International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 24 (2015) pp 43975-43982

Dimensionality Reduction for Text Preprocessing in Text Mining Using NLTK

Rashmi S[1]

Department of Computer Science and Applications, Jnanabharathi Campus, Bangalore University, Bangalore-560056, India
rashmi.karthik123@bub.ernet.in

Hanumanthappa M[2]

Department of Computer Science and Applications, Jnanabharathi Campus Bangalore University, Bangalore-560056, India
hanu6572@bub.ernet.in

Jyothi N M[3]

Department of MCA, Bapuji Institute of Engineering and Technology, Davangere-577004, India
Jyothi_nm@yahoo.com

Abstract
Today we are enjoying the fruits of digital era. The technology now blossoms with the speed of lightening causing turbulence for creating many smart devices. The impact of such a revolution has led to a huge source of information. Perceiving a meticulous piece of data by data querying imposes an enormous amount of time and therefore accuracy becomes a challenge. The result of an information retrieval system for the user query is comparative to the index of data storage. In this paper, the techniques of Natural Language Processing such as Tokenization, Stop Word Removal and Stemming is revisited by combining with various methodologies of Data Mining namely Clustering and Classification in order to achieve increased efficiency, accuracy and decreased time quotient. The result obtained with such an approach is analyzed through a comparative study based on the parameters namely domain of token, feature of tokens, number of tokens generated and time taken for this preprocessing.

Keywords: Clustering, Classification, Information Retrieval, Stemming, Stop-Words, Tokenization.

INTRODUCTION

Digitization, a new globe where we are living is over-flown with inexhaustible information. The data is stored for over many years compensating to its size to grow in large bytes. Looking for required information has become a challenge and is a time-consuming task. In order to solve this problem there are many techniques available. Text mining is one among these techniques that contributes for efficient Information Retrieval (IR). Text mining/knowledge discovery is a process of capturing the interesting data patterns from unstructured documents often required in a web search. The obtained information is tagged for logical finding or general hypotheses about the necessary resource. IR is a science of information searching process for large documents, databases and WWW. IR is divided into sequence of steps as illustrated in figure 1. Here, the user using some form of search engine issues a query say q; q is taken for textual operations to detect if the users' query is in compliance with the structure defined in the IR system and as a result of this a redefined query q' is obtained; q' is further analyzed for query optimization where the source of users' request is defined in its logical view and then indexed for subsequent retrieval.

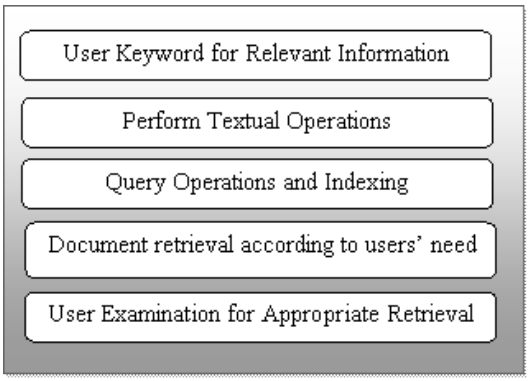

Fig.1. Process Flow of Information Retrieval

Traditional approach of IR creates lot of chaos and awry to provide good results hence for better retrieval IR is combined with various approaches of text mining such as Tokenization, Stop-Words Removal (SWR), Stemming, POS identification and outlier's removal as depicted in figure 2. Illustrations portrayed in figure 2 shows the implications of text preprocessing for an efficient IR system. The principle behind an IR system is to reduce the complexities of the document searching process. However, if the size of the document is huge the retrieval process takes lot longer than expected time. Therefore in order to reduce these discrepancies the use of various text processing modules and effective algorithms are essential. The document when exposed to these modules will be transformed into much simpler version of data that greatly aids for smooth retrieval operations.

Figure 3, shows the flowchart of the pre-processing stages in a text mining environment. These preprocessing stages yields the data query in a much more refined form that facilitates for the fast information retrieval in an IR system. In this research paper, the scope of tokenization, SWR and stemming is evaluated by implementing the proposed algorithm on NLTK interface.

