An Evaluation of Preprocessing Techniques for Text Classification

Ammar Kadhim
College of Medicine \Baghdad University
17 PUBLICATIONS **696** CITATIONS

An Evaluation of Preprocessing Techniques for Text Classification

Ammar Ismael Kadhim
Department of Computer Science
College of Medicine
University of Baghdad, Iraq
ammarusm70@gmail.com

Abstract: Text preprocessing is a vital stage in text classification (TC) particularly and text mining generally. Text preprocessing tools is to reduce multiple forms of the word to one form. In addition, text preprocessing techniques are provided a lot of significance and widely studied in machine learning. The basic phase in text classification involves preprocessing features, extracting relevant features against the features in a database. However, they have a great impact on reducing the time requirement and speed resources needed. The effect of the preprocessing tools on English text classification is an area of research. This paper provides an evaluation study of several preprocessing tools for English text classification. The study includes using the raw text, the tokenization, the stop words, and the stemmed. Two different methods chi-square and TF-IDF with cosine similarity score for feature extraction are used based on BBC English dataset. The Experimental results show that the text preprocessing effect on the feature extraction methods that enhances the performance of English text classification especially for small threshold values.

Keywords: - Text preprocessing, text classification, tokenization, stop words removal, stemming, chi-square, TF-IDF.

1. Introduction

Text classification (TC) is the task in which texts are classified into one or more of predefined classes based on their contents (Kobayashi, et al., 2017). In addition, text classification that means automatic classification of electronic documents in internet and other-fields such as news, article … etc. TC (also known as text classification or topic spotting) is the task of automatically sorting a set of electronic documents into classes (or categories, or topics) from a predefined set. A topic is simply a collection of words that occur frequently with each other (Sowmya, V., Kiran, K. K. and Putta, 2017). It has an important role to play in the field of natural language processing or other text-based knowledge applications, especially with the recent explosion of readily available text data such as electronic news articles, digital libraries, blogs and Microblogging. Associative text classification, a task that combines the capabilities of association rule mining and classification, is performed in a series of sequential subtasks. They are the preprocessing, the association rule creation, the pruning and the actual classification (Antonie, M. L .and Zaiane, O. R., 2002).

The major objective of text preprocessing is to obtain the key features or key terms from datasets text document and to improve the relevancy between word and document and the relevancy between word and class. It has already been proven that the time spent on preprocessing can take from 50% up to 80% of the entire classification process (Srividhya and Anitha, 2010), which clearly proves the importance of preprocessing in text classification processes.

This paper discuss the types of text preprocessing techniques used in the present research work and analyzes the effect of preprocessing on text classification using machine learning algorithms. Section 2 describes an overview of the work in text preprocessing. Section 3 presents the text preprocessing steps used. Experimental results are included in section 4. Summarization of work narrated in section 5.

2. Related work

The text preprocessing stage of TC is to convert the original textual data to a raw-data structure, where the most significant text-features that serve to distinguish between text-categories are identified. This stage is the most critical and complex process that leads to the representation for each text documents through selection a set of index terms. It is the process of incorporating a new document into an information retrieval system. An effective preprocessor represents the document efficiently in terms of both space (for storing the document) and time (for processing retrieval requests) requirements and maintain good retrieval performance (precision, recall and accuracy). The main objective of text preprocessing is to obtain the key features or key terms from datasets of text documents and to improve the relevancy between word and document and the relevancy between word and class.

