JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-8, Aug 2018 E-ISSN: 2347-2693

Evaluating Techniques for Pre-Processing of Unstructured Text For Text Classification

Susan Koshy[1*], R. Padmajavalli[2]

[1]Bharathiar University, Coimbatore, Tamil Nadu, India
[1]Department of Computer Science, St. Thomas College of Arts and Science, Chennai, India.
[2]Department of Computer Applications, Bhaktavatsalam Memorial College for Women, Chennai, India

[*]*Corresponding Author: susanabraham90@gmail.com*

Available online at: www.ijcseonline.org

Accepted: 12/Aug/2018, Published: 31/Aug/2018

Abstract –The availability of digital information over the internet can be analyzed for knowledge discovery and intelligent decision making. Text categorization is an important and extensively studied problem in machine learning. Text classification or grouping of text into appropriate categories requires pre-processing techniques and machine learning algorithms. Pre-processing or data cleaning involves removal of html characters, tokenization, stop words removal, stemming, lemmatization and advanced processes such as parts of speech tagging followed by representation in appropriate form for machine learning. This paper experimentally evaluates the impact of stemming and tokenization techniques on text classification on five text datasets.

***Keywords*—**Tokenisation, stemming, parts of speech tagging, document representation, vector space model

I. INTRODUCTION

The exponential increase in availability of digital information like emails, journals, electronic-books, news and social media can be analysed and useful information extracted. This requires automated text classification and can be used for solving business issues. The task of text classification is to assign the text documents into one or more pre-defined groups. Some of the applications are obtaining the sentiment from social media, identifying spam and non-spam emails, automatic tagging of customer queries and complaints and classification of news articles into pre-defined topics. Pre-processing is the first and most central step before text categorization. The significance of preprocessing is due to the fact that each type of text is highly unstructured and not of uniform standard for example, social media data has informal communication like typing errors, bad grammar, usage of slang words, unwanted content like URLs address and html tags. These have to be removed in order to find patterns. This requires a lot of time before actual text classification can be performed. The following steps are performed as follows - first the documents for analysis are retrieved. Next natural language processing is used to convert raw text using structural, statistical and linguistic techniques. Statistical pattern-matching and similarity techniques are employed to classify documents to a specified group or label. Finally machine learning algorithms are applied and the classification model is evaluated for its performance.

The first section of this paper discusses related work, the next section is the pre-processing of text and discusses basic processing, stemming techniques in particular followed by document representation. The next section is the experimental evaluation of five text datasets using pre-processing techniques followed by classification. The results are tabulated followed by conclusion.

II. RELATED WORK

William B. Frakes and Christopher J Fox (2003) evaluated the strength and similarity among four affix removal stemming algorithms and were evaluated based on the Hamming distance measure [1].

Julie Beth Lovins in her paper “Development of a Stemming Algorithm” in 1968 discussed important linguistic issues in stemming and the variation in spelling of stems and many viable programming solutions are proposed [2].

Chris D. Paice in “An Evaluation Method for Stemming Algorithms” in 1994 describes a technique in which stemming performance is evaluated against predefine concept groups of words using three stemming algorithms [5].

MF Porter in his paper “An Algorithm for Suffix Stripping” in 1980 developed an algorithm for suffix stripping and it performs slightly better than a much more elaborate system. The removed suffix depends upon the form of the remaining stem and considers the length of the syllable [6].

Leon Derczynski et al. in their paper "Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data" in 2013 presented a detailed error analysis of existing taggers when analysing Twitter text which is noisy, with linguistic errors [9].

Anjali Ganesh Jivani in her paper "A comparative study of stemming algorithms" in 2011 has discussed different methods of stemming algorithms and their comparisons and has highlighted the advantages and disadvantages in their applications [10].

Majumder, Prasenjit, et al. in the paper "YASS: Yet another suffix stripper", in 2007 describe an technique which is clustering-based to find out similar classes of root words and their various other forms. This approach was compared with Porter and Lovins stemmer and its performance was on par [11].

III. Pre-processing Text

The text data needs to be cleaned before applying any machine learning algorithms. The pre-processing of text involves basic data cleaning like tokenization, removal of html characters, stop words removal, stemming, lemmatization and advanced processes such as parts of speech tagging. The tokenized words are represented as term vectors in order for machine learning algorithms to be applied. It is important to choose representative words to describe the meaning, and remove words that do not enable to differentiate between the documents. The type of pre-processing will depend on the problem and not all pre-processing activities need to be done. For example converting all the text to lower case cannot be applied always because if "US" is the actual text which means United States and if it is preprocessed to lower case it becomes "us" and the meaning changes. Similarly while tokenizing all white spaces if removed to get tokens will result in loss of meaning.

