Modified TF-IDF Term Weighting Strategies for Text Categorization

Conference Paper · October 2018

DOI: 10.1109/INDICON.2017.8487593

1 author:

Jajati Keshari Sahoo
Birla Institute of Technology and Science Pilani
52 PUBLICATIONS **101** CITATIONS

SEE PROFILE

Modified TF-IDF Term Weighting Strategies for Text Categorization

Rajendra Kumar Roul
Department of Computer Science
BITS Pilani-K.K.Birla Goa Campus
Zuarinagar, Goa-403726, India
Email: rkroul@goa.bits-pilani.ac.in

Jajati Keshari Sahoo
Department of Mathematics
BITS Pilani-K.K.Birla Goa Campus
Zuarinagar, Goa-403726, India
Email: jksahoo@goa.bits-pilani.ac.in

Kushagr Arora
Department of Computer Science
BITS Pilani-K.K.Birla Goa Campus
Zuarinagar, Goa-403726, India
Email: kushagrarora786@gmail.com

***Abstract*—Text mining is a well-known technique in the domain of information retrieval which derives high quality of information from the text. To develop strategies for such text processing, an appropriate domain representation is required. Vectorized Term Frequency and Inverse Document Frequency (TF-IDF) representation of documents is one of the current strategies in use. Traditional TF-IDF uses term frequencies and document frequencies to generate a weighted term which is used for document representation. This method works sufficiently well, however, it is quite simplistic and overlooks many details that should ideally be relevant while processing the text such as document length, frequency distribution etc. To handle those shortcomings, this paper proposes four vector representation of documents which is the modified version of the traditional TF-IDF. In order to check the performance of the proposed techniques, different state-of-the-art classifiers are used to classify a corpus of documents. Experimental results on different benchmark datasets show that the performances of different classifiers using the proposed techniques are better than the traditional TF-IDF.**

***Keywords*—Classification, Cosine-similarity, Inverse Document Frequency, Term Frequency, Vector Space Model**

I. Introduction

The base of the language model is how to formalize or represent the documents, such as vector space model (VSM), topic model, latent semantic indexing model etc. The formalization of documents also affects areas where natural language processing is applicable such as text mining, information retrieval, image processing etc. In order to formalize the documents, it is necessary to convert them into vector forms. Once the documents are converted into vectors, then any type of mathematical operation can take place to utilize them for different text mining purposes such as clustering, classification, ranking, summarization etc. Document constitute terms and to make the vector representation more accurate, term weighting plays a big role. TF-IDF is one of the popular and most widely used term weighting technique in today's IR system.

TF-IDF [1] constitute two terms: Term Frequency (TF) and Inverse Document Frequency (IDF). 'TF' tells us that if a term 't' is very frequent in a document 'd' then t is very important for d. Similarly, 'IDF' measures the rarity of 't' with respect to the entire collection i.e. terms that appear in many documents are less informative. The number of documents which contain the term t in the entire collection gives its document frequency or global frequency. For example, stop-words such as 'a', 'an', 'the' etc. are less informative because they are common to all the documents in a corpus. A term which is very rare in the entire corpus but frequent in a document is considered as highly important. TF-IDF has been found to work well but has many limitations which are discussed in section 2.3. Many researchers have proposed different modified version of TF-IDF [2][3][4].

The next important topic in text mining is classification of text document which is a powerful machine learning technique that categorizes an unseen document into its respective predefined class. Two basic classifications of web pages are there: subject-based classification and genre-based classification [5]. In subject-based classification, web pages are classified based on their subject or content. Topic hierarchies of web pages are built by this approach. Web pages in genre-based classification are classified into genre or functional related factors. For example, some web pages genres are "multimedia", "home page", "online transaction" and "news headlines". These classification help users to find their immediate interest. There are different classification techniques that exist in real and can be divided into two broad categories: *eager learner* and *lazy learner*. According to *eager learner*, the learner built a classification model when the training dataset is given before it receives the test dataset. It can be thought as if the learning model is ready and eager to classify the new test dataset. Examples of this are decision tree, Bayesian network, support vector machine, rule and association based classifier. But in *lazy learner*, the things are different. Here, instead of building a classification model, it simply stores the training dataset and when it sees the test dataset, it does the classification based on the similarity to the stored training dataset, hence consumes extra memories. Examples include k-nearest neighbors (k-NN), case-based reasoning etc. Many research works have been done in the field of web document classification [6][7][8][9].