1. In [4], Forman presented an extensive comparative study of feature selection metrics for the high-dimensional domain of TC, focusing on support vector machines and 2-class problems, typically with high class skew. It revealed the surprising performance of a new feature selection metric, Bi-Normal Separation. Another contribution of this paper is a novel evaluation methodology that considers the common problem of trying to select one or two metrics that have the best chances of obtaining the best performance for a given dataset. Somewhat surprisingly, selecting the two best performing metrics can be sub-optimal: when the best metric fails, the other may have correlated failures, as it is the case for information gain (IG) and Chi for maximizing precision.
2. In [5], Debole and Sebastiani proposed supervised term weighting (STW), a term weighting methodology specifically designed for IR applications relating supervised learning, such as TC and text filtering. Supervised term indexing leverages on the training data by weighting a term according to how different its distribution is in the positive and negative training examples. In addition, they present that this should obtain the form of replacing inverse document frequency (IDF) by the category based term evaluation function that has previously been used in the term selection phase; as such, STW is also efficient, since it reuses for weighting purposes the scores already computed for term selection purposes
3. In [6], Soucy and Mineau who propose a new technique (ConfWeight) to weight features in the vector space model for text categorization by leveraging the classification task. So far, the most commonly used method is TF-IDF, which is unsupervised; however, there has been little discussion about TC learning.
4. In [7], Ikonomakis et al. presented that automated TC has been considered as a crucial method to manage and process a vast amount of documents in digital forms that are common and continuously increasing. In general, TC plays an essential role in information extraction and summarization, text retrieval, and question answering.
5. In [8], Kamruzzaman and Haider presented a new methodology for TC that requires fewer documents for training. The researchers used word relation i.e association rules from these words are used to derive feature set from pre-classified text documents.
6. In [9], Shi et al. studied TC, that is an important sub-task in the field of text mining. By considering the frequency, dispersion and concentration concurrently, an enhanced feature selection, feature weighting and term reduction method tackling large set of Chinese texts in the real world is planned. Several studies have produced estimates of TC earning techniques, but there is still insufficient data for this field. Most studies in TC have only been carried out in a small number of areas.

3. Text preprocessing steps

The aim behind text preprocessing is to represent each document as a feature vector that is to split the text into individual words. The text documents are formed as transactions. Choosing the keyword through the feature selection process and the main text preprocessing step is necessary for the indexing of documents. On the other hand, text preprocessing stage after reading the input text documents, it divides the text document to features which are called (tokenization, words, terms or attributes), it represents that text document in a data representation as a vector space whose components are that features and their weights which are obtained by the frequency of each feature in that text document, after that it removes the non-informative features such as (stop words, numbers and special characters). The remaining features are next standardized by reducing them to their root using the stemming process. In spite of the non-informative features removal and the stemming process, the dimensionality of the feature space may still be too high. Therefore; the study applies specific thresholds to reduce the size of the feature space for each input text document based on the frequency of each feature in that text document (Kadhim, A. I., Cheah, Y. N. and Ahamed, N. H., 2014).

3.1 Text documents collection

Text documents collection will be divided into two models: training model and testing model. The former refers to pre-classified set of text documents which is used for training the classifier. On the other hand, training models are identified manually to support different text documents classifiers to make database for each topic, while testing model determines the accuracy of the classifier based on the count of correct and incorrect classifications for each text document in that set which is classified by the classifier into suitable main classes.

The training model includes two sets: the first set collects 300 news text documents which were distributed in 10 main categories such as Business, Clothes, Computer, Food, Geography, Medicine, Military, Money and Sport.

3.2 Tokenization

The first step converts HTML files to texts by removing tags from HTML and others tags. Tokenization is commonly understood as any type of natural language text preprocessing. Tokenization is the process of replacing sensitive data, which have a unique identification symbols and it has retained all the essential information about the data without compromising security. Extended Tokenization in our sense does not only separate strings into basic processing units, but also interprets and groups isolated tokens to create higher level tokens. Raw texts are preprocessed and segmented into textual units. The data must be processed in the three operations: the first operation is to convert document to word counts which is equal to bag of word (BOW). The second operation is removing empty sequence i.e. this step comprises cleansing and filtering (e.g., whitespace collapsing, stripping extraneous control characters). Finally, each input text document is segmented into a list of features which are also called (tokens, words, terms or attributes).

3.3 Stop Words Removal

A stop words list is a list of commonly repeated features which emerge in every text document. The common features such as conjunctions such as or, and, but and pronouns he, she, it etc. need to be removed due to it does not have effect and these words add a very little or no value on the classification process (i.e., each feature should be removed when it matches any feature in the stop words list). For the same reason, if the feature is a special character or a number then that feature should be removed. In order to find the stop words, we can arrange our list of terms by frequency and pick the high frequent ones according to their lack of semantics value. They should be removed from words, can also remove very rare words, e.g., words that only occur in m or fewer document, for example m=6.