A. Basic data cleaning or noise removal

Tokenization is the first step to break the stream of characters into words called tokens. The tokenizer has to be customized according to the need of the application because most languages and particular domains have unusual specific tokens that need to be recognized as terms e.g., "Tamil Nadu", "Uttar Pradesh", "San Francisco" which have to be treated as one word although there is a white space.

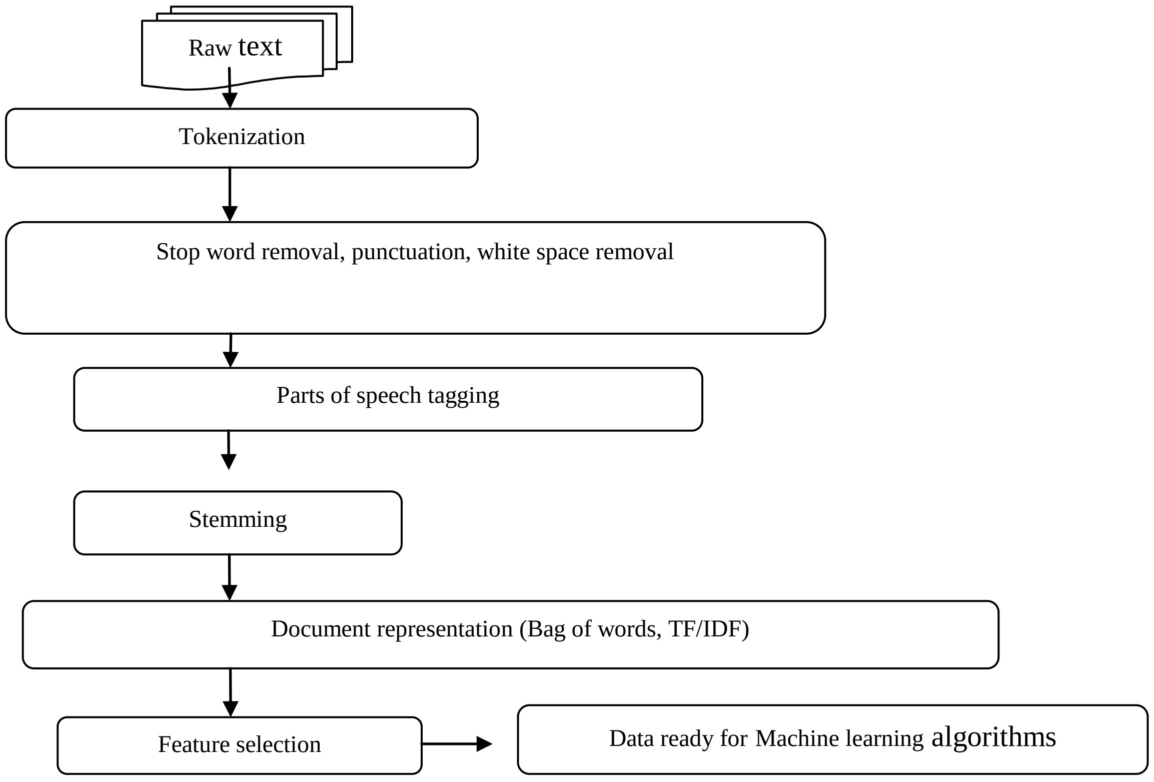
Raw text
Tokenization
Stop word removal, punctuation, white space removal
Parts of speech tagging
Stemming
Document representation (Bag of words, TF/IDF)
Feature selection
Data ready for Machine learning algorithms

Figure 1 Text Pre-processing techniques

Removal of Stop-words: the commonly occurring words (stop-words) should be removed. This can be done by creating a long list of stop-words or predefined language specific libraries available in programming languages such as Python and R can be used.

Removal of Punctuations: Important punctuations have to be retained while others need to be removed for example the full stop after Dr is retained.

Split Attached Words: In social forums text data contain informal words which may have to be split using simple rules. Similarly slangs have to be converted using look up dictionary.
The above are some of the methods to standardize the documents from their raw form and appropriate methods are available in standard programming languages such as PYTHON and R to do the basic data cleaning.

B. Stemming
Stemming is the combining of the variant forms of a word into a single representation called the stem or root. For example, the words “combination”, “combined” and “combining” could all be stemmed to “combine”. The stem need not be a valid word, alternatively it must capture the meaning of the word. Stemming helps to find morphological variants of terms. Conflation can be done manually using regular expressions or use programs called stemmers. Stemming is a lexicon normalization step and required to covert the high dimensional features that are different representation of the same word into lower dimensional feature. Stemming algorithms can be either when the affix is removed or where there is a table or list which is referred [7].