- Tokenization: Process of dividing the documents into token which are the basic building block of a language
- Stop word removal: Process of removing the stop words from the documents. Stop words are the high frequency words in a document that do not play any key role in textual analysis. For example, "the, are, would" are some of the stop words in English language.
- Stemming: Process of tagging the sub-ordinate words to its base forms. Example for stemming process "Loving, Lover, Lovely, Lovable" are the sub-ordinate forms of the base form "Love".
- Parsing / POS detection: Process of breaking down the document into various forms of parsers such as noun phrases, verb phrases and so on In the upcoming section detailed implementation of IR system is explained. To implement IR system, NLTK (Natural Language tool kit)

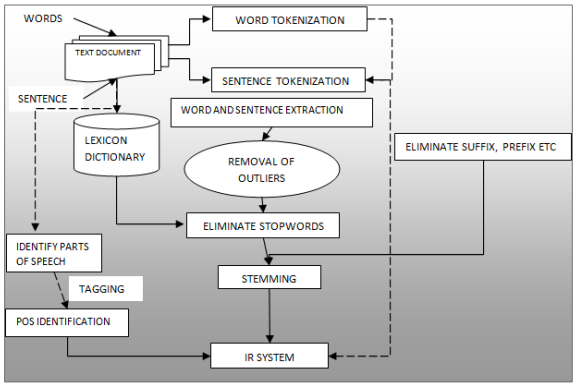

Fig.2. Architecture of the Proposed System

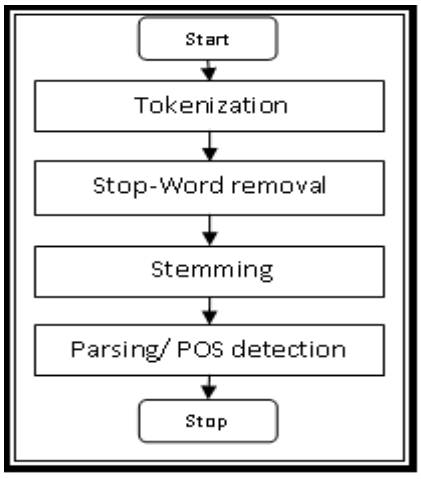

Fig.3. Flow Chart of Text Pre-processing

1.1 Tools used

Natural Language Tool Kit (NLTK) is one of the leading data mining platforms to implement the techniques of text mining and to exert with natural language data. It has over 52 corpora inbuilt for various types of lexical analysis. Using these corpora major types of text processing procedures such as tokenization, tagging, text classification, stemming can be implemented. NLTK is built on python programs that assist diverse range subjects under computational linguistics. It also provides numerous library functions for working with natural languages and to study the structure of lexicons. This tool is freely available online alongside with good API documentation that guide on the usage [1] and is compatible with multiple operating systems such as Windows, MAC and Linux.

LITERATURE SURVEY AND BACKGROUND

Exploration in text mining dates back to few decades from now. Over the years improvisation in terms of accuracy and efficiency in this process is gradually happening. Syntactic structure of a language concentrates on natural language processing at different levels such as text mining, web mining, opinion mining and few others under text classification. In 2010, a model based in k-means algorithm was proposed by Bouras [2]. This included generation of hypernyms through WordNet and constructing "bag-of-words" leading to generation of label. A method to find the frequent item sets were studied using Apriori algorithm [3]. A supervised classification approach was studied and analyzed to reduce the computational time and to increase the accuracy on a preprocessed data [4]. Vikram Singh and Balwinder Singh [5] have proposed a tokenization algorithm relied on the training data and the results were tested across the scoring of IR probabilistic measures. Author Tanu Verma et al [6] has explored the concepts of text mining by using RapidMiner and has performed the tokenization process by using the length as criteria to filter the tokens.

METHODOLOGIES AND IMPLEMENTATION

As explained in previous section the textual preprocessing is series of operations performed on the document that helps for easy retrieval of information. In this section, various techniques and methodologies adopted to achieve syntactic structure of a natural language is described. A step by step procedure to achieve the above said mechanism is interpreted by providing the implementation details as well.