3.4 Stemming

Stemming is the process of removing affixes (prefixes and suffixes) from features i.e. the process derived for reducing inflected (or sometimes derived) words to their stem. The stem need not to be identified to the original morphological root of the word and it is usually sufficiently related through words map to the similar stem. This process is used to reduce the number of features in the feature space and improve the performance of the classifier when the different forms of features are stemmed into a single feature.

For example: (connect, connects, connected, and connecting)

From the mentioned above example, the set of features is conflated into a single feature by removal of the different suffixes -s, -ed, -ing to get the single feature connect.

There are different kinds of the stemming algorithms; some of them can generate incomplete stems which do not have meaning. One of the most common stemming algorithms uses a set of rules to remove suffixes from features, this process continues until none of the rules apply. This algorithm has some drawbacks such as it is not limited to generate feature stems, for example, "retrieval" becomes "retriev". And it does not deal with prefixes completely, so "relevant" and "irrelevant" stay as unrelated features.

The study implements the stemming process by applying a set of rules in specific way. The rules of the stemming process are as follows:

- Remove all prefixes such as pre-, en-, dis-, etc from features, if the prefix exists in features.

- Use a lexicon to find the root for each irregular feature. Where the lexicon has four major irregular tables (irregular verb, irregular noun, irregular adjective and irregular adverb), each table has some fields which represent unify of each feature such as the irregular verb table has (the verb root, past, past participle, present participle and plural) fields. If the feature matches any feature in the fields of the irregular tables thereby feature should be converted to its stem (root) form which exists in the first field of each irregular table.
  When the only difference among the similar features in the first characters is (-s, -d, -es, -ed, -ly, -er, -ar, -ing, -ance, -ence, -tion, -sion or any other suffixes), thereby features are conflated under the shortest one among them. Thereafter, the weights of the shortest feature results from summing the frequencies of the conflated features.

3.5 Text Document Representation

Typically, each character in text (object) in the training set is represented as a vector in the form (x, d), where $x \in R^n$ is a vector of measurements and d is the class label. Each dimension of this space represents a single feature of that vector and its weight which is computed by the frequency of occurrence for each feature in that text document (i.e for the TC, the IR vector space model is frequently used as the data representation of texts). This study will represent each document vector d as $d=(w_1, w_2,....., w_n)$.
Where w_i is the weight of i^{th} term of document d. This representation is called data representation or vector space model. In this step, each feature is given an initial weight equal to 1. This weight may increase depend on the frequency of each feature in the input text document (i.e., the similar features in size and characters are conflated under a single feature. The weight of a single feature results from summing the initial frequencies of the conflated features).
These steps mentioned above are used to prepare the text document as depicted in Figure 1.
The pseudo code used for preprocessing explains as below:

```
For each document in datasets do
Remove tags from HTML files
End for
For each remaining text document in the dataset do
Remove white space and special character
For each remaining text document in the dataset do
Remove stop words
End for
For each remaining word in the dataset do
Perform Stemming using lexical language and store in a vector (Wordlist)
End for
```