Types of stemming
1. *Affix removal* algorithms use the approach of removing suffixes and/or prefixes from terms resulting in a stem. These algorithms occasionally transform the resultant stem. A common example of an affix removal stemmer is removal of plurals from terms [1]. A set of rules for such a stemmer according to (Harman 1991) is as follows.

Example
If a word ends in "ies” it is replaced with “y” provided there is no “e” or “a” before the “ies”

2. *Successor variety* stemmers developed by Hafer and Weiss in 1974 uses a list of letters in a text corpus as the basis of stemming.

Example
Test Word: CAPABLE
Corpus: ABLE, APE, CAPABILITY, CABLE, CAPACITY, CAPS, CAP, COPE, PIPE,CAPABLE

Prefix	Successor Variety	Letters
C	2	A,O
CA	2	B,P
CAP	1	A
CAPA	2	B,C
CAPAB	2	I, L
CAPABL	1	L
CAPABLE	0	BLANK

If the prefix is ‘C’ the number of letters which follow it in the corpus are listed in the successor variety which is 2 (ABLE, APE, CAPABILITY, CABLE, CAPACITY, CAPS, CAP, COPE, PIPE, CAPABLE) that is A and O of which ‘A’ occurs maximum and hence ‘CA’ becomes the new prefix. The steps are repeated till there are no more words in corpus.

3. *Table Lookup* stemming stores a table of index terms and their roots using a B Tree or Hash Table. The problem with this approach is that it is domain dependent as each domain has its own particular vocabulary . It also requires large overhead storage and the computation is expensive.

Example

Term	Stem
stemming	stem
stemmed	stem
stems	stem

4.*N-grams* is a stemming reported by Adamson and Boreham (1974) which is the method of conflating terms called the shared digram method. A pair of consecutive letters is called digram. Since trigrams, or n-grams could be used, it is known as the n-gram method.
Example
computer => co om mp pu ut te er
unique digrams = co om mp pu ut te er (7 unique digrams for the word ‘computer’)

Stemming algorithms

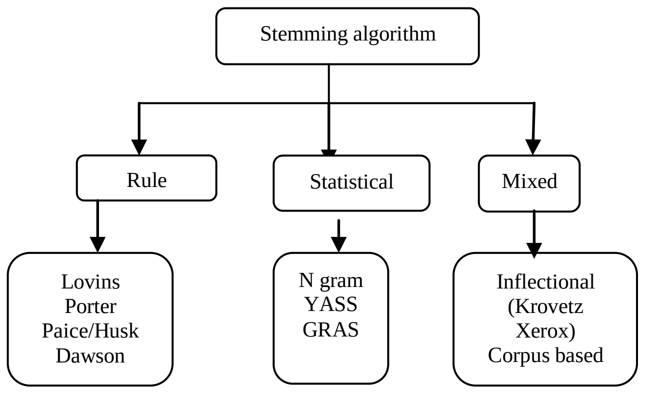

Figure 2 Stemming Algorithms

1. Rule Based

Lovins stemmer is a rule based context sensitive stemmer proposed in 1968. It is language independent and works on the principle of storing a list of suffixes for matching the words with the text document. It executes by searching a lookup table having 294 endings, 29 conditions and 35 rules of transformation. The disadvantage of this stemmer is over stemming of words [2]. Table 1 is the output for raw text when Lovins and Snowball stemmer are applied using WEKA [12].

Table 1 Output of LOVINS stemmer and SNOWBALL stemmer on sample text using WEKA

PlainText	LOVINS stemmer wordtokenizer	SNOWBALL /ngram tokenizer
'We are admiring'	admir	'We are admiring'
'It is awesome'	awesom	'It is awesome'
'You must dispose'	dispos	'You must dispose'
'It is irritating'	irritit	'It is irritating'
'I am loving'	lov	'I am loving'
'I am placed'	plac	'I am placed'

Porters stemming algorithm is one of the most popular stemming methods proposed in 1980. It is based approach that the suffixes in the English language about 1200 are made up of a combination of smaller and simpler suffixes. There are five steps and rules are applied at each step until one of the conditions is satisfied [6]. Porter algorithm is made in the assumption of the absence of a stem dictionary. The program is given an explicit list of suffixes and with each suffix the condition for it to be removed from a word to leave a valid stem.

An example of errors in stemming is as follows

SAND SANDER are conflated and the stem is SAND

WAND WANDER are conflated and the stem is WAND

In the above example suffix removal will alter the meaning of the word and in which case it is not meaningful. The success rate for suffix removal will not be 100% and if more rules are added it will increase the performance in one area of vocabulary but there will be equal degradation elsewhere. Table 2 is the output for raw text when Porter stemmer is implemented using Java.