To handle the limitations of existing TF-IDF, in this paper four modified TF-IDF techniques are proposed which improves the text classification process. The following points briefly discuss them.

i. *TF-IDF based on inter-class dispersion*: Inter-class dispersion defined the contribution of a term in a class

978-1-5386-4318-1/17/$31.00 ©2017 IEEE

which helps the classifier to take a correct decision during classification of a document.

ii. *TF-IDF based on modified inverse document frequency*: Consider those words which are very frequent in documents and also in the entire collection (except stop-words). This helps to increase the discriminative power of traditional IDF.

iii. *TF-IDF based on class frequency*: The class frequency helps to identify whether a term is relevant to a particular class or not. It gives weight to the terms in class characteristics. In text classification, adding weights to the terms based on their class frequency will improve the classification accuracy.

iv. *TF-IDF based on normalized length*: By adding the length normalizing factor to TF-IDF will increase the importance of TF and reduces the weights of the terms which are less frequent but have relatively higher TF-weighting in the document.

Experimental results on three benchmark datasets show that the proposed four modified TF-IDF approaches are better than the traditional TF-IDF technique. The remaining parts of the paper consist the following: Section 2 discusses the basic preliminaries of some concepts. In Section 3, we have discussed the four modified TF-IDF techniques. Section 4 discusses the empirical analysis of the proposed work and Section 5 concludes the work.

II. Basic Preliminaries

A. Vector Space Model

An algebraic model called Vector Space Model (VSM) [10], aimed to facilitate information retrieval by modeling the documents as a set of terms. VSM is transformed from a full text version to a vector which has various pattern of occurrence. It represents document d as a vector of terms, d = $(t_{1j}, t_{2j}, t_{3j}, t_{4j}, ..., t_{nj})$, where t_{ij} is the weight of i^{th} term in j^{th} document.

B. Term Frequency and Inverse Document Frequency

$TF\text{-}IDF$ is a technique which measures the importance of a term t in a document d with respect to the entire corpus P and is calculated using equation 1.

$$TF\text{-}IDF_{t,d} = TF_{t,d} * IDF_t \tag{1}$$

where,

$$TF_{t,d} = \frac{no.\ of\ occurance\ of\ t\ \in\ d}{total\ length\ of\ d}$$

$$IDF_t = log_{10}\Big(\frac{no.\ of\ documents\ \in\ P}{no.\ of\ documents\ of\ P\ contain\ the\ term\ t}\Big)$$

C. Limitations of traditional TF-IDF

TF-IDF is commonly used in text mining, natural language processing and information retrieval. In the recent study, there are other models such as VSM, topic models, word embedding and latent semantic indexing are used for better indexing. In spite of many advantages of TF-IDF, it has many drawbacks which can not be ignored and some of them are highlighted below:

- It calculates the term weight based on the term frequency. It does not care the position of a term in the text, its semantics and co-occurrence with other terms in a document.
- Though the document similarity is computed directly in the vector space, hence it may be slow for the large corpus.
- It has no capability for inter-class dispersion and hence misleads the classifier during the classification.
- It may reduce the discriminative power of IDF.
- As TF-IDF is an unsupervised feature selection technique, hence it does not say anything whether a term is relevant to a particular class or not. It is only bounded by the document.
- TF-IDF assumes that counting of different terms provides independent evidence of similarity which is not always true.

The proposed approach improves some of the above limitations of the traditional TF-IDF and those are discussed in section 3.3 in details.

