*I.J. Modern Education and Computer Science,* **2018, 9, 20-25**
Published Online September 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2018.09.03

# Comparative Analysis of Stemming Algorithms for Web Text Mining

**Mr. Muhammad Haroon**
Department of Computing & Information Technology
University of Gujrat Lahore Sub Campus, Lahore, Pakistan
Email: haroon.capricorn@gmail.com

Received: 18 July 2018; Accepted: 07 August 2018; Published: 08 September 2018

***Abstract*—**As the massive data is increasing exponentially on web and information retrieval systems and the data retrieval has now become challenging. Stemming is used to produce meaningful terms by stemming characters which finally result in accurate and most relevant results. The core purpose of stemming algorithm is to get useful terms and to reduce grammatical forms in morphological structure of some language. This paper describes the different types of stemming algorithms which work differently in different types of corpus and explains the comparative study of stemming algorithms on the basis of stem production, efficiency and effectiveness in information retrieval systems.

***Index Terms*—**Stemming Algorithms, Stemmers, Information Retrieval, NLP, Morphology, Web Mining.

## I. Introduction

As the data bank is increasing day by day on the web and the search capacity against user query has increased in last few months. The traditional web search works by searching word by word for the given query in document corpus. This method takes a lot of time to search something from a large corpus. To overcome these type of situations, Stemming works in a better way in web text mining. This is the reason that **why we use Stemming Algorithm** in web text mining. User can put query in search engines by any way and the search engine is responsible to extract most relevant results. Stemming is extensively used in many Information Retrieval Systems to meet the exact and accurate answers against user query. Stemming works on a principle to "stem" the word to its root level. Technically, stemming is the conflation of many forms of a word into a single word by stemming it [2]. For example, the terms Close, Closed, Closely and Closing can be stemmed into Close, that is the root term of all given terms.

The words Close, Closed, Closely and Closing are stemmed into word Clos. Searching on the basis of these four terms in query, the results are gathered against stemmed word. In this way, stemming increases the efficiency of a query and represents the most relevant results [1]. The actual reason for this efficiency is to get the root term results from the document corpus.

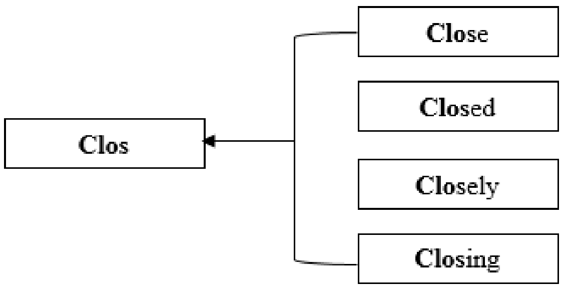

Fig.1

## II. How Stemmer Works?

The working of a stemmer is also very important to understand for complete description of the matter. It is common observation that most of the words are morphologically similar and have similar semantics and those type of words can be considered same for search in Information Retrieval Applications. The stemmer cuts out the terms to make it root term like the root term of 'consideration' is 'consider'. Different type of stemmers are used for this purpose that stem terms into their root terms and the searching is done on stemmed terms instead of actual term to get more authentic results from the large corpus of documents by reducing the processing time [3].

## III. Types of Stemming Algorithms

Stemming algorithms are of many types and work according to their strength, stemming capacity and style of stemming terms. All the stemming algorithms are discussed in this paper with suitable examples to make it more understandable. The classification of algorithms are:

1. Rule Based Stemmers

   i. Porter
   ii. Lovin
   iii. Paice & Husk
   iv. Dawson

2. Statistical Based Stemmers
   i. N-Gram
   ii. YASS
   iii. HMM

3. Corpus Based Stemmers
4. Context Sensitive Stemmers

The above mentioned classification of stemmers are explained in following:

### *3.1 Rule Based Stemmers*

As the name describes, the rule based stemmers work based on some specific rules on which stemming is performed [3]. Many rules are applied for stemming the keywords into their root term [4]. There are many rules applied in different stemmers with different techniques. For example, some stemmers trim keywords by large possible sequence, some trim keywords on some condition etc. the rule based stemmers are comparatively fast as compared to same level stemmers. Also these stemmers work very well by producing a large number of retrieval results in English like languages. There are few rule based stemmers discussed here:

i. Porter's Stemmer
ii. Lovin's Stemmer
iii. Paice & Husk's Stemmer
iv. Dawson's Stemmer

#### *3.1.1 Porter'S Stemmer*

Porter Stemming algorithm [4][6] works on term truncating technique which stem input terms by truncating them into stemmed terms. It is also called ***Affix Removal*** Technique [5]. In Porter's Stemmer, algorithm works well with English language. According to the algorithm, the vowels (A, E, I, O, U) and consonants (Except Vowels) are considered and denoted as Vo and Co respectively. Any word in English can be written as regular expression as:

$$[Co]\ VoCoVoCo\ldots\ [Vo] \tag{1}$$

It can also be simplified as:

$$[Co]\ (VoCo)^m [Vo] \tag{2}$$

Where *'m'* is called as *measure* of any part of word and *'m'* must be greater than or equal to 0. Squared brackets means optional existence of enclosed item. $(VoCo)^m$ means that Vowel and Consonant can repeat infinitely.