Table 2 Text before and after using PORTER stemming algorithm using JAVA

Plain Text	After Porter Stemming
discovery	discoveri
(KDT)	kdt
knowledge	knowledg
intelligent	intellig
analysis	analysi

Snowball Stemmer is used for stemming and developed by M.F. Porter in 2001 to address the drawbacks of the porters stemming algorithm. T can be used for languages other an English.

Paice/Husk Stemmer is an iterative, rule based algorithm with 120 rules which are ordered by the last letter of a suffix. Every iteration step tries to locate a rule in order to delete or replace an ending. It is simple and over stemming is its disadvantage [5].

Dawson Stemmer and Lovins approach are alike and it covers a an extensive list of almost 1200 suffixes. It is a single pass stemmer and is swift. The suffixes are stored and ordered by their length and last letter in the reverse order. It is a very complex stemmer and is wanting in reusable implementation.

2. Statistical

N-Gram Stemmer is a language independent method and string-similarity technique is used to convert word inflation to its stem. An n-gram is a set of n consecutive characters extracted from a word. The stemmer has the disadvantage that it requires a large amount of memory and storage for creating and storing the n-grams and hence is not a very practical approach [3].

YASS and GRAS: Yet another suffix stemmer(YASS) and Graph based stemmer(GRAS) are statistical context free algorithms. They are both language independent. In YASS algorithm the string distance measure is used to check the similarity between two words and if the distance is less it shows that the words are similar. GRAS is modeled as a graph and words are represented as nodes and edges connect related nodes. It is computationally less expensive and language independent. The usefulness of stemming depends largely on the application need as there is a chance of over stemming and under stemming.

3. Inflectional/Mixed stemming

This is another methodology to stemming and it involves both the inflectional and modification of words to express different meaning and derivational morphology analysis. In inflectional stemming the variations of the word are related to the language specific syntactic patterns like plural, gender, case, and voice, tense and in derivational the word variants are related to the part-of-speech (POS) of a sentence in which the word occurs.

Krovetz Stemmer (KSTEM) was released in 1993 by Robert Krovetz and is a linguistic based lexical validation stemmer. It is based on inflectional property of words and the syntax of languages, it is very complex in nature. This stemmer transforming the plurals of a word to its singular form converts the tense of a word from past to present and removes the suffix 'ing' from words. This is done by checking with a dictionary. These stemmers have the ability to produce morphologically correct stems, handle exceptions and process prefixes/ suffixes.

C. Advanced text pre-processing

Lemmatization is similar to stemming but is a more organized procedure of obtaining the root form of the word and makes use of vocabulary from dictionary, importance of words and morphological analysis which is word structure and grammar relations [4].

Parts of speech tagging are useful because of the enormous information they provide about a word and its neighbours. Knowledge of parts of speech, that is noun or verb will indicate its neighbouring words. For example nouns are preceded by determiners ('The man') and adjectives and verbs are preceded by nouns ('beautiful dress'). Parts of speech are valuable information features for finding named entities like people or organizations in text and other information extraction tasks. There are training sets for POS tagging namely the Brown Corpus and Penn TreeBank which is commonly used in natural language processing. The tags in Penn Tree Bank are given in Table 3.

Table 3 Partial List of Penn TreeBank

Tag	Description	Example
CC	conjunction, coordinating	*and, or, but*
DT	determiner	*the, a, these*
IN	conjunction, subordinating or preposition	*of, on, under, with*
JJ	adjective	*pretty, tough*
JJR	adjective, comparative	*prettier, tougher*
JJS	adjective, superlative	*prettiest, toughest*
NN	noun, singular	*lion, table, banter*
NNS	noun, plural	*lions, tables, ants*

The following method is used in python programming language to obtain the parts of speech for a given text using the Natural Language Tool kit package. The raw text is first tokenized and then POS tagging method is done.

Example syntax using python

```
>>>text=word_tokenize ("They refuse to permit us to obtain the refuse permit")
>>>nltk.pos_tag(text)
```

Output =

```
[('They','PRP'),('refuse','VBP'),('to','TO'),('permit','VB'),('us','PRP'),('to','TO'),('obtain',
'VB'),('the','DT'),('refuse','NN'),('permit','NN')]
```

This example uses the same word 'refuse' both as a noun and verb and only through Parts of speech they can be identified. The words *refuse* and *permit* both appear as a present tense verb (VBP) and a noun (NN).Parts of speech tagging is useful to identify sentiment in text for sentiment analysis.