D. Cohesion

Cohesion (compactness, tightness), determines the closeness of terms in a cluster by computing the distance of each term t from the centroid of the cluster. If the term t is highly cohesive i.e. the distance between the term t and the centroid of the cluster is very small compared to other terms in the same cluster then it defines the cluster very well. If the term t is present at the border of the cluster (far away from the centroid) then t is poorly cohesive to the cluster. Euclidean distance can be used to measure the distance of a term t from the centroid of the cluster, c' as follows: If $\vec{t} = (t_1, t_2)$ and $\vec{c'} = (c'_1, c'_2)$, then $||\vec{c'} - \vec{t}|| = \sqrt{(c'_1 - t_1)^2 + (c'_2 - t_2)^2}$, where $\vec{t}$ and $\vec{c'}$ are term and centroid vectors, respectively.

E. Fuzzy C-Means

Fuzzy C-Means (FCM) clustering algorithm [11] tries to distribute a finite collection of n documents into c clusters. It returns a list of c cluster centroids along with a matrix $[U]$ which shows the degree of membership of each document to different clusters. It aims to minimize the following function:

$$T_m = \sum_{i=1}^{n}\sum_{j=1}^{c} v_{ij}^m ||dist_{ij}||^2$$

where, distance $dist_{ij} = x_i - c_j$,
m is the fuzzy coefficient which is generally set to 2,
c_j is the centroid (vector) of cluster j,
x_i is the i^{th} document,
v_{ij} $\in$ range $[0, 1]$ is the degree of membership of x_i with respect to c_j subject to the following conditions:

$$\sum_{j=1}^{c} v_{ji} = 1, \quad i = 1, ..., n,\ and\ 0 < \sum_{i=1}^{n} v_{ij} < n, \quad j = 1, ..., c$$

Initially the number of clusters to be formed, fuzziness coefficient (m) and the accuracy margin (ε) will be supplied as the input. A membership matrix is created at the beginning by assigning randomly the membership values to each document for all the clusters i.e. $U_0 = [v_{ij}]$ such that it follows the

conditions of the above formulas. One can iteratively find the values of c_j and v_{ij} updated with each iteration by using the following formulas.

$$c_j = \frac{\sum_{i=1}^{n}(v_{ij})^m.x_i}{\sum_{i=1}^{n}(v_{ij})^m},\ and\ v_{ij} = \frac{1}{\sum_{k=1}^{c}(\frac{||d_{ij}||}{||x_i - c_k||})^{\frac{2}{m-1}}}$$

The membership matrix U is updated at each iteration and the process will stop when $||U^{i+1} - U^i|| < \varepsilon$. Finally, based on the membership values of each document, one can decide a document will belongs to which of the c clusters.

III. Proposed Methodology

Text classification requires the string of characters per document to be transformed into a representation which is comprehensible and fit to be fed to a classifier (known as the vector forms of the documents). The four different modified TF-IDF techniques which convert the documents into vectors are discussed in this section.

A. Pre-Processing the corpus

Given a corpus P having n classes, first, all the documents of each class are pre-processed using a pre-processing algorithm and then the documents of all n classes are collected together which make the dimension of P as $r \times p$, where r and p are total numbers of terms and documents in the corpus. In order to convert the documents into vector, the TF-IDF weight of each term is calculated in the document space.

B. Modified TF-IDF techniques

In addition to the TF-IDF form, the documents are represented in four different vector forms which are discussed below where $Weight(i,j)$ represents the weight of the i^{th} term in j^{th} document.

1. ***TF-IDF based on inter-class dispersion (W1):***
Equation 2 shows the modified TF-IDF based on inter-class dispersion.

$$Weight(i,j) = W_{1_{ij}} = TF(i,j) \star IDF(i) \star D(i) \tag{2}$$

where $D(i)$ is the inter-class dispersion coefficient of the term i and it is calculated as

$$D(i) = \frac{1}{n}\sum_{i=0}^{n}(F(t,i) - avg(F(t,i))^2$$

Here, n is the number of classes and $F(t,i)$ is the number of documents having the term t and belongs to the class to which the term i belongs. Also,

$$avg(F(t,i)) = \frac{1}{n}\sum_{i=0}^{n}F(t,i)$$

Along with the advantages of traditional TF-IDF, equation 2 also checks how well a term contributes to the classification process. If a term is uniformly distributed amongst the classes then the inter-class dispersion, D will be low. Hence, the weight contributed by the term will be low. But if a term has a high variance which means that it is a good feature to use for the classification. Accordingly, the value of D will be high for that term.