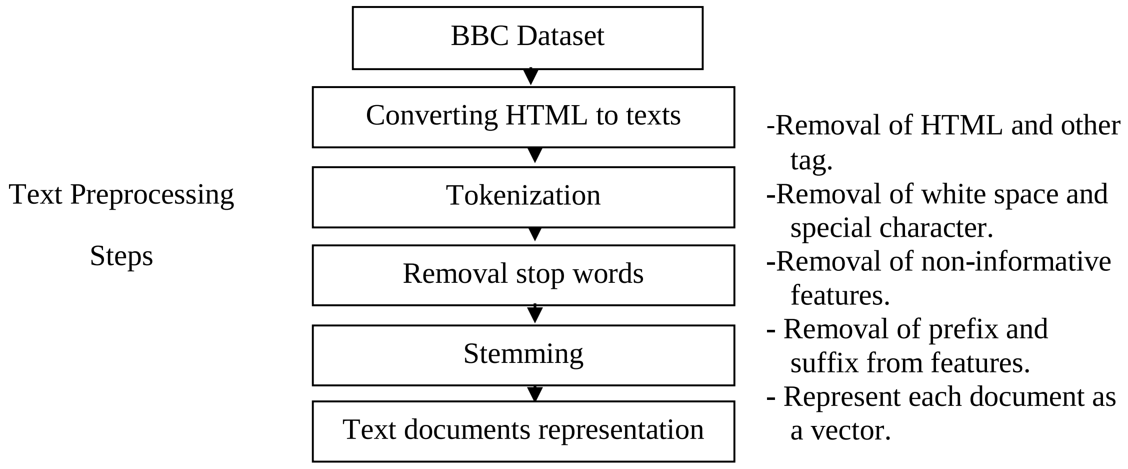
BBC Dataset
Converting HTML to texts
Tokenization
Removal stop words
Stemming
Text documents representation
Text Preprocessing Steps
-Removal of HTML and other tag.
-Removal of white space and special character.
-Removal of non-informative features.
- Removal of prefix and suffix from features.
- Represent each document as a vector.

Figure 1. Text preprocessing steps.

3.6 Indexing Techniques

The major objective of document indexing is to increase the efficiency by extracting from the resulting document a selected set of terms to be used for indexing the document. Document indexing involves choosing the suitable set of keywords based on the whole corpus of documents, and assigning weights to those keywords for each particular document, thus transforming each document into a vector of keyword weights. The weight normally is related to the frequency of occurrence of the term in the document and the number of documents that use that term.

3.6.1 Feature extraction using document frequency

Document frequency (DF) is the number of documents in which a term occurs. Document frequency thresholding is the simplest technique for vocabulary reduction. Stop words elimination explained previously, removes all high frequency words that are irrelevant to the classification task, while DF thresholding removes infrequent words. All words that occur in less than 'm' documents of the text collection are not considered as features, where 'm' is a pre-determined threshold and is shown in Equation 1 given by (Alelyani and Liu, 2012). Document frequency thresholding is based on the assumption that infrequent words are non-informative for category prediction. Document frequency thresholding easily scales to a very large corpora and has the advantage of easy implementation. In the present study, during classification, the DF threshold is set as 1 so that terms that appear in only one document are removed. To compute "m" value by using chi-square equation below:

$$x^2(t,c) = \sum_{i=1}^{N}\sum_{j=1}^{M}\frac{(P_{t,c} - N_{t,c})^2}{N_{t,c}} \qquad (1)$$

Where t is the term in text document, c is the class, N is the number of text document, M is the number of classes, $P_{t,c}$ is the correct term belong to each class, and $N_{t,c}$ is the incorrect term not belong to each class. Figure 2 shows the flowchart of document frequency steps.

Chi-square is a measure of how many term correct counts $P_{t,c}$ and incorrect term counts $N_{t,c}$ deviate from each other. The score can be globalized over all classes in two ways:

1. The first way is to compute the weighted average score all classes.
2. The second way is to choose the maximum score among all classes.