After performing parts of speech tagging the data is of the form (word, tag) and is an association between a word and a part-of-speech tag. The program should assign a tag to a word and the tag is the most likely part of speech in a given context. This is similar to mapping from words to tags. The Mapping has to be stored using an associative array or hash array and in some programming languages as a dictionary.

IV. DOCUMENT REPRESENTATION

Bag-of-words and vector space model

When using machine learning we cannot work with text directly and they have to be converted to numbers. Therefore documents have to be converted to fixed-length vectors of numbers. The bag of words model does not consider the order of information in the words and focuses on the occurrence of words in a document. Each word is assigned a unique number. Then any document can be encoded as a fixed-length vector with the length of the vocabulary of known words. The value in each position in the vector can be filled with the number or frequency of each word in the encoded document. An alternative to word frequencies is called Term frequency –Inverse document frequency. Term Frequency summarizes how often a given word appears within a document as in Table 4. Inverse Document Frequency scales words that appear often across documents. TF-IDF highlight words that are more interesting, e.g. frequent in a document but not across documents. The bag of words model uses all words in a document as features and the dimension of the feature space will be the number of words present in the document. In some methods weights are given to the features one when a word is present and zero if the word is not present in the document [8].

A measure for determining the importance of a word(t) in a document(d) on the basis of how often it appeared in the document and a given collection of documents are term frequency(tf) and inverse document frequency(idf). The logic for this measure is that if a word appears frequently in a document, it is given a high score. But if a word appears in many other documents, it is not a unique identifier and therefore we should assign a lower score to that word. tf(t,d) is the number of times each word appeared in each document. Inverse document frequency determines the amount of information a word provides. It identifies if a term is common or rare across all documents. It is the log of the inverse fraction of the documents that contain a word obtained by the division of the total number of documents containing the word

$$idf(t,d) = \log \frac{N}{|1 + \{d \in D : t \in d\}|} \quad (1)$$

where N is the number of documents in the corpus N=|D| and $\{d \in D: t \in d\}$ is the number of documents where the term is present.

$$Tf - idf(t,d,D) = tf(t,d) * idf(t,D) \quad (2)$$

A high weight in tf-idf is obtained by a high word frequency in the document and a low document frequency of the word in the whole collection of documents. The weights filter out common words. Since the ratio inside the idf's log function is always more than or equal to 1, the value of idf and tf-idf is larger than or equal to 0 when the word occurs many times in a document.

Example

Assume a document containing 100 words in which a word 'mouse' appears 3 times. The term frequency (i.e., tf) for 'mouse' is (3 / 100) = 0.03. If there are 5000 documents and the word 'mouse' appears in a 5 of these. Then, the inverse document frequency (i.e., idf) is calculated as log (5000 / 5) = 3. Therefore, the Tf-idf weight is the product of these quantities: 0.03 *3 = 0.09.

Table 4 Term document matrix

		Features/terms						
		t1	t2	t3	t4	t5	…	tn
Documents	d1	1	0	0	0	1		0
	d2	1	1	0	1	1		0
	d3	0	0	1	0	1		0
	d4	0	0	1	1	0		1
	d5	0	1	0	0	0		1
	…							
	dm	1	0	0	0	1		1

Table 5 Comparison of classification results on text datasets for different stemmers and different tokenizers