For m=0, the (2) becomes [Co] [Vo] and can produce following strings:

EE, FR, FREE

For m=1, the (2) becomes [Co] (VoCo) [Vo] and can produce following strings:

DIMPLE, EAT, GREEK, APPLE

For m=2, the (2) becomes

[Co] (VoCoVoCo) [Vo] and can produce following strings:

PRIVATE, LEMON, UMBRELLA, ASIAN

And so on…

There are some common rules in this algorithm to remove suffix from the string:

- SSES → SS
  - Processes → Process
  - Caresses → Caress
- IES → I
  - Skies → Ski
  - Ponies → Poni
- ATIONAL → ATE
  - Rotational → Rotate
- TIONAL → TION
  - National → Nation
  - Fractional → Fraction
- S → " "
  - Bats → Bat
  - Cars → Car

Some rules are based on condition like:

- (m > 1) EMENT → " " *(whatever comes before emenet has length greater than 1, replace emenet with null)*
  - Replacement → Replac
  - Cement → Cement

There are total 62 rules in Stemmer's algorithm.

#### *3.1.2 Lovin's Stemmer*

Lovin's Stemmer also works with some conditions for Affix Removal of terms. The list of suffix is stored in the system called as *endings*. The algorithm performs two steps [3]. In first step, the longest possible substring from right side is trimmed by matching from already stored ending list in the system. For example, the word "Suffocation" is stemmed into term "Suffoc" that was already present in our ending list. In second step, spelling exceptions are handled if there exist any. Taking term as example "absorption" gives the term "absorpt" and "absorbing" gives "absorb" after applying the stemming process from first step. Now, we get two different terms after stemming from which only one is matched from the ending list. Such type of issues are handled in the second step by applying some post stemming techniques like *partial matching* or *recording*. There are total 29 rules/conditions in this stemmer. There are also 35 transformation rules acting in this stemmer that perform several transformations of keywords. For example, *Removal of Letter Doubling* like 'sitting' is converted into 'sitt' that has letter doubling issue which transformed into 'sit' which is correct one and *Irregular Plural* like matrix has plural matrices and index has plural indices. This is heavy stemmer that produces high data reduction [9].

### *3.1.3 Paice & Husk's Stemmer*

This stemmer works on iterative manner with two steps involved in the algorithm [7]. First step is responsible for removing or replacing endings from the string and the second step is responsible for rewriting rules. There is a separate file that consists of endings' list from which ending is matched and further action is performed. 115 rules are applied in first step. Every algorithm must halt at some time. As it is the iterative based algorithm, so it also has some termination states that are:

- A word which starts with a vowel letter and only two consonant letters left behind.
- A word which starts with a consonant letter and only three characters left behind.

This algorithm is comparatively slow because of iterations that may result *over-stemming* sometimes.

### *3.1.4 Dawson's Stemmer*

Lovin's Stemmer does Affix Removal in its first step and spelling correction is done in second step in which two techniques are majorly used i.e. recording and partial matching. Dawson's Stemmer is actually a refined version of Lovin's Stemmer with more endings in the list [3]. It has two modifications as compared to Lovin's [8].

i. It has more endings in list i.e. 1200 endings that are more than Lovin's Algorithm.
ii. This ignored the recording phase and only partial-matching is used to for word conflation purpose.

Partial matching technique works to match terms till certain limit. The endings are stored in reverse order in ending-to-index order. There are indices against every endings and the ending is revealed by its unique index. Unlike Paice & Husk Stemmer, it is single-passed stemmer in which there is no iteration. Due to this, this is comparatively fast [9].