2. ***TF-IDF based on modified inverse document frequency (W2):***
The TF-IDF weight is now modified using equation 4.

$$\text{Weight}(i,j) = W_{2_{ij}} = TF(i,j) \star \text{Modified}_{IDF}(i) \tag{3}$$

where,

$$\text{Modified}_{IDF}(i) = \log_{10}\left(\frac{\text{no. of documents in P} + 1}{\text{document frequency of term } i}\right)$$

If a term occurred in every document then traditional IDF will give it a weight 0 for all its occurrences in the term-document matrix. One is added to the total number of documents to increase the IDF for both the unique and non-unique terms and hence does not reduce the discriminative power of IDF. But if a term has a high frequency in a document but is actually not a unique word, then the traditional TF-IDF will give it a weight of 0 whereas equation 3 will assign it weight proportional to its frequency. By this, a word which is important to all the documents and has high TF in a document will not be ignored and hence allows its term frequency to affect the final weight. For this, special treatment is required to handle the stop-words as they are very frequent in all the documents[1].

3. ***TF-IDF based on class frequency (W3):***
Equation 4 is used to modify the existing TF-IDF technique based on the class frequency.

$$\text{Weight}(i,j) = W_{3_{ij}} = TF(i,j) \star \text{Modified}_{IDF}(i) \star \text{Class frequency} \tag{4}$$

Here the $\text{Modified}_{IDF}(i)$ has the same meaning as mentioned in equation 3 and the class frequency is defined as

$$\text{Class frequency} = \frac{n(c_{ij})}{N(c_i)}$$

where $n(c_{ij})$ denotes the total number of documents having the term j and belongs to the class c_i to which document i belongs and $N(c_i)$ represents the total number of documents of c_i. For normalizing class frequency, it is necessary to divide by the number of documents in the class c_i because of the following reasons:

- Case 1: class c_i has 10 documents and the term i appears in c_i 10 times.
- Case 2: class c_i has 100 documents and the term i appears in c_i 10 times.

The term represents the class in Case 1 is better than in Case 2, hence dividing by number of documents would represent this better, as after dividing, the

[1] either completely remove the stop-words or special marker will be used to identify those words

class frequency factor for the term becomes 1 in Case 1 and 0.1 in Case 2. Hence, the number of documents in the class will normalize the frequency of the terms in the class and gives importance to the terms based on their class frequency.

4. ***TF-IDF based on normalized length (W4):*** Equation 5 modified existing TF-IDF based on normalized length.

$$\text{Weight}(i,j) = W_{4_{ij}} = TF(i,j) \star \text{Modified}_{IDF}(i) \star \text{Normalized length} \quad (5)$$

$$\text{Normalized length} = \log_{10}\left(\frac{\text{L}}{\text{L} - TF(i,j) + 1}\right)$$

and the length of document (L) is the sum of the term frequencies of the unique terms. In equation 5, an adjusting factor (normalized length) is added to TF-IDF for term length normalization i.e. to give equal opportunities to both lengthy and short documents and is discussed in the example below.

Example:
Suppose there are 20 documents in a corpus out of which 3 documents ($d_1, d_2\ and\ d_3$) contain the term t. Let,

$$TF(t,d_i)/(\text{document length of } i), i \in \{1,2,3\}$$

be $15/1000$, $15/200$ and $2/10$ respectively. Using traditional TF-IDF formula, the weights of three documents are 28.456, 28.456 and 3.794 respectively. With the adjustment made, the normalized modified TF-IDF values of three documents are 0.412, 2.118 and 0.410 respectively. Accordingly, the document with length of 200 words is the top ranked document, which is probably the required (relevant/useful) document. This makes sense because a small document would contain very little information and a very large document having the same term frequency as have a medium sized document would probably have the relevant information in a part of the document and the rest would not be useful. Hence, we would favor the medium sized document which is desirable in this case.