3.1 Tokenization

Tokenization is an intrinsic segment in IR [7] which involves breaking down the documents into individual tokens. A habitual approach for tokenization is implanted using the spaces as a filter to break the document into group of tokens. The outlook of this method is called word tokenization. Tokenization can be performed at sentence level also in which case the document will be divided into set of sentences. However though this task looks simple it contains few challenges 1) Sentence tokenization can be achieved by keeping "." as a filter however for many abbreviations the use

of "." is evident. For example, observe the following sentence. "Mr. Roy lived in New York. He worked in Royal Inc. and earned lot of income for his living." In this sentence definitely "." near Mr. and Inc. is not the right place to break the sentence. However if "." is used as filter criteria for sentence tokenization the output would appear as follows: ["Mr"], ["Roy lived in New York"], ["He worked in Royal Inc"], ["and earned lot of income for his living"]. 2) Observe the following sentence "Mr. Richard O' Colonel and Mrs. Renee O' Colonel aren't taking part in tomorrow's occasion". For O' Colonel which is the correct form of tokenization? [Colonel], [OColonel], [O'Colonel], [O, Colonel]? And for aren't, is it [arent], [are, n, t], [aren, t] or [are, n't]? 3) Problem with hyphenation. The use of hyphens in English is quite confusing and tricky. In few cases hyphens are used while separating the vowels (e.g. Co-operation) and in other places it is used to connect names (e.g. C-DAC). 4) When performing word tokenization, sometimes white spaces cannot be used as filter to build the token. For e.g. the city names such as Hong Kong, New York, Rio de Janeiro and so on exhibits this problem.

In order to overcome these challenges some form of heuristic rules has to be applied. Breaking down the tokens at wrong places in a document is really bad and is totally inappropriate. For instance, if a search is made for "York University" and the IR retrieves "New York University" then it sounds meaningless and incoherent. Therefore it is very difficult to be guaranteed as to which one is a consistent way of achieving tokenization.

In our implementation, a rule based classifier is used to perform tokenization. WordNet, a lexical database and a thesaurus is made use to attain tokenization. Table 1 illustrates the proposed algorithm to obtain set of tokens (word & sentences) in a document. Figure 4 shows the diagrammatic view of this procedure for one instance. Consider the example, "Mr. Richard O' Colonel and Mrs. Renee O' Colonel aren't going to take part in tomorrow's occasion that will be held at San Francisco. Mr. Robert is the chief co-ordinator for this event". By applying the proposed RBC algorithm we get the clusters as shown in figure 4.

Table 1 Proposed Algorithm (Rule Based Classifier RBC)

Rule Based classifier for Word/Sentence Tokenization:
Input: Document
Output: Tokens
Step 1: Perform k-means cluster [Section 3.1.1]
Step 2: Words that appear in 1-cluster form the first set of tokens, words in 2-cluster is another set of tokens.
Filters used for the formation of tokens:

- '.', white space for word tokenization
- .: are used for sentence delimiter

Step 3: Repeat step 2 until the words in all the clusters are grouped as tokens

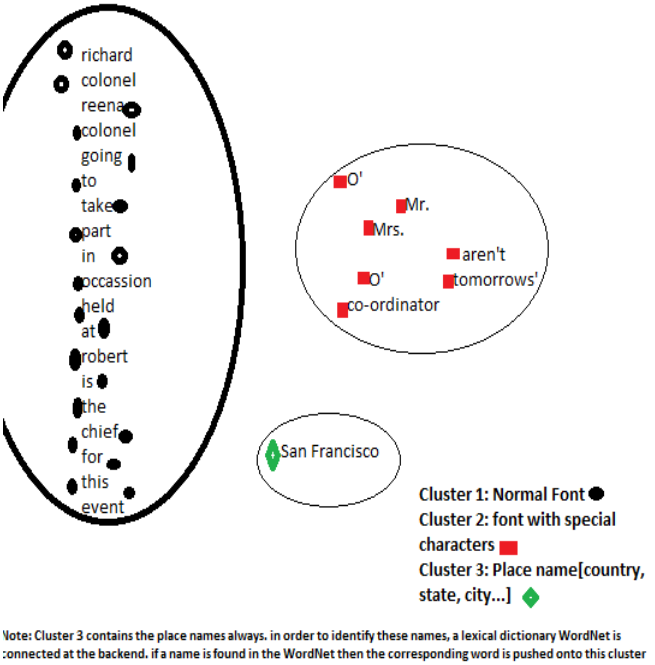

Fig.4. Document Clustering using k-means for Word Tokenization

When the clusters of words are formed, the tagging of WordNet and the contents of these clusters are made. This is truly helpful for the cluster 3 as this contains the names of the places. If the word in found in the WordNet dictionary then the exact match of the word is retrieved from the dictionary. If otherwise, the normal procedure of tokenization is performed. The paradigm of above methodologies is implemented by using one of the efficient computational linguistics tool NLTK which makes use of python programs. This is shown in figure 5 & figure 6. The word tokenization and sentence tokenization is performed for the given string. As observed the correct form of tokenization is exhibited at all the scenarios.