Eval Metric	Stemmer	Movie review D1			Hotel Review D2			Food review D3			Twitter review D4			Amazon D5		
		J48 classifiers														
		Tokenizers (Word tokenizer, alphabetic tokenizer, N Gram tokenizer														
		WT	AT	N T	WT	AT	N T	WT	AT	N T	WT	AT	N T	WT	AT	N T
	Null	92.52	89.77	92.66	95.85	95.65	96.95	49.94	49.94	49.94	60.69	60.69	60.69	67.62	67.62	67.62
	Iterative	92.06	89.31	92.48	96.6	95.95	96.45	49.94	49.94	49.94	60.69	60.69	60.69	67.62	67.62	67.62
	Lovins	91.92	89.2	92.59	95.8	96.05	96.55	49.94	49.94	49.94	60.69	60.69	60.69	67.62	67.62	67.62
CC	Snow	92.52	89.77	92.66	95.85	95.65	96.95	49.94	49.94	49.94	60.69	60.69	60.69	67.62	67.62	67.62
	Null	0.632	0.44	0.642	0.917	0.913	0.939	0	0	0	0	0	0	0	0	0
	Iterative	0.601	0.396	0.631	0.932	0.919	0.929	0	0	0	0	0	0	0	0	0
	Lovins	0.595	0.396	0.639	0.916	0.921	0.931	0	0	0	0	0	0	0	0	0
KS	Snow	0.632	0.44	0.642	0.917	0.913	0.939	0	0	0	0	0	0	0	0	0
	Null	0.137	0.182	0.133	0.068	0.074	0.052	0.402	0.402	0.402	0.346	0.346	0.346	0.198	0.198	0.198
	Iterative	0.144	0.19	0.135	0.059	0.071	0.062	0.402	0.402	0.402	0.346	0.346	0.346	0.198	0.198	0.198
	Lovins	0.146	0.191	0.133	0.073	0.068	0.06	0.402	0.402	0.402	0.346	0.346	0.346	0.198	0.198	0.198
MAE	Snow	0.137	0.182	0.133	0.068	0.074	0.052	0.402	0.402	0.402	0.346	0.346	0.346	0.198	0.198	0.198
	Null	0.261	0.301	0.258	0.185	0.192	0.161	0.449	0.449	0.449	0.416	0.416	0.416	0.315	0.315	0.315
	Iterative	0.268	0.308	0.26	0.171	0.189	0.176	0.449	0.449	0.449	0.416	0.416	0.416	0.315	0.315	0.315
	Lovins	0.271	0.309	0.258	0.191	0.184	0.173	0.449	0.449	0.449	0.416	0.416	0.416	0.315	0.315	0.315
RMSE	Snow	0.261	0.301	0.258	0.185	0.192	0.161	0.449	0.449	0.449	0.416	0.416	0.416	0.315	0.315	0.315

Table 6 Comparison of classification results on text datasets using Instance based K nearest classifier IBK

	IBK Classifier				
Evalation metrics	Movie review(D1)	Hotel reviews(D2)	Food reviews(D3)	Twitter reviews(D4)	Amazon reviews(D5)
Correctly Clas(CC)	100	100	99.84	99.93	99.92
Kappa Statistic(KS)	1	1	0.99	0.99	0.99
Mean Abs Err(MAE)	0.0005	0.0005	0.0014	0.0007	0.0006
Rt Mean Sq Err(RMSE)	0.0005	0.0005	0.0236	0.0161	0.0146

Table 7 List of text datasets

Text datasets	No of instances
Movie reviews(D1)	2834
Hotel reviews(D2)	2000
Food reviews(D3)	4998
Twitter reviews(D4)	8918
Amazon product reviews(D5)	10261

V. Experimentation and Results

The experiments were performed using WEKA [15] an open source software issued under the GNU General Public License. Text reviews of movies, hotel, food, twitter product reviews and amazon product reviews have been used for experimental evaluation. The five text datasets have been obtained from various online sources are given in Table 7 [13] [14]. The datasets are in CSV file format. The dataset is loaded and class attribute assigned. Filtered classifier is chosen for classification and J48 classifier is used. A string to word vector unsupervised attribute filter was chosen and four stemmers including null stemmer which is a dummy stemmer are applied. The four stemmers are Iterated LOVINS (ITLovins), LOVINS, Null stemmer and SNOWBALL. In addition for each stemmer three tokenizers were chosen for analysis namely Alphabetic Tokenizer(AT), N Gram tokenizer(NG) and Word Tokenizer(WT). In all 120 evaluations were done. Evaluation metrics chosen for comparison were Correctly classified instances (CC), Kappa Statistics(KS), Mean Absolute error (MAE) and Root Mean Squared error (RMSE).

Kappa statistics should be close to 1 and root mean squared error and mean squared error should be close to 0 for a good classifier.

The movie reviews and hotel reviews datasets having 2834 and 2000 instances respectively have been negligibly been influenced by the stemmer and tokenizer when classification is performed.

Datasets with larger number of instances namely food reviews, twitter product reviews and amazon product reviews do not exhibit any change in classification metrics for the J48 classifier for different stemmers or different tokenizers as can be seen in Table 5. Fig 3 to 6 is the graphical representation of Table 5. The correctly classified instances are high for only first two small datasets when the J48 classifier is used. Similarly they have exhibited high kappa statistic and low errors. The reason can point to the fact that they contain fewer instances. When IBK classifier(Instance based K nearest neighbor) is used, the evaluation metrics is uniformly improved for all the five datasets irrespective of stemmer or tokenizer for pre-processing, as given in Table 6 and Fig 7. All the evaluation metrics namely correctly classified instances, kappa statistic have increased and the mean absolute error and root mean squared error have decreased when IBK classifier is used.

