the generating vector space model for which we need term weighting method to be calculated first. Here as we are going to use classic tf.idf and its normalization form ntf.idf, will have to calculate their weight as well to get VSM and proper term weight.

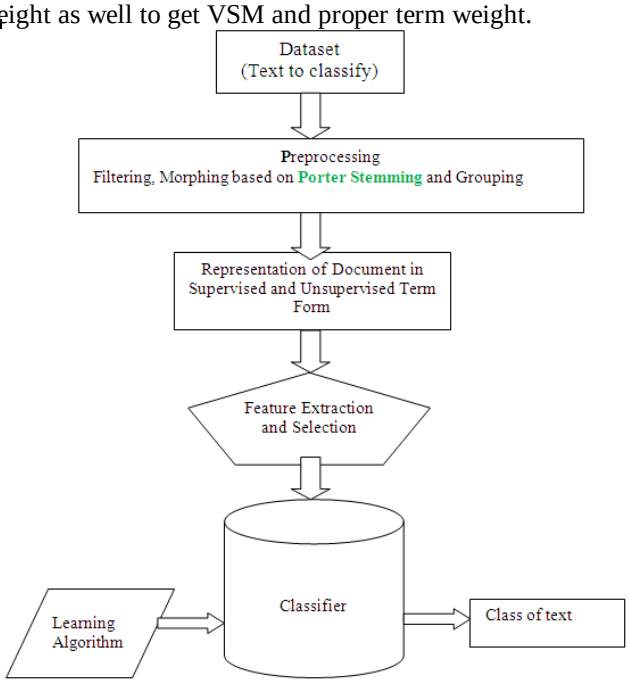

Fig 2. block diagram of proposed system[17].

C. Flowchart of Proposed System

The basic design of this work is to transform documents into a representation suitable for categorization and then categorize documents to the predefined categories based on the training weights and the flow is as shown in figure

ISSN: 2278 – 1323

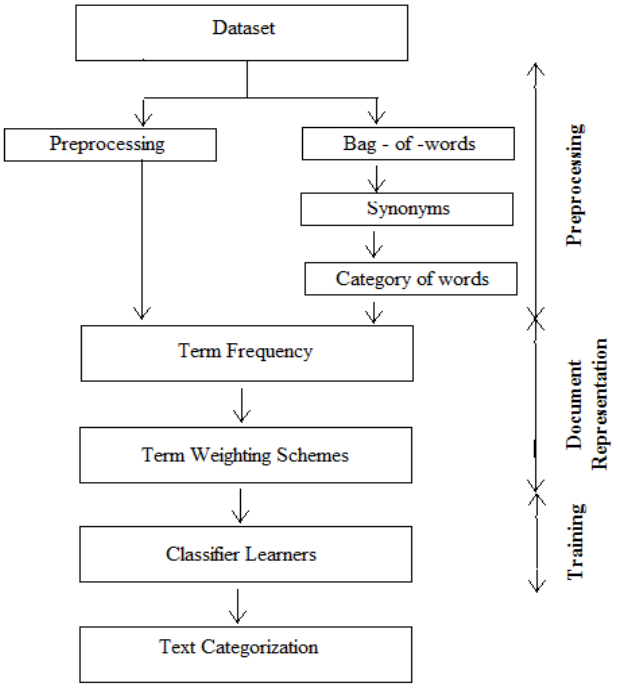

Fig 3. flowchart of proposed system.

D. Algorithm of Proposed System

Proposed algorithm for text categorization:

Input

Set of documents D = {D_1, D_2, . . . , Dm}
// Set of documents to be classified
Fixed set of categories C = {C_1, C_2, . . . , Cn}
//set of categories in which documents should be classified

Output: Determine d_i Ɛ C // where d_i is i^{th} document from Set of documents D.

Steps:

1. Wordlist {m, 1}← each word in the document
2. // read each word in the document to be classified and set each word in different row which will make a list
3. Wordlist {m, 2} ← Filtering (Wordlist)
   // above generated list will be given to filtering phase which will remove special characters from the list and will modify it in second column
4. Wordlist {m, 3} ← Tokenization (Wordlist)
   // list from second column will be accepted as an input and all stop words will be removed from the list. Modified list will be in third column
5. Wordlist {m, 4} ← Stemming (Wordlist)
   // list from third column will be accepted as an input and all words will be converted in their proper verb form. Modified list will be in fourth column
6. Wordlist ← Pruning (Wordlist)
   // list from fourth column will be accepted as an input and all words will be counted by their number of occurrences in the list. Modified list will be given to another uitable.
7. *ntf* ← (*tf*, k, t)
   // by accepting the output of pruning and performing operation as given in equation (1) we will get normalized term frequency.
8. *idf* ← log(n/N) // inverse document frequency will be calculated
9. VSM ← *ntf* * *idf* // vector space model and normalized vector space model is calculated.
10. Crate Classifier // depending on the methodology getting used classifier will be created
11. Use Classifier C (Y/X) // by passing appropriate parameters to the above created classifier document get categorized
12. Select the best classification result // finally we will get the classification results

***E Method*:** Proposed system will work in three major steps

Step 1: Preprocessing

- Filtering: Filters the input dataset.
- Tokenization: It removes lowercased and all punctuation.
- Stemming: IT converts words to their root stems.
- Pruning: It counts the number of times a particular term is occurring in the document.