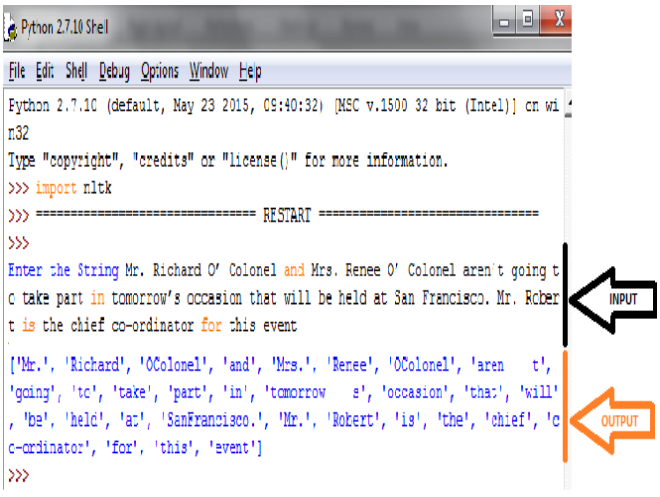

Fig.5. Word tokenization achieved using NLTK tool

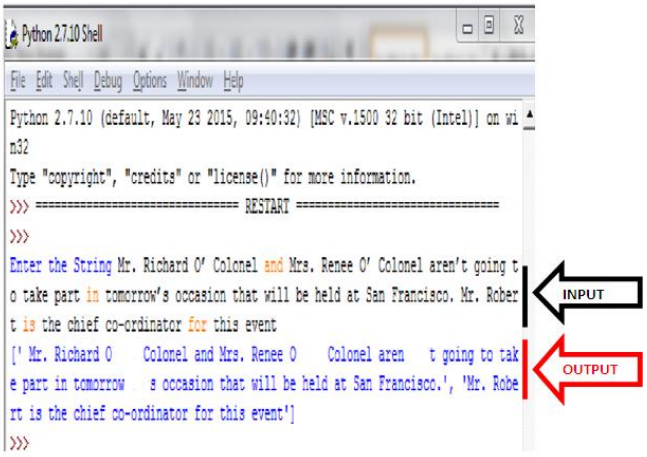

Fig.6. Sentence tokenization achieved using NLTK tool

3. 1. 1 K-means clustering

Clustering is defined as the process of grouping the test data into number of clusters. For instance, in a supermarket, the items are clustered into various categories (apparels, jeans, pants, trousers, t-shirts, formal wear and so on are clusters in clothing sector. Further, this can be once again split into two clusters as men's and women's clothing). In general, we have 'm' data points that need to be partitioned in k-clusters as Mk such that k=1, 2, 3,.n. K-means aims at calculating a Centroid point that minimizes distance of each of the data points to the related cluster.

$$\arg\min_S \sum_{j=1}^{k} \sum_{\mu \in S_i} d(\mu, \sigma_i) = \arg\min_s \sum_{j=1}^{k} \sum_{\mu \in S_i} \| \mu - \sigma_i \|^2 ---1$$

Equation 1 shows the point variance using K-means clustering. Here, "S" is the set of points that belong to any individual cluster say "j". The value of "j" ranges from 1 to k as k is the number of cluster formed for a given document. σ represents the positions across the series of data points in the cluster. Table 2 describes k-means algorithms for data cluster.