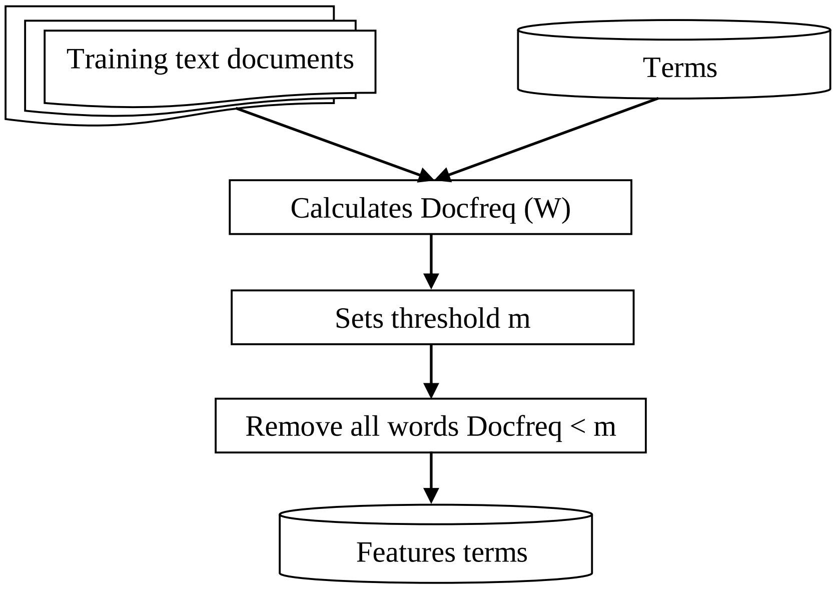

Figure 2. Flowchart of DF steps.

The pseudo code used for DF explains as follows:

```
For each text document in datasets do
```

```
Remove tags from HTML files
End for
For each remaining text document in the dataset do
Remove white space and special character
For each remaining text document in the dataset do
Remove stop words
End for
For each remaining word in the dataset do
Perform Stemming using lexical language and store in a vector (Wordlist)
End for
For each word in the Wordlist do
Calculate chi-square and store the result in a weight matrix
End for
For each element in weight matrix
Set the threshold 'm'
Calculate DF for each term
If DF< m then
Remove the term along with its weight from weight matrix
End if
End for
```

3.6.3 Feature extraction using TF-IDF (term weighting)

Term weighting can be as simple as binary representation or as detailed as a mix of term and dataset existence probabilities stemmed from complex information theoretic underlying concepts. Method like term frequency-relevance frequency (TFRF) express that it is better to award the terms with the highest frequencies in the positive category and penalize the terms with the highest frequencies in the negative category. More or less, TF/IDF is the most widely known and used weighting method, and it is remain even comparable with novel methods. The aim behind text preprocessing stage is to represent each document as a feature vector that is to separate the text into individual words. In TF-IDF terms weighting, the text documents are modeled as transactions. Selecting the keyword for the feature selection process is the main preprocessing step necessary for the indexing of documents. This study used TF-IDF to weight the terms in term-document matrices of our evaluation datasets, given a document collection 'D', a word 'w', and an individual document d. D, the weight w is calculated using Equation 2 and 3 given by (Ramya, M., & Pinakas, A., 2014) as follows

$$TF = TF \times IDF \tag{2}$$

$$w_d = f_{w,d} \times \log \frac{|D|}{f_{w,d}} \tag{3}$$

where fw,d or TF is the number of times 'w' appears in a document 'd', |D| is the size of the dataset, fw,D or IDF is the number of documents in which 'w' appears in D. The result of TF-IDF is a vector with the various terms along with their term weight.

3.6.4 Cosine Similarity Measure

In this study, the researcher used the cosine similarity function between a pair of text documents depends upon the word frequency and number of common words in both text documents. A longer document has higher word frequencies and higher number of unique words than shorter documents producing higher similarity to user queries than shorter documents. So the cosine similarity values of two text documents in the training dataset are found. The cosine similarity between a query vector $\vec{q} = (q_{11}, q_{12}, \ldots, q_{1N1}, q_{1n2}, \ldots, q_{mNm})$ and a text document vector $\vec{d} = (d_{11}, d_{12}, \ldots, d_{1N1}, d_{1n2}, \ldots, d_{mNm})$ is computed in Equation 4 given by (Mao and Chu, 2007) as follows:

$$sim(q,d)=\frac{\vec{q}.\vec{d}}{|\vec{q}|\times|\vec{d}|}=\frac{\sum_{i=1}^{m}\sum_{j=1}^{N_i}q_{ij}\times d_{ij}}{\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{N_i}q_{ij}^2}\times\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{N_i}d_{ij}^2}} \tag{4}$$

Where $|\vec{q}|$ and $|\vec{d}|$ are the norms of the document vectors, q_i is the TF-IDF weight of term *i* in the query, d_i is the TF/IDF weight of term *i* in the document, m is the number of document in the features set, and N_i is the number of the terms belong to document i in the features set. Since $q_{ij} \geq 0$ and $d_{ij} \geq 0$, (q,d) varies from zero to +1 and shows the degree of similarity between a query q and a text documents d. If the similarity between a query and a text document is greater than a predefined threshold value then the text document is the best in features set; otherwise repeat until to all features set.