C. Clusters formation

After converting the documents into vector forms using five different vector forms (one traditional TF-IDF and four modified TF-IDF), the documents are clustered into different groups having each group of similar documents. For this purpose, Fuzzy c-means clustering algorithm is applied on the corpus and it generates k clusters known as *term-doc clusters* $\{td_1, td_2, ..., td_k\}$. Now, the aim is to select important terms from each cluster for maintaining the uniformity without excluding any collection.

D. Important features selection from each cluster

The term-doc vectors are clustered into k term-doc clusters using standard Fuzzy c-means clustering algorithm (i.e. all the terms (r) of the entire corpus P are partitioned into k clusters) and the dimension of each term-doc cluster td_i is now $q \times p$, where q is the number of terms in each td_i. For each term-doc cluster, the centroid of all the term vectors (i.e. the term-doc cluster centroid) is calculated. The cohesion score of each term with respect to the corresponding term-doc cluster centroid is calculated as follows:

i) Centroid of each term-doc cluster td_i is computed using equation 6.

$$\vec{c_i} = \frac{\sum_{j=1}^{q} \vec{t_j}}{q} \quad (6)$$

where, $\vec{c_i}$ is the centroid vector of the i^{th} term-doc cluster and the dimension of $\vec{c_i}$ is $1 \times p$.

ii) To measure how cohesive is the term, $\vec{t_j} \in td_i$ to the centroid, $\vec{c_i} \in td_i$, the Euclidean distance is computed between $\vec{t_j}$ and $\vec{c_i}$ using the equation 7.

$$cohesion(\vec{t_j}) = (||\vec{c_i} - \vec{t_j}||) \quad (7)$$

iii) Now each term of td_i receives a cohesive score. The terms in each term-doc cluster are ranks based on their high cohesive scores and top m% terms are selected from each cluster which then merge together into a global list ($td_{global-list}$).

E. Training the classifier

Different classifiers are trained using the global list ($td_{global-list}$). During testing, by comparing the known class label with the predicted class label, the performance of different classifiers are computed.

F. Performance Evaluation

The following parameters are used to measure the performance of the classifier.

i. *Accuracy (acc)* is the ratio between the sum of true positive cases, *TP* (number of sentences that are near-duplicate and are retrieved by the approach) and true negative cases, *TN* (number of sentences that are not near-duplicate and are not retrieved by the approach) with the total number of documents, *N = TP + FP + TN + FN*. It can be represented as follows:

$$acc = \frac{(TP + TN)}{N}$$

where,
FP: number of documents that are not near-duplicate and are retrieved by the approach and *FN*: number of documents that are near-duplicate and are not retrieved by the approach.

ii. *Precision (pr)* is the fraction of the retrieved documents by the classifier that are relevant.

$$(pr) = \frac{(relevant_{documents}) \cap (retrieved_{documents})}{retrieved_{documents}}$$

iii. *Recall(re)* is the fraction of the relevant documents which are retrieved by the classifiers.

$$(re) = \frac{(relevant_{documents}) \cap (retrieved_{documents})}{relevant_{documents}}$$

iv. *F-measure (F)* is the harmonic mean of *pr* and *re*.

$$\textit{F-measure (F)} = 2 * \left(\frac{pr * re}{pr + re}\right)$$

IV. Experimental Results

Three benchmark datasets are used for experimental work (20-Newsgroups[2], Classic3[3] and Reuters[4]). Nine state-of-the-art classifiers (SVM, Gaussian Naive-Bayes(G-NB), Multinomial Naive-Bayes (M-NB), Binomial Naive-Bayes (B-NB), AdaBoost, Decision Trees (DT), Random Forest (RF), Extra Trees (ET) and KNN (K = 5)) are used for text classification. The details of these three datasets are discussed below.