Table 2 K-Means Algorithm for Cluster Points

k-means algorithm
Step 1: Initialize the clusters to a seed value (j) that ranges from 1 to k
Step 2: Mark the closest data point to cluster I to belong to "i" cluster
Step 3: Choose the minimum distance between these clusters as shown in equation 1.
Step 4: calculate the Centroid [Euclidean Distance] for each cluster until the clusters are convergent

3.2 Stop word Removal

In text mining, the accuracy of mined data is of utmost crucial. In a context there are many words which do not play any key role in textual search. The words which do not have any significance in a key search are termed as 'stop words'. These words can also be called as 'noise'. Hence removal of noise is very important for any text mining procedure as noise takes up lot of CPU time, search time and memory. Some of the advantages of stop words removal are listed below

- Reduces the textual data in text mining search
- Improves performance by increasing the accuracy and thereby decreasing the time complexity
- The storage space is reduced to minimal
- The overall response time and the throughput is increased as the search for the context takes place in few thousands of documents when compared to millions of documents where the unnecessary words are more likely to be present

In this section we discuss how to eliminate the stop words from a given context using NLTK interface. The procedure is tabulated in the below table 3

Table 3 Proposed Algorithm (Stop word Removal SWR)

Step 1: A list of stop words in English language is prepared before hand
Step 2: The given document is split into token of words
Step 3: Scan each token of the document for a match of a word in list prepared
Step 4: When there is a match between the token scanned and the word in stop words list use string.remove to eliminate the "matched string"
Step 5: Repeat step 4 until EOF is reached

As shown in table 3, the procedure for trimming the stop words is straightforward and comprehensible. The above procedure is compiled using NLTK. When a string of words is passed as an input to this system, it first divides the string into tokens and each token is then compared with the stop words list [figure 6]. When an appropriate match is found the token is removed from the given context. The output is shown in figure 7 with appropriate indication of steps.

```
'all',
'just', 'being', 'over', 'both',
'through', 'yourselves', 'its', 'before', 'herself',
'had', 'should', 'to', 'only', 'under', 'ours', 'has', 'do', 'them',
'his', 'very', 'they', 'not', 'during', 'now', 'him', 'nor', 'did', 'this',
'she', 'each', 'further', 'where', 'few', 'because', 'doing', 'some', 'are',
'our', 'ourselves', 'out', 'what', 'for', 'while', 'does', 'above', 'between',
't', 'be', 'we', 'who', 'were', 'here', 'hers', 'by', 'on', 'about', 'of',
'against', 's', 'or', 'own', 'into', 'yourself', 'down',
'your', 'from', 'her', 'their', 'there',
'been', 'whom', 'too', 'themselves', 'was',
'until', 'more', 'himself', 'that', 'but', 'don', 'with', 'than',
'those', 'he', 'me', 'myself', 'these', 'up', 'will', 'below', 'can',
'theirs', 'my', 'and', 'then', 'is', 'am', 'it', 'an', 'as', 'itself', 'at',
'have', 'in', 'any', 'if', 'again', 'no', 'when', 'same', 'how', 'other',
'which', 'you', 'after', 'most', 'such', 'why', 'a', 'off', 'i', 'yours',
'so', 'the', 'having', 'once']
"""
```

Fig.7. List of stop words saved as a corpus in NLTK

Fig.8. Output showing the input context (black line), Split into tokens (Red line) & Stop Word Removal (SWR) (Green line)

3.3 Stemming/Lemmatization

Stemming is a process of mapping a word to its base form. It is one way of forming tokens to its associated stem. For e.g. 'create', 'creating', 'creator', 'created' and 'creates' all are mapped to the base form 'create'. Stemming plays a vital role in indexing for a searching process which can also affect the performance of search as well. A user's search query for the term related to 'love' should retrieve all the documents containing any word with its base form 'create'. This increases the recall ratio and accuracy of the IR system is also improved.

3.3.1 Proposed methodology and an Example

Stemmer algorithm aims to remove suffixes from a given word. This enables a faster retrieval of information in an IR system. Further the size of the input data query is greatly reduced and the complexity of the search is minimized. In this section we discuss an efficient stemming procedure that is found to remove the suffixes with better results. The algorithm shows the extension of Porter Stemmer algorithm inclusive of enhanced rules. Before studying the procedure of the stemming algorithm there are some considerations to be reserved. These measures are listed below

- Consider two words Wi and Wj which are deliberated to produce a single stemmed word W.
- The above step can be achieved if there are no differences between the 2 words Wi and Wj in two sentences say S1 and S2. For instance, W1= 'love' and W2= 'lovely'. In this example W1 and W2 can be conflated to 'love' without any discrepancies. However if W1= 'objects' and W2= 'objection' are enumerated in two sentences then suffix stemming serves no good purpose here as the meaning of the words are diverse and if they are to be stemmed to 'object' then it gives inappropriate meaning for the context where W2 is present.