## *3.2 Statistical based Stemmers*

Information retrieval systems also work with another type of stemmer that is Statistical Based Stemmer [11]. Recent researches [12] have declared this type as an alternative of Rule Based Stemmers. It works good with the languages that has no wide linguistic resources because it does not require knowledgebase of the language specified [13]. It uses statistical approach for stemming. Like Bengali language lacks in linguistic resources but the stemming of Bengali terms are carried out by statistical approach from a large corpus of input language to learn morphological structure of the language. There is no manual task done in these type of stemmers [10]. Everything is done on calculation based due to unsupervised learning. There are few statistical based stemmers discussed here:

i. N – Gram Stemmer
ii. YASS (Yet Another Suffix Stripper) Stemmer
iii. HMM (Hidden Markov Model) Stemmer

### *3.2.1 N – Gram Stemmer*

N – Gram Stemmer is from family of statistical based stemmers [16]. It is language independent stemmer that does not need to know the details of the language. The main technique of this stemmer is to work on consecutive characters of a term forming pairs two characters, three, four and so on. There are called n-grams [15][17]. If a term has *'n'* letters, *n+1* bigrams and *n+2* trigrams are produced. The pair of two letters is called bigram, three letters pair is called trigram and so on. From this, pair of words are determined on the basis of unique diagram, triagrams etc. This is calculated by Dice's Coefficient [14]. The terms "PRODUCTION" and "PRODUCE" has following bigrams, trigrams and tetragrams:

Table 1.

| **Term: PRODUCTION *(n=10)*** | |
|---|---|
| Bigram | *P, PR, RO, OD, DU, UC, CT, TI, IO, ON, N* *(11 bigrams)* |
| Trigram | **P, *PR, PRO, ROD, ODU, DUC, UCT, CTI, TIO, ION, ON*, N** *(12 trigrams)* |
| Tetragram | ***P, **PR, *PRO, PROD, RODU, ODUC, DUCT, UCIT, CTIO, TION, ION*, ON**, N*** *(13 tetragrams)* |
| **Term: PRODUCE *(n=7)*** | |
| Bigram | *P, PR, RO, OD, DU, UC, CE, E* *(8 bigrams)* |
| Trigram | **P, *PR, PRO, ROD, ODU, DUC, UCE, CE*, E** *(9 trigrams)* |
| Tetragram | ***P, **PR, *PRO, PROD, RODU, ODUC, DUCE, UCE*, CE**, E*** *(10 tetragrams)* |

* denotes the padding space.

The above Table.1 describes the formation of Bigram, Trigram and Tetragram of two terms 'PROCUCTION' and 'PRODUCE'. The first term has 10 letters, so it produces 11 bigrams, 12 trigrams and 13 tetragrams. Bigram is the pair of two letters like 'PRODUCTION' has bigrams *P, PR, RO, OD, DU, UC, CT, TI, IO, ON, N* where * shows the padding space. This is done with all grams.

The Dice Coefficient uses Similarity Measure which has formula:

$$S = \frac{2C}{A+B} \tag{3}$$

Where S is similarity measure, A is the value of unique grams in first term, B is the value of unique grams in second term and C is the common grams in first and second term.

Table 2.

| **Similarity Measure of Given Bigrams** |
|---|
| A=11 *(No. of unique bigrams in PRODUCTION)*<br>B=8 *(No. of unique bigrams in PRODUCE)*<br>C=6 (*No. of common bigrams in PRODUCTION and PRODUCE*)<br>$S = \frac{2C}{A+B}$<br>$S = \frac{2(6)}{11+8}$<br>$S = {}^{12}\!/_{19}$<br>$S = 0.63$ |
| **Similarity Measure of Given Trigrams** |
| A=12 *(No. of unique trigrams in PRODUCTION)*<br>B=9 *(No. of unique trigrams in PRODUCE)*<br>C=6 (*No. of common trigrams in PRODUCTION and PRODUCE*)<br>$S = \frac{2C}{A+B}$<br>$S = {}^{12}\!/_{21}$<br>$S = 0.57$ |
| **Similarity Measure of Given Tetragrams** |
| A=14 *(No. of unique tetragrams in PRODUCTION)*<br>B=10 *(No. of unique tetragrams in PRODUCE)*<br>C=6 (*No. of common tetragrams in PRODUCTION and PRODUCE*)<br>$S = \frac{2C}{A+B}$<br>$S = \frac{2(6)}{14+10}$<br>$S = {}^{12}\!/_{24}$<br>$S = 0.50$ |

The result depicts that Dice Coefficient (DC) decreases on elevation of grams or number of pairs of terms. With taken two terms ‘PRODUCE’ and ‘PRODUCTION’, bigrams has DC 0.63, trigrams has DC 0.57 and tetragrams has value 0.50.