The pseudo code for the calculation of TF-IDF is depicted as follows:

```
For each text document in Twitter datasets do
Remove tags from HTML files
End for
For each remaining text document in the dataset do
Remove white space and special character
For each remaining text document in the dataset do
Remove stop words
End for
For each remaining word in the dataset do
Perform Stemming using lexical language and store in a vector (Wordlist)
End for
For each word in the Wordlist do
Determine TF, calculate its corresponding weight and store it in
Weight matrix (WD)
Determine IDF
If IDF = zero then
Remove the word from the Wordlist
Remove the corresponding TF from the WD
Else
Calculate TF-IDF and store normalized TF-IDF in the corresponding element of the weight
matrix
End if
Calculate the similarity function
Set the threshold "n"
If TF-IDF <"n"
Remove the term along with its weight from weight matrix
End if
End for
```

4. Experimental results

4.1 Datasets

There are three datasets are used in this paper which is:

- **BBC English Dataset (BBC English)** is an English dataset collected manually from BBC online newspaper. The dataset consists of 4470 articles published from 2012 to 2013. Since there is no standard to split the document this dataset for training and testing, cross validation is performed. Ten random chosen splits were constructed such that the training documents in each split represent fifty of the total number of documents.

4.2 Performance evaluation

A chosen text documents sets are divided into training set and testing set. The training set is used to identify word, which are then used to recast all text documents in the set as bags of words. The accuracy of text documents of the resulting is performed by using tenfold cross-validation.

Initial experiments examined the three methods outlined above for defining key phrases (features); the selection of values for the Boolean weighting, document frequency and TF-IDF with the sets threshold (n) for cosine similarity score and variations in the support and chi-square thresholds (m) used in the application of the documents frequency algorithm.
Each word in the class is used as a classifier to classify both positive and negative text documents in the training dataset. To determine the chi-square function of a word the researcher uses predictive accuracy (Kastrin, A., Peterlin, B., & Hristovski, D., 2010) which is defined as:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{5}$$

According to Table 1, true positive (TP) the number of text document correctly assigned to this category, true negative (TN) the number of text document incorrectly assigned to this category, false positive (FP) the number of text document incorrectly assigned to this category, false negative (FN) the number of text document correctly rejected assigned to this category (Zhang, W., Yoshida, T., & Tang, X., 2011). These terms are used to compute the precision (P), the recall (R), and the F1-measure as follows:

$$\Pr ecision(P) = \frac{TP}{TP+FP} \tag{6}$$

$$\mathrm{Re}\, call(R) = \frac{TP}{TP+FN} \tag{7}$$

$$F1-measure = \frac{2\times P_i \times R_i}{(P_i + R_i)} \tag{8}$$

Table 1. The contingency matrix for a 2-class classification metrics.

Actual category	Predicted category	
	Classified positive	Classified negative
Actual positive	TP	FN
Actual negative	FP	TN

The precision, recall, accuracy and F1-meseaure are shown in the Table 2 for each category by using the two different methods.