20-Newsgroups is a standard machine learning dataset and it has 11293 training and 7528 testing documents classified into 20 classes. Three classes are taken into consideration (*alt.atheism*, *soc.religion.christian* and *misc.forsale*) for experimental purpose, consisting of total 1663 training and 1107 test documents. Total number of terms used is 20422 and among them, 16270 are used for training. Classic4 is a well-known benchmark dataset in text mining. It has 4257 training and 2838 test documents classified into 4 classes - *cacm, cisi, cran, med*, having 3204, 1460, 1400 and 1033 documents respectively. All the classes are considered in evaluation. The total vocabulary contained in all documents is 21299 and from them, 15971 terms are selected for training. Reuters is a widely used text mining dataset. It has 5485 training and 2189 testing documents classified into 8 classes, where all class documents are considered for evaluation. The total number of terms used is 17582 and among them, 13531 are used for training.

Tables I and II show the F-measure and accuracy comparisons of five techniques on different datasets, where $W0$ represents the traditional TF-IDF. The bold results in the tables indicate the maximum performance achieved by a classifier using the respective four different modified TF-IDF techniques. Figure 1 shows the F-measure of different classifiers using each individual proposed technique on various datasets. Empirical results show that the proposed four techniques outperform the traditional TF-IDF. Some key points that can be observed from the results are discussed below:

(a) If the classes contain distinguishing terms then the proposed technique ($W1$) which takes interclass disparity into consideration would yield good results and that is reflected in Reuters dataset where most of the classifiers performances using $W1$ are better compared to other three proposed techniques. This is because most of the classes of Reuters have distinguishing terms.

(b) The performance of modified TF-IDF technique ($W2$) is either better or comparable than TF-IDF technique. But its results are not good compared to other three proposed techniques.

(c) Segregation into classes as considered in the third modified technique ($W3$) gives comparable results in classification. Many classifiers have justified it by showing the better performances using $W3$ in classic4 dataset compared to other three techniques.

(d) If the dataset contains documents of different lengths then the normalized length technique ($W4$) would yield good results. It is evident in 20-Newsgroups dataset where majority of the classifiers have shown better performances using $W4$ compared to other three proposed techniques. This is because most of the documents of 20-Newsgroups are of different length.

[2]http://qwone.com/∼jason/20Newsgroups/

[3]http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets/

[4]http://www.daviddlewis.com/resources/testcollections/reuters21578/

V. Conclusion

TF-IDF is a widespread technique generally used for vector representation of a document but has many drawbacks. This paper addresses such drawbacks and proposed four different techniques to represent the document in vector space by modifying the existing TF-IDF technique. To compare their performances against the traditional TF-IDF, text classification using different established classifiers are done on three benchmark datasets. The proposed modified TF-IDF techniques give better or comparable results in most of the cases compared to the traditional TF-IDF technique which shows the effectiveness of the proposed approach. Further scope includes testing the modified TF-IDF techniques in other domains of text mining such as clustering, ranking, text summarization, spam and duplicate page detection etc. Also, these techniques can be tested in semantic-based retrieval system.

References

[1] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[2] G. Domeniconi, G. Moro, R. Pasolini, and C. Sartori, "A comparison of term weighting schemes for text classification and sentiment analysis with a supervised variant of tf. idf," in *International Conference on Data Management Technologies and Applications*. Springer, 2015, pp. 39–58.

[3] S. Albitar, S. Fournier, and B. Espinasse, "An effective tf/idf-based text-to-text semantic similarity measure for text classification," in *International Conference on Web Information Systems Engineering*. Springer, 2014, pp. 105–114.

[4] H. Calvo, "Simple tf· idf is not the best you can get for regionalism classification," in *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 2014, pp. 92–101.

[5] X. Qi and B. D. Davison, "Web page classification: Features and algorithms," *ACM computing surveys (CSUR)*, vol. 41, no. 2, p. 12, 2009.