3.3.2 Algorithmic Procedure:

Vowels: A, E, I, O, U are called vowels

Consonants: All characters apart from vowels are called consonants

Notations used:

c: Single Consonant
C: Group of Consonants {ccc,…..}
v: Single Vowel
V: Group of Vowels {vvv,…..}
W: Word → Group of vowels and consonants
S: Sentence → Collection of one or more words.

Since word is a combination of vowels and consonants each word can take any form as indicated below in equation I.

CV …….. C
CV …….. V
VC …….. C
VC …….. V I

The generalization of I can be represented as,

[C] VC…….. [V] II

The square brackets in equation II indicates that an arbitrary sequence of either consonants [C] or vowels [V] can be encountered. The suffix removal can be denoted using the following rules:

The suffix removal takes the form, (Condition) W1 → W2

This shows that if the condition is met for a word 'W1' then it can be replaced with 'W2'. Let us consider the length of a word is 'n'. Table 4 (Glossary-1) describes the rules and the example.

Table 4-Proposed Algorithm (Stemmer) and the examples

Rule	Condition	Examples
Rule 1	[(n-1)>1] ment →	Management → Manage Development → Develop
Rule 2	[(n-1)>1] s →	Loves → Love Cares → Care
Rule 3	[(n-1)>1] ed →	Engaged → Engage Discussed →Discuss
Rule 4	[(n-1)>1] ies → y	Accessories →Accessory Accuracies → Accuracy
Rule 5	[(n-1)>1] ss → ss	Discuss → Discuss Access → Access
Rule 6	[(n-1)>1] pping → p	Hopping → Hop Chopping → Chop
Rule 7	[(n-1)>1] ize →	Civilize → Civil Hybridize → Hybrid
Rule 8	[(n-1)>1] ed → e	Created → Create Agreed → Agree
Rule 9	[(n-1)>1] ing →	Talking → Talk Monitoring → Monitor
Rule 10	[(n-1)>1] ization → ize	Rationalization → Rationalize Civilization → Civilize
Rule 11	[(n-1)>1] ational → ate	Relational → Relate Educational → Educate
Rule 12	[(n-1)>1] tional → tion	Additional → Addition Conventional → Convention

Rule 13	[(n-1)>1] izer → ize	Civilizer → Civilize Hybridizer → Hybridize
Rule 14	[(n-1)>1] tive → t	Abstractive → Abstract Attractive → Attract
Rule 15	[(n-1)>1] ator → ate	Creator → Create Accelerator → Accelerate
Rule 16	[(n-1)>1] fulness → ful	Awfulness → Awful Beautifulness → Beautiful
Rule 17	[(n-1)>1] iveness → ive	Effectiveness → Effective Abortiveness → Abortive
Rule 18	[(n-1)>1] lize → l	Generalize → General Animalize → Animal
Rule 19	[(n-1)>1] ivity → ive	Creativity → Creative Sensitivity → Sensitive
Rule 20	[(n-1)>1] alism → al	Socialism → Social Fatalism → Fatal
Rule 21	[(n-1)>1] zable → ze	Fertilizable → Fertilize Recognizable → Recognize
Rule 22	[(n-1)>1] ation → ate	Creation → Create Abbreviation → Abbreviate
Rule 23	[(n-1)>1] sness → s	Biasness → Bias Consciousness → Conscious
Rule 24	[(n-1)>1] lessness →	Agelessness → Age Shamelessness → Shame
Rule 25	[(n)>=1] able → e	Achievable → Achieve Able → e [not satisfied as able = (!(n>=1))]
Rule 26	[(n-1)>1] ly →	Lovely → Love Safely → Safe
Rule 27	[(n-1)>1] ily → y	Happily → Happy Angrily → Angry
Rule 28	[(n-1)>1] ness →	Effectiveness → Effective Kindness → Kind
Rule 29	[(n-1)>1] ful →	Helpful → Help Mindful → Mind
Rule 30	[(n-1)>1] ical → ic	Magical → Magic Ethical → Ethic
Rule 31	[(n-1)>1] iness → y	Happiness → Happy Angriness → Angry
Rule 32	[(n-1)>1] able →	Achievable → Achiev Approachable → Approach
Rule 33	[(n-1)>1] al →	Accidental → Accident Botanical → Botanic
Rule 34	[(n-1)>1] ive →	Creative → Creat Abortive → Abort
Rule 35	[(n-1)>1] ier → y	Happier → Happy Easier → Easy
Rule 36	[(n-1)>1] iest →y	Easiest →Easy Happiest → Happy
Rule 37	[(n-1)>1] zing →ze	Amazing →Amaze