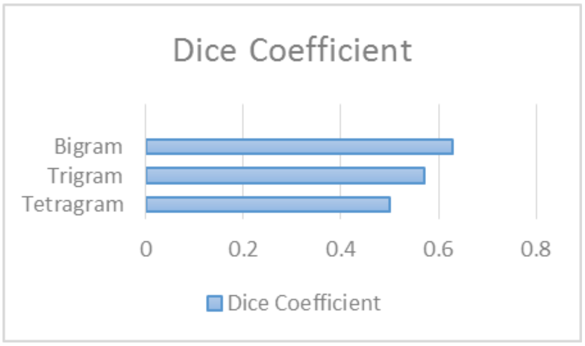

Fig.2.

### *3.2.2 YASS (Yet Another Suffix Stripper) Stemmer*

YASS is another statistical based language independent stemmer. According to [19], the methodology used in this algorithm is to check similarity of two terms based on string distance measure by calculating string distance. Length of both terms must be equal otherwise padding is used to equalize the lengths. It does not rely upon linguistic structure of language [21]. The distance function is devised to calculate the distance of pair of terms. Smaller value of distance depicts the more similarity between terms [1] and vice versa.

Take two strings, A and B defined as:

$$A = a_0, a_1, a_2 \ldots a_n$$
$$B = b_0, b_1, b_2 \ldots b_n$$

And distance function is defined as:

$$p_i = \begin{cases} 0 \quad if\ a_i = b_i \quad 0 \le i \le \min(n, n') \\ \quad 1 \quad otherwise \end{cases}$$

It is a Boolean distance function that returns 0 in case of matching characters at some index *i* and 1 otherwise [19].

The distance measures defined for terms A and B are $D_1$, $D_2$, $D_3$ and $D_4$ are used to store and analyse values.

$$D_1(A,B) = \sum_{i=0}^{n} \frac{1}{2^i} p_i \tag{4}$$

$$D_2(A,B) = \frac{1}{m} \sum_{i-m}^{n} \frac{1}{2^{i-m}} \tag{5}$$

$$D_3(A,B) = \frac{n-m+1}{m} \sum_{i-m}^{n} \frac{1}{2^{i-m}} \tag{6}$$

$$D_4(A,B) = \frac{n-m+1}{n+1} \sum_{i-m}^{n} \frac{1}{2^{i-m}} \tag{7}$$

Where *‘n’* is the total number of characters in term. To elaborate quantitatively, let’s take two terms “BEAUTIFUL” and “BEAUTIFY”.

Table 3.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | E | A | U | T | I | F | U | L |
| B | E | A | U | T | I | F | Y | * |

The mismatch character start from number 7 as the characters 0 – 6 are same in both terms but the different is created on character 7 which is the 8th character, so distance measure set ($D_1$, $D_2$, $D_3$, $D_4$) values are:

$$D_1 = \sum_{i=0}^{8} \frac{1}{2^i} p_i$$

$$D_1 = \frac{1}{2^7} + \frac{1}{2^8} = 0.0039$$

$$D_2 = \frac{1}{7}\left(\frac{1}{2^0} + \frac{1}{2^1}\right) = 0.2142$$

$$D_3 = \frac{8-7+1}{7}\left(\frac{1}{2^0} + \frac{1}{2^1}\right) = 0.4285$$

$$D_4 = \frac{8-7+1}{9}\left(\frac{1}{2^0} + \frac{1}{2^1}\right) = 0.3334$$

The edit distance is 2 in this case. **Edit Distance** is defined as the distance from end to the point where first mismatch occurs.

After calculating distance measures, word clusters are formed by applying complete-linkage algorithm [20].

### *3.2.3 HMM (Hidden Markov Model) Stemmer*

This stemming algorithm is devised by Melucci and Orio [10] which is based on Hidden Markov Model which is a statistical model in which the system work with hidden states. It works on unsupervised learning and does not require prior knowledge of the language. HMM Model is finite state automata in which probability functions are used for stemming purpose. For a term, the predicted path from initial to final states in finite automata is produced a transition from roots to suffixes [10]. Eventually, a stem is the sequence of letters before this point. It may also cause over stemming which is a common issue in stemming algorithm.

### *3.3 Corpus Based Stemmers*

These types of stemmers work on the whole corpus. The main idea is to resolve conflation issues that are still present after Porter's Stemmer [11]. For example, the terms "police" and "policy" are conflating terms as they have similar stem word "polic" but they have entirely different meanings. On the other hand, the terms "matrix" and "matrices" do not conflate with each other as they are from same root but different stemmed terms. Porter's Stemmer still lags to resolve these type of issues. Corpus based stemmers work on a principle of automatic modification of conflating classes because it involves the complete corpus while stemming process. Like the terms having same roots are stemmed using statistical techniques. Some or many terms re-occur