[6] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining text data*. Springer, 2012, pp. 163–222.

[7] X. Qiu, X. Huang, Z. Liu, and J. Zhou, "Hierarchical text classification with latent concepts," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, 2011, pp. 598–602.

[8] R. K. Roul, A. Bhalla, and A. Srivastava, "Commonality-rarity score computation: A novel feature selection technique using extended feature space of elm for text classification," in *Proceedings of the 8th annual meeting of the Forum on Information Retrieval Evaluation*. ACM, 2016, pp. 37–41.

[9] R. K. Roul and P. Rai, "A new feature selection technique combined with elm feature space for text classification," in *13th International Conference on Natural Language Processing*, 2016, p. 285.

[10] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

TABLE I: F-measure comparisons on different datasets

Classifier	20-NG (F-Measure-%)					Classic4 (F-Measure-%)					Reuters (F-Measure-%)				
	W_0	W_1	W_2	W_3	W_4	W_0	W_1	W_2	W_3	W_4	W_0	W_1	W_2	W_3	W_4
Linear SVM	82.4	87.3	82.9	90.4	**90.6**	80.5	82.7	80.7	**87.6**	84.7	81.7	**86.7**	82.7	83.4	83.4
G-NB	76.4	80.2	77.8	80.5	**81.7**	73.4	77.5	74.2	75.6	**77.8**	79.7	81.2	80.4	81.5	**86.4**
M-NB	78.4	81.3	78.7	81.5	**82.7**	71.7	71.9	71.5	**76.8**	75.2	80.3	82.4	80.3	83.4	**87.6**
B-NB	75.3	**79.5**	76.1	77.8	78.3	73.8	78.7	73.1	79.7	**80.5**	78.7	**84.4**	78.6	82.4	82.6
AdaBoost	81.4	85.2	81.8	86.7	**88.9**	74.8	76.2	75.1	**82.5**	80.7	80.5	81.7	81.6	**86.3**	86.1
DT	70.4	**78.3**	71.2	76.5	78.1	71.2	**77.6**	70.8	73.2	75.6	76.2	**82.3**	77.2	81.5	80.7
RF	82.4	83.4	82.9	91.5	**92.7**	75.6	77.8	76.2	**82.5**	79.6	81.6	**86.4**	81.7	82.4	83.4
ET	80.7	86.5	80.6	84.7	**87.3**	78.4	78.8	78.6	**83.4**	81.7	80.3	83.2	81.7	85.7	**87.3**
KNN	68.2	73.4	69.5	**74.4**	74.2	70.2	**77.7**	71.2	74.2	75.6	71.7	**75.4**	72.8	73.4	75.2

TABLE II: Accuracy comparisons on different datasets

Classifier	20-NG (F-Measure-%)					Classic4 (F-Measure-%)					Reuters (F-Measure-%)				
	W_0	W_1	W_2	W_3	W_4	W_0	W_1	W_2	W_3	W_4	W_0	W_1	W_2	W_3	W_4
Linear SVM	84.8	89.2	84.9	93.9	**94.7**	82.5	84.3	82.7	**90.8**	86.8	83.5	**88.7**	83.7	86.7	85.4
G-NB	78.4	82.1	80.8	**85.7**	82.5	74.4	76.5	75.2	80.6	**84.8**	80.7	84.2	86.2	84.5	**90.4**
M-NB	81.4	84.3	81.7	86.5	**89.7**	73.4	75.3	74.8	**82.3**	80.3	83.7	85.9	83.3	86.4	**90.7**
B-NB	77.2	83.5	78.9	**86.8**	81.2	76.3	82.7	76.1	82.9	**89.5**	80.3	**92.8**	80.6	86.4	87.6
AdaBoost	83.5	88.2	83.8	90.7	**95.9**	76.9	79.2	76.1	**86.5**	83.7	82.3	85.7	82.6	90.3	**92.1**
DT	72.6	**85.3**	74.2	80.4	81.1	74.9	82.6	75.8	**85.2**	84.6	79.2	**85.3**	79.2	83.5	82.7
RF	84.4	85.4	84.9	93.6	**94.9**	73.6	79.8	79.2	**84.6**	81.5	83.5	**89.4**	83.7	85.4	84.4
ET	82.4	**89.5**	82.6	86.7	88.3	80.4	80.8	81.6	**88.4**	83.7	83.3	85.2	83.7	86.7	**90.3**
KNN	70.2	75.3	71.5	77.2	**78.4**	74.2	**82.7**	76.2	79.2	79.6	72.9	**81.4**	76.8	74.4	77.2