`

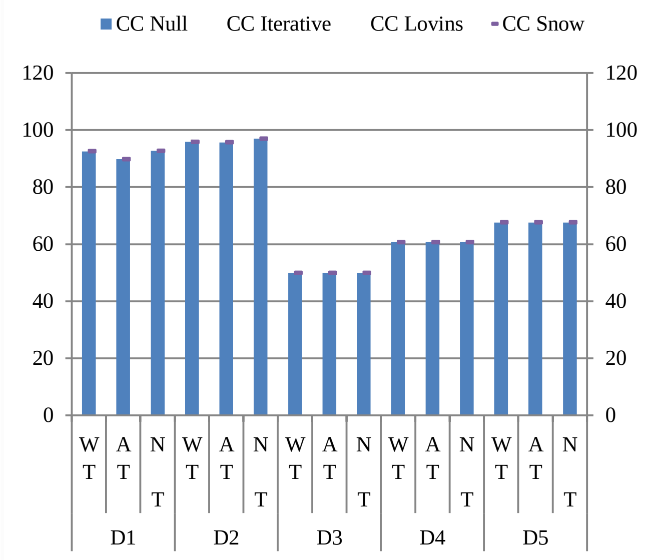

Figure 3 Correctly classified instances for J48 classifier (Table 5)

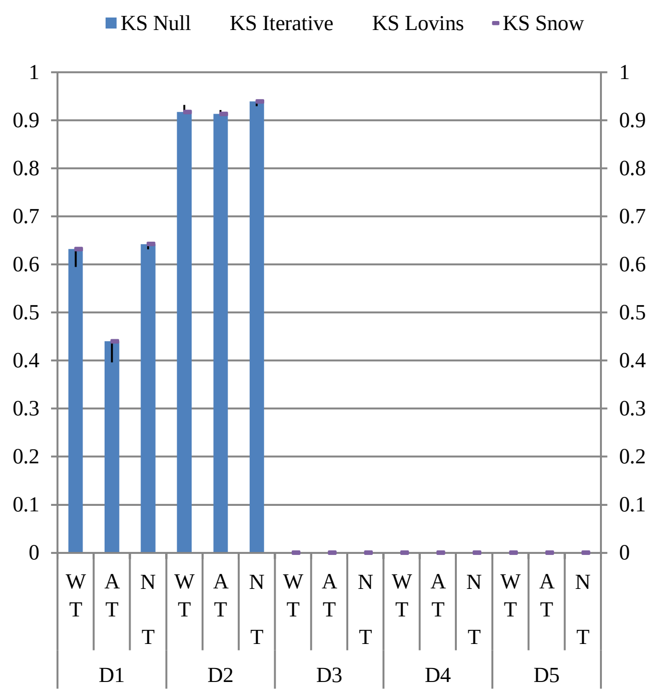

Figure 4 Kappa statistics for J48 classifier (Table 5)

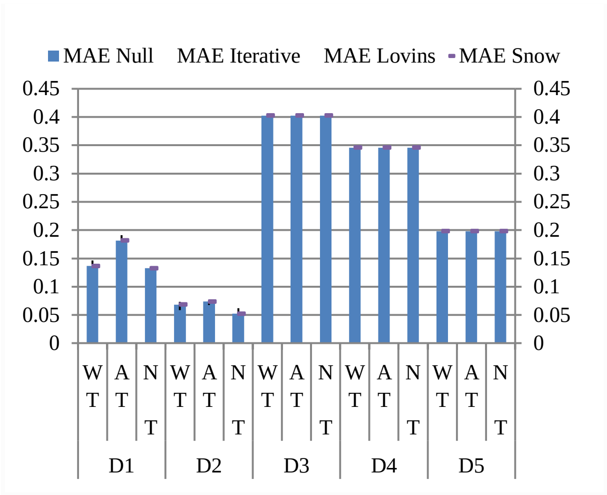

Figure 5 Mean Absolute errors for J48 classifier (Table 5)

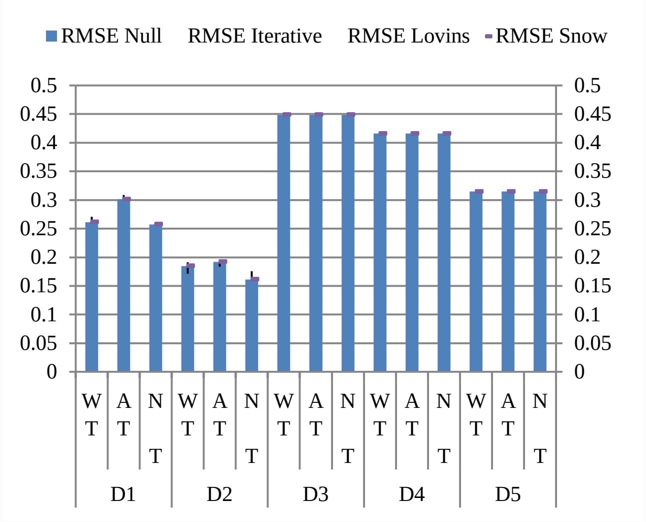

Figure 6 Root Mean Square Error for J48 classifier (Table 5)