Step 2: Training the Classifier

- Calculating tf, idf, VSM: calculate the weights as well to get VSM and proper term weight.
- Keyword List: It contains all terms occurring in the documents of respective category.
- Training the classifier: It trains in supervised and unsupervised manner.

Step 3: Testing the classifier: It test the accuracy of classifier by providing unseen dataset[17].

III. Working of Proposed System

A. Superviesd Classifier

Once the classifier is trained we can give any number of documents to get classified. Before applying any document for categorization SVM needs to be get trained first, in training it trains the classifier for classification. This training is not needed in k-NN as this classifier is based on neighbour approach it does not need to be getting trained. SVM train function creates accepts the data in the above given format and creates a classifier as follows:

Group = svmclassify(*SVMStruct*, *Sample*)

Classifies each row of the data in Sample using the information in a support vector machine classifier structure SVMStruct, created using the svmtrain function. Sample must have the same number of columns as the data used to train the classifier in svmtrain. Group indicates the group to which each row of Sample has been assigned.

Group = svmclassify(*SVMStruct*, *Sample*, 'Showplot', *ShowplotValue*)

Controls the plotting of the sample data in the figure created using the Showplot property with the svmtrain function.

ISSN: 2278 – 1323

B. Unsupervised Classifier

To classify the document with help of k-NN classifier we need to give input in specific format which will create the classifier according to our problem statement and requirements. The basic syntax for k-NN classifier is:

Class = knnclassify(*Sample*, *Training*, *Group*).

Classifies the rows of the data matrix *Sample* into groups, based on the grouping of the rows of *Training*. *Sample* and *Training* must be matrices with the same number of columns. *Group* is a vector whose distinct values define the grouping of the rows in *Training*. Each row of *Training* belongs to the group whose value is the corresponding entry of *Group*. knnclassify assigns each row of Sample to the group for the closest row of *Training*. *Group* can be a numeric vector, a string array, or a cell array of strings. *Training* and *Group* must have the same number of rows. knnclassify treats NaNs or empty strings in *Group* as missing values, and ignores the corresponding rows of *Training*. *Class* indicates which group each row of *Sample* has been assigned to, and is of the same type as *Group*.

Class = knnclassify(*Sample*, *Training*, *Group*, *k*)

Enables you to specify k, the number of nearest neighbors used in the classification. Default is 1.

Class = knnclassify(*Sample*, *Training*, *Group*, *k*, *distance*)

Enables you to specify the distance metric. Choices for *distance* are:

1. 'Euclidean' — Euclidean distance (default)
2. 'cityblock' — Sum of absolute differences
3. 'cosine' — One minus the cosine of the included angle between points (treated as vectors)
4. 'correlation' — One minus the sample correlation between points (treated as sequences of values)
5. 'hamming' — Percentage of bits that differ (suitable only for binary data)

Class = knnclassify(*Sample*, *Training*, *Group*, *k*, *distance*, *rule*)

enables you to specify the rule used to decide how to classify the sample. Choices for *rule* are:

1. 'nearest' — Majority rule with nearest point tie-break (default)
2. 'random' — Majority rule with random point tie-break
3. 'consensus' — Consensus rule

The default behavior is to use majority rule. That is, a sample point is assigned to the class the majority of the k nearest neighbors is from. Use 'consensus' to require a consensus, as opposed to majority rule. When using the 'consensus' option, points where not all of the k nearest neighbors are from the same class are not assigned to one of the classes. Instead the output Class for these points is NaN for numerical groups or '' for string named groups. When classifying to more than two groups or when using an even value for k, it might be necessary to break a tie in the number of nearest neighbors. Options are 'random', which selects a random tiebreaker, and 'nearest', which uses the nearest neighbor among the tied groups to break the tie. The default behavior is majority rule, with nearest tie-break.

IV. EXPERIMENTAL ANALYSIS

We have 20NewsGroup dataset to evaluate supervised and unsupervised classification on the same number of training examples. Following table shows the accuracy, precision and recall of both the classifiers.