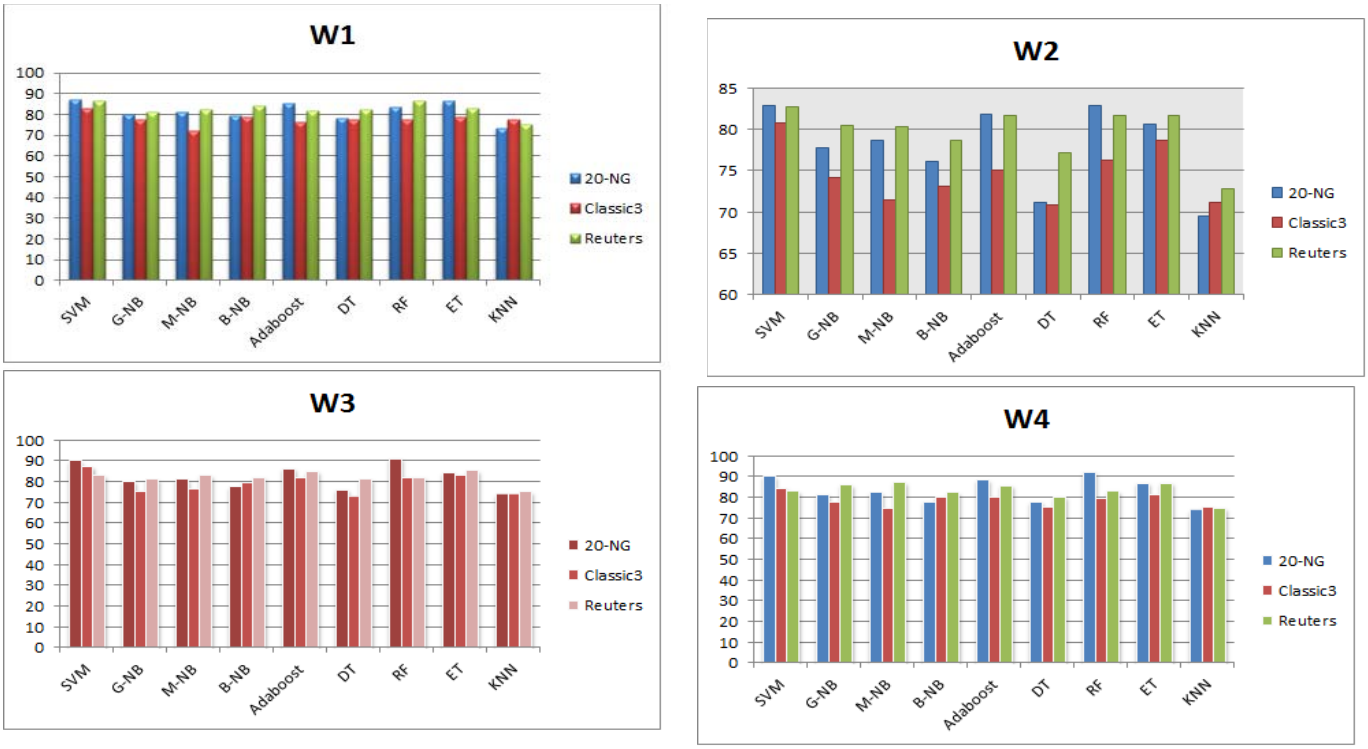

Fig. 1: F-measure comparisons of classifiers on each proposed technique

[11] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2, pp. 191–203, 1984.