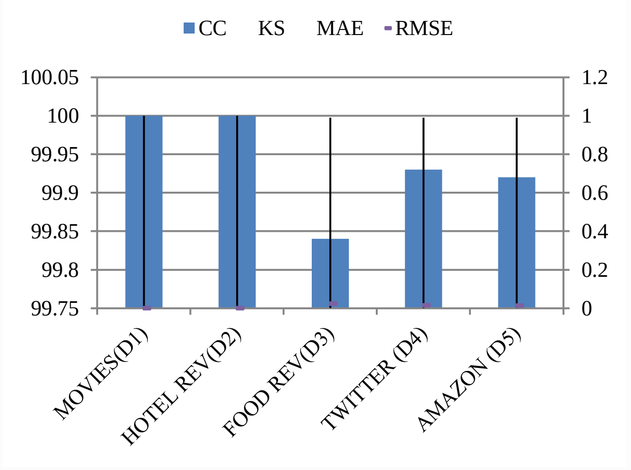

Figure 7 All the evaluation metrics for IBK classifier (Table 6)

VI. CONCLUSION

The purpose of text pre-processing is to clean unstructured raw text and structuring it into a form for machine learning algorithms to be applied. The impact of different stemming and tokenistion techniques on text classification has been evaluated in this paper. Stemming and tokenisation enable to reduce the words to their root form and in turn the dimension of the dataset.

In this paper five text datasets were used to compare the effect of different stemmers and tokenizers on the evaluation metrics of text classification.. Datasets with larger number instances namely food reviews, twitter product reviews and amazon product reviews do not have a huge impact on classification metrics for the J48 classifier for different stemmers or different tokenizers. When IBK classifier (Instance based K nearest neighbor) is used the performance is uniformly improved for all the five datasets irrespective of stemmer or tokenizer used. This also validates previous study that the choice of stemmer does not affect classification [16].

REFERENCES

[1] Frakes William B. “Strength and similarity of affix removal stemming algorithms”. ACM SIGIR Forum, Volume 37, No. 1. 2003, 26-30.

[2] J. B. Lovins, “Development of a stemming algorithm,” Mechanical Translation and Computer Linguistic., vol.11, no.1/2, pp. 22-31, 1968.

[3] Mayfield James and McNamee Paul. “Single Ngram stemming”. Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval. 2003, 415-416.

[4] Mladenic Dunja. “Automatic word lemmatization”. Proceedings B of the 5th International Multi- Conference Information Society IS. 2002, 153-159.

[5] Paice Chris D. “An evaluation method for stemming algorithms”. Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. 1994, 42- 50.

[6] Porter M.F. “An algorithm for suffix stripping”. Program. 1980; 14, 130-137. Porter M.F. “Snowball: A language for stemming algorithms”. 2001.

[7] Hull David A. and Grefenstette Gregory. “A detailed analysis of English stemming algorithms”. Rank Xerox ResearchCenter Technical Report. 1996. (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[8] Han, Jiawei, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011

[9] Derczynski, Leon, et al. "Twitter part-of-speech tagging for all: Overcoming sparse and noisy data." Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013. 2013.

[10] Jivani, Anjali Ganesh. "A comparative study of stemming algorithms." Int. J. Comp. Tech. Appl 2.6 (2011): 1930-1938.

[11] Majumder, Prasenjit, et al. "YASS: Yet another suffix stripper." ACM transactions on information systems (TOIS) 25.4 (2007): 18.

[12] Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical

Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.
[13] https://www.kaggle.com/ranjitha1/hotel-reviews-city-chennai/version/2#
[14] https://www.kaggle.com/uciml/sms-spam-collection-dataset/data
[15] https://sourceforge.net/projects/weka/
[16] Pomikálek, J., & Rehurek, R. (2007). The Influence of preprocessing parameters on text categorization. International Journal of Applied Science, Engineering and Technology, 1, 430-434.

Authors Profile

Susan Koshy currently pursuing Ph.D at Bharathiar University, Coimabatore and working as Assistant Professor at St.Thomas College of Arts and Science, Chennai, India since 2005.

Dr. R.Padmajavalli Associate Professor, Department of Computer Applications, Bhaktavatsalam Memorial College for Women, Chennai, India