Table 2. Performance evaluation for the chi-square and TF-IDF with cosine similarity score methods.

category	Chi-square				TF-IDF			
	Acc.	P		Acc.	P		Acc.	P
Business	70.00	75.00	85.71	80.00	82.00	90.00	87.80	88.89
Clothes	70.00	78.21	82.43	80.26	82.00	91.03	86.59	88.75
Computer	66.00	72.94	84.93	78.48	80.00	89.41	87.36	88.37
Food	70.00	74.12	88.73	80.77	78.00	83.53	89.87	86.59

Geography	63.00	71.43	82.19	76.43	77.00	88.10	85.06	86.55
Medicine	73.00	79.01	86.49	82.58	83.00	91.36	88.10	89.70
Military	72.00	73.26	92.65	81.82	85.00	88.37	93.83	91.02
Money	75.00	80.23	89.61	84.66	83.00	89.53	90.59	90.06
Sport	67.00	74.70	83.78	78.98	77.00	86.75	85.71	86.23
Industry	69.00	78.21	81.33	79.74	82.00	91.36	87.06	89.16
Avg.	69.50	75.71	85.79	80.37	80.90	88.94	88.20	88.53
STDEV	3.50	2.98	3.65	2.29	2.77	2.42	2.60	1.62

Note: Acc.= accuracy, P=precision, R=recall and F1=F-measure.
The performance evaluation of chi-square and TF-IDF with cosine similarity score methods for each category as shown in Figure 3.

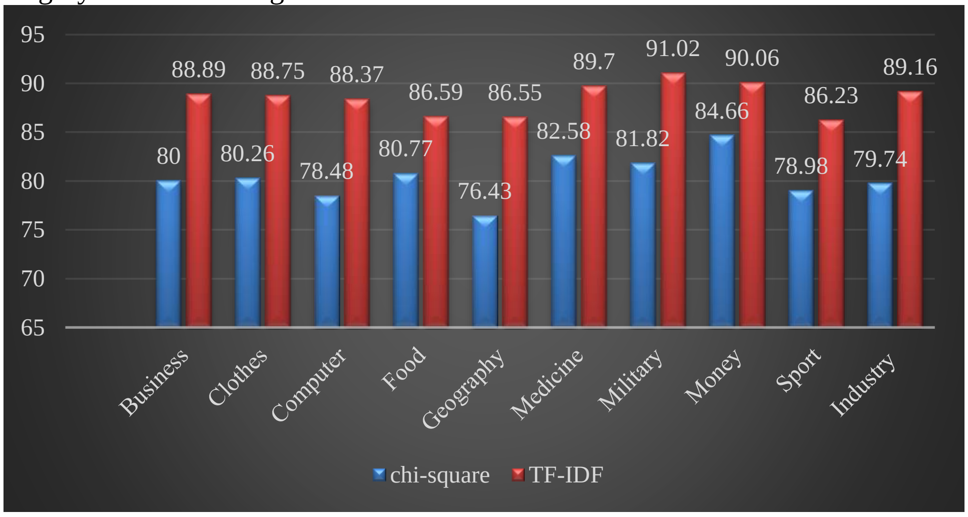

Figure 3. Comparison between chi-square and TF-IDF with cosine similarity score with respect to F1-measure.

The proposed system technique for classifying texts can be classified into one or more topics based on their contents. Figure 4 shows the performance comparison graph between chi-square and TF-IDF with cosine similarity score methods with respect to F1-measure. Blue color indicates F1-measure of chi-square while red color indicates F1-measure of TF-IDF with cosine similarity score method.