The rules proposed in table 4 (Glossary-1) are user detersive. The search is more composed with the application of these rules. The results obtained with the proposed rules are more polished. However there are many complex words with complex stem. The cut off mechanism is quite not straightforward at all the times. The procedure has to be manifested depending upon the entanglement of the word given as an input or the words present in the document. Furthermore the procedure of suffix stripping is sequential and has to be performed in stages rather than enforcing brute force appeal. To illustrate, observe the word "CHANNELIZATIONS" which takes the following forms:
Step 1: Apply Rule 2 Result: CHANNELIZATION
Step 2: Apply Rule 10 Result: CHANNELIZE
Step 3: Apply Rule 18 Result: CHANNEL

The result of the proposed procedure is shown in figure 9 using NLTK.

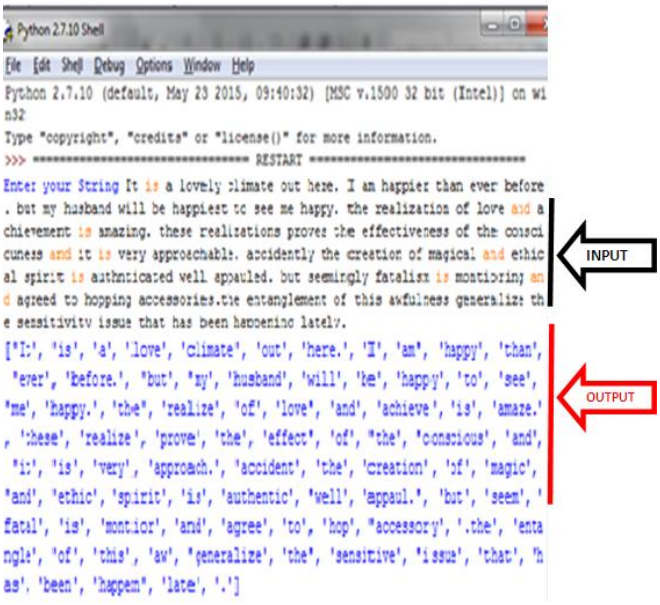

Fig.9. Output of Stemmer algorithm using NLTK

Experimental Results

Analysis test is conducted to study the accuracy of all the proposed algorithms and procedure in various stages in tokenization, SWR and stemming process. To begin with, initially tokenization was carried out as shown in section 3.1. This analogy segregates the given context into individual words in the form of tokens. The results obtained in this evaluation prove that the suggested algorithm can be adopted efficiently for multiple variants of documents with varying size and complexity. Table 5 shows the evaluation outcome. The universal accuracy was calculated from the obtained results and was perceived to be 94% accurate. This manifold implies that espousal of the proposed algorithm is efficient when collated with existing approaches.

In the second step the SWR technique is evaluated. The procedure of SWR is tested for the complexity ratio. The number of tokens is counted in two phases 1) Before applying SWR and 2) After applying SWR. Once the output is obtained the number of words reduced is calculated and the percentage of this reduction is calculated. The test is performed for 3 different instances. The result of this evaluation is indicated in the table 6. As observed from the table SWR procedure reduces the size of context almost by half. The overall

percentage of size reduction of the context is found to be 49%. This is very useful in a key search in text mining. The list of stop words can be scaled up by adding few more words that are not necessarily required for the document search. It is also studied that with the increase in this list the search is refined more with faster and better results.