Table IV. Comparison of accuracy, precision and recall of Supervised and unsupervised

Category	Supervised classifier			Unsupervised Cla	
	Accuracy	Precision	Recall	Accuracy	Precision
'comp.graphics'	0.663636	1	0.63	0.990909	0.990099
'alt.atheism'	0.818182	0.9	0.9	1	1
comp.os.ms.windows.misc'	0.090909	0	0	0.090909	0
comp.sys.ibm.pc.hardware'	0.090909	0	0	0.481818	1
comp.sys.mac.hardware'	0.772727	0	0.75	0.3	1
comp.windows.x'	0.890909	1	0.88	0.827273	0.965517
misc.forsale'	0.090909	0	0	0.090909	0
rec.autos'	0.981818	0.980392	1	0.981818	0.980392
rec.motorcycles'	0.909091	0.909091	1	0.827273	0.90099
rec.sport.baseball'	0.909091	0.909091	1	0.809091	0.89899
rec.sport.hockey'	0.090909	0	0	0.163636	1
sci.crypt'	0.909091	0.909091	1	0.909091	0.909091
sci.electronics'	0.090909	0	0	0.090909	0
sci.med'	0.090909	0	0	0.090909	0
'sci.space'	0.927273	1	0.92	0.863636	1
soc.religion.christian'	0.909091	0.909091	1	0.972727	0.970874
talk.politics.guns'	0.8	1	0.78	0.090909	0
talk.politics.mideast'	0.818182	1	0.8	0.6	1
talk.politics.misc'	0.827273	0.90099	0.91	0.981818	0.980392
talk.religion.misc'	0.090909		0	0.090909	0

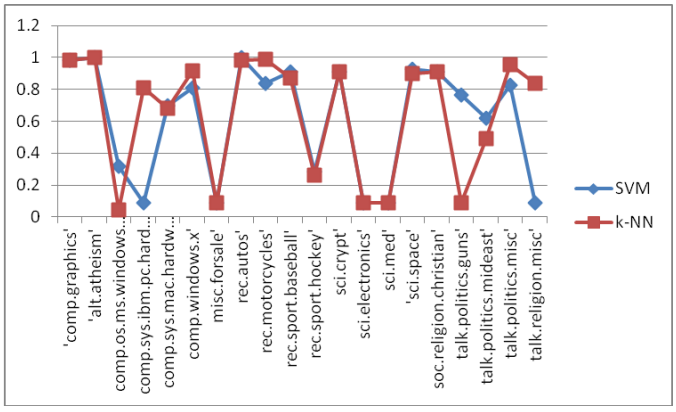

Figure : Accuracy of Supervised and Unsupervised Classifiers

From above graph, it is clear that for most of the documents supervised classifier giving better accuracy than unsupervised

Table V. Comparison of results for various authors

Category	Previous Work	Proposed Method
'comp.graphics'	0.86	0.99
'alt.atheism'	0.78	1
comp.os.ms.windows.misc'	0.88	1
comp.sys.ibm.pc.hardware'	0.87	1
comp.sys.mac.hardware'	0.84	0.87
comp.windows.x'	0.85	1
misc.forsale'	0.85	0.86
rec.autos'	0.93	1
rec.motorcycles'	0.93	1
rec.sport.baseball'	0.98	1
rec.sport.hockey'	0.99	1
sci.crypt'	0.96	1
sci.electronics'	0.77	0.97
sci.med'	0.93	0.99
'sci.space'	0.96	1
soc.religion.christian'	0.9	1

ISSN: 2278 – 1323

talk.politics.guns'	0.97	1
talk.politics.mideast'	0.9	1
talk.politics.misc'	0.9	0.8
talk.religion.misc'	0.85	0.8