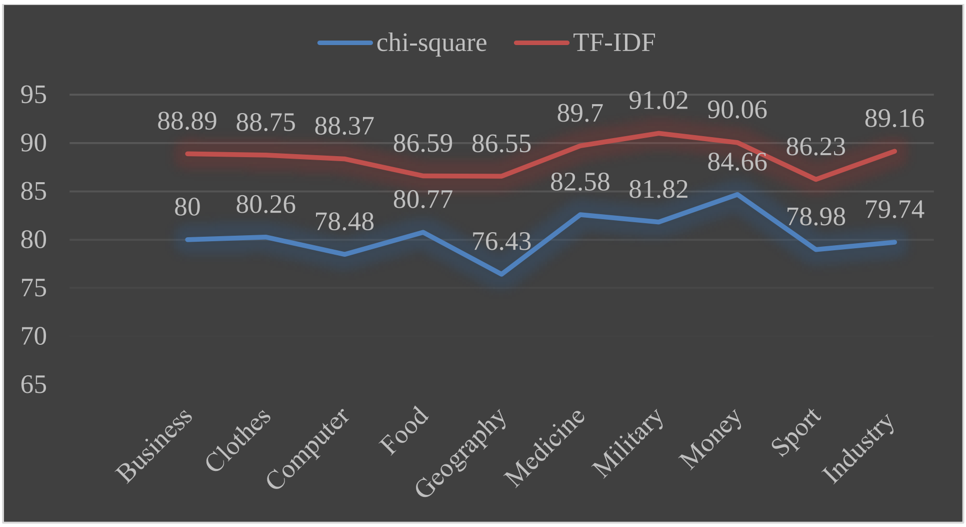

Figure 4. The performance comparison graph between the chi-square and TF-IDF with cosine similarity score methods with respect to F1-measure.

Overall, TF-IDF with cosine similarity score performed better for all categories based on F1-measure.

5. Conclusion:

The paper presents comparing between chi-square and TF-IDF with cosine similarity score methods to classify texts in one or more based on their contents. The proposed system can be used text preprocessing, feature extraction based on the thresholding parameter to classify the texts into one or more categories. TF-IDF with cosine similarity score performed better in classifying the ten general categories based on evaluation metrics. The results showed that text preprocessing can improve the recognition degree of texts and the system performance for text classification.

6. References:

1. Kobayashi, V. B., Mol, S. T., Berkers, H. A., Kismihók, G., & Den Hartog, D. N. (2017). Text classification for organizational researchers: A tutorial. *Organizational Research Methods*, DOI, 1094428117719322.
2. Sowmya, V., Kiran, K. K., & Putta, T. Semantic textual similarity using machine learning algorithms. International journal of current engineering and scientific research (IJCESR) 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-8, 2017.
3. Antonie, M. L., & Zaiane, O. R. (2002). Text document categorization by term association. In Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on (pp. 19-26). IEEE.
4. Srividhya, V., & Anitha, R. (2010). Evaluating preprocessing techniques in text categorization. International journal of computer science and application, 47(11), 49-51.
5. Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. Journal of machine learning research, 3(Mar), 1289-1305.
6. Soucy, P., & Mineau, G. W. (2005, July). Beyond TFIDF weighting for text categorization in the vector space model. In *IJCAI* (Vol. 5, pp. 1130-1135).

7. Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS transactions on computers*, *4*(8), 966-974.
8. Kamruzzaman, S. M., Haider, F., & Hasan, A. R. (2010). Text classification using data mining. *arXiv preprint arXiv:1009.4987*.
9. Shi, L., Mihalcea, R., & Tian, M. (2010, October). Cross language text classification by model translation and semi-supervised learning. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 1057-1067). Association for Computational Linguistics.
10. Kadhim, A. I., Cheah, Y. N., & Ahamed, N. H. (2014, December). Text Document Preprocessing and Dimension Reduction Techniques for Text Document Clustering. In *Artificial Intelligence with Applications in Engineering and Technology (ICAIET), 2014 4th International Conference on* (pp. 69-73). IEEE.
11. Alelyani, S., & Liu, H. (2012, December). Supervised Low Rank Matrix Approximation for Stable Feature Selection. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on* (Vol. 1, pp. 324-329). IEEE.
12. Ramya, M., & Pinakas, A. (2014). J.: Different type of feature selection for text classification. *Intl. J. Comput. Trends Technol.(IJCTT)*, *10*(2), 102-107.
13. Mao, W., & Chu, W. W. (2007). The phrase-based vector space model for automatic retrieval of free-text medical documents. *Data & Knowledge Engineering*, *61*(1), 76-92.
14. Kastrin, A., Peterlin, B., & Hristovski, D. (2010). Chi-square-based scoring function for categorization of MEDLINE citations. *arXiv preprint arXiv:1006.1029*.
15. Zhang, W., Yoshida, T., & Tang, X. (2011). A comparative study of TF* IDF, LSI and multi-words for text classification. *Expert Systems with Applications*, *38*(3), 2758-2765.