Table 5 Evaluation result of the proposed RBC procedure

	No. of words in the input context	No. of words correctly tokenized	No. of words not tokenized/ tokenized incorrectly	% of correctness & accuracy
Document 1	94	89	5	95%
Document 2	256	245	11	96%
Document 3	1068	972	96	91%

Table 6 Evaluation result of the proposed SWR procedure

	No. of tokens before SWR	No. of tokens after SWR	Number of words reduced	% of size reduction of the document
Document 1	94	54	40	42%
Document 2	256	198	148	57%
Document 3	1068	589	459	44%

In the last and final stage the performance of stemming algorithm is examined. A corpus of 5000 words was taken to study the performance of the algorithm. The corpus was divided into number of tokens which resulted in 5000 tokens. Proposed stemmer algorithm procedure was followed to accomplish suffix stripping. This analysis is shown in the below table 7. From the table 7, total words that are correctly stemmed are 4454 out of 5000 words. Therefore the overall accuracy achieved by the proposed stemmer procedure is 89%. The result obtained in this appraisal was very much encouraging and it indicates that the data query given to an IR system must be varnished with heterogeneous phases of text pre-processing. When the query undergoes these processing the IR system provides better search results that are apt for the user. Table 8 shows the evaluation test conducted for a profuse range of corpora available in NLTK. NLTK has many inbuilt corpus. A pilot study was coordinated on five of the selected corpora from NLTK.

Therefore from table 8 it is clear that when the text pre-processing phases such as tokenization, stop word removal and stemming is applied the overall document reduction will of 80% less. From these results it is evident that the proposed methodologies can be utilized to improvise the search and to procure more accurate content. It must also be noted that the results attained in this work is independent of the domain and feature space. The number of tokens abundantly affects the overall performance. The nature of tokens, its features affects the stemmer variations of the proposed procedures. The effectiveness of these procedures depends on the utilization of some of the efficient data mining approaches such as classification and clustering. The data samples were studied and the study reveals that there is high reduction rate in terms of size and tokens. This also indicates the time complexity reduction.

Table 7 Evaluation result of the proposed Stemmer procedure

	No. of words
Stemmed in step 1	2987
Stemmed in step 2	1028
Stemmed in step 3	390
Stemmed in step 4	49
Words not stemmed	128
Words stemmed wrongly	418

Table 8 Overall Evaluation Result of the proposed procedures [Tokenization, SWR, and Stemmer]

	No. of tokens	Total No. of tokens after SWR	% of document reduction after SWR	Total No. of tokens after Stemmer algorithm	% of document reduction after Stemmer algorithm
Crubadan	5288	1655	68%	298	81%
Australian Broadcasting Commission 2006	8735	2858	67%	572	79%
ComTrans	11904	4518	62%	914	80%
Brown	16608	4600	72%	796	82%
INSPEC	27468	9672	64%	1853	81%

Conclusion

In this research paper the artifacts of the text pre-processing in text mining is studied. The phases discussed in this work show the techniques to remove the 'noise' which is a major criterion in efficient IR systems. In order to improve the overall performance of an IR interface it is very important to eliminate these 'noises'. We have identified these 'noises' in text mining and classified them in two stages as Stop Word Removal (SWR) and Stemming. To go through in these categories, the data query has to be tokenized based on either words or sentences. To enhance the results data mining techniques are used. K-means algorithm makes tokenization easier and simpler. The final result of this investigation amidst the challenges and shortcomings is encouraging when compared to the existing ones.

Future Enhancements

The extension of this work can be done to evaluate various features of text mining such as POS tagging for a syntactic

language structure, semantic indexing for the knowledge representation and opinion mining for a sentiment analysis.

References:

[1] Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.

[2] Bouras, C. & Tsogkas, V. 2010. "*W-kmeans: Clustering News Articles Using WordNet*".

[3] Maheshwari, P. & Agrawal, J. 2010. "*Centroid Based Text Clustering*". Retrieved October 3, 2013

[4] Qiujun, L. 2010. *"Extraction of News Content for Text Mining Based on Edit Distance"*

[5] Vikram Singh and Balwinder Saini *"Probabilistic Ranking of Documents using Vectors in Information Retrieval"* Balwinder Saini,Vikram Singh Proceedings of the International Conference on Computational Intelligence in Data Mining, 20-21, Volume 33, 2015, pp 613-624 Springer

[6] Tanu Verma et al *"Tokenization and Filtering Process in RapidMiner*" International Journal of Applied Information Systems (IJAIS)-ISSN: 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 7-No. 2, April 2014

[7] S. Ceri et al., *Web Information Retrieval*, "*Informational Retrieval Process*" Data-Centric Systems and Applications, DOI 10.1007/978-3-642-39314-3_2, © Springer-Verlag Berlin Heidelberg 2013