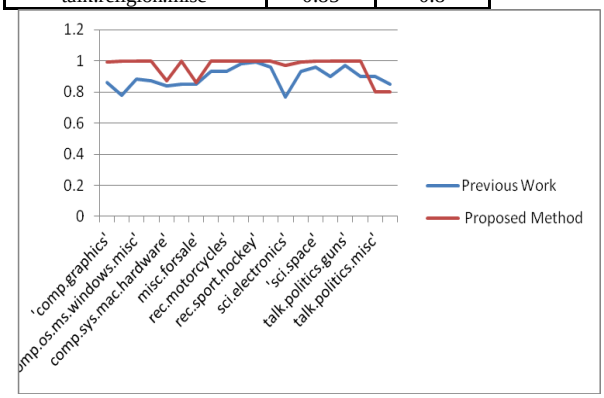

Figure: Graph for Accuracy of Proposed work and Previous work

V. CONCLUSION AND FUTURE WORK

The performance of term weighting methods specially unsupervised term weighting methods has close relationship with learning algorithms and data corpora and we can also state that ntf.rf and ntf.idf will give better performance than tf.rf and tf has no relationship with algorithm and data corpora. Proposed system will perform very well in all cases. There is no doubt that tf.rf performs well is proved by all evidences but this is not well suited when there are large number of categories and more number of keywords hence in that case ntf.rf and even more that that ntf.idf is well doing. And a good text categorization can be performed in both supervises and unsupervised machine learning. Proposed system will use term weighting methods with preprocessing, so it will not requires labeled data and with the help of this, automatically results are improved in the form of precession, recall and accuracy. Proposed system improved the accuracy as compared to previous work and in that, supervised classifier is having more accuracy than unsupervised classifier.

In future we are planning to test the data for all other term weighting methods as well, for supervised as well as unsupervised. Along with this we want to test the data for different natural language datasets, which are globally available.

REFERENCES

[1]Hisham Al-Mubaid and Syed A. Umair, "A New Text Categorization Technique Using Distributional Clustering and Learning Logic", *IEEE Transactions on Knowledge And Data Engineering*, Vol. 18, NO. 9, September 2006

[2] Yiming Yang. Jan o. Pederson, "A comparative study on feature selection in text categorization", *Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997 (ICML '97). - ISBN 1-55860-486-3, S. 412—420

[3] P´adraig Cunningham1 and Sarah Jane Delany2, "k-Nearest Neighbour Classifiers", *Technical Report* UCD-CSI-2007-4 March 27, 2007

[4] Man Lan, Chew Lim Tan, Senior Member, IEEE, Jian Su, and Yue Lu, "Supervised and Traditional Term Weighting Methods for Automatic Text Categorization", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, Vol. 31, No. 4, April 2009, ISBN 987-6-7803-9183-3

[5] C. Deisy, M. Gowri, S. Baskar3, S.M.A. Kalaiarasi, N. Ramraj, "A Novel Term Weighting Scheme Midf For Text Categorization*", Journal of Engineering Science and Technology* Vol. 5, No. 1 (2010) 94 – 107 © School of Engineering, Taylor's University College

[6] Pascal Soucy, Gay Mineau, "Beyond TFIDF Weighting for Text Categorization in the Vector Space Model", *At the time these experiments were conducted, the LYRL2004 split was not yet released*

[7] Franca Debole and FabrizioSebastiani Institute of Science and Technologies "Supervised Term Weighting for Automated Text Categorization**",** 18th Proceedings of Symposium on Applied Computing, Melbourne, US, SAC-03, ACM, 2003, pp. 784–788.

[8] C. Deisy, M. Gowri, S. Baskar3, S.M.A. Kalaiarasi, N. Ramraj, "A Novel Term Weighting Scheme Midf For Text Categorization*", Journal of Engineering Science and Technology* Vol. 5, No. 1 (2010) 94 – 107 © School of Engineering, Taylor's University College

[9] SauravSahay, "Support Vector Machines and Document Classification", *Journal of Advances in Information Technology*ISSN 1798-2340, Volume 1, Number 1, February 2010

[10] Karen Spärck Jones, "A statistical interpretation of term specificity and its application in retrieval", *Journal of Technical Documentation* Volume 60 Number 5 2004 pp. 493-502 Copyright © MCB University Press ISSN 0022-0418

[11] Georges Siolas, Florence d'Alch&Buc "Support Vector Machines based on a Semantic Kernel for Text Categorization", 2000 IEEE /0-7695-0619-4/00

[12]V. Srividhya, R. Anitha "Evaluating Preprocessing Techniques in Text Categorization", *International Journal of Computer Science and Application* Issue 2010

[13] By Marti A. Hearst University of California, Berkeley "SVM" *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 18-28, July/August, 1998.

[14] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung HyonMyaeng Some "Effective Techniques for Naive Bayes Text Classification"*IEEE Transactions on Knowledge and Data Engineering*, Vol. 18, No. 11, November 2006

[15] M. Ikonomakis, S. Kotsiantis, V. Tampakas, "Text Classification Using Machine Learning Techniques", WSEAS Transactions on Computers, Issue 8, Volume 4, August 2005, pp. 966-974

[16]V. Srividhya, R. Anitha,"Evaluating Preprocessing Techniques in Text Categorization, "*International Journal of Computer Science and Application*, Issue 2, 2010

[17]Chandrashekhar P. Bhamare, Dinesh D. Patil, "Review of Various Text Categorization Methods", *IOSR Journal of Computer Engineering (IOSR-JCE),* e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 3, Ver. 1 (May – Jun. 2015), PP 13-19.

ISSN: 2278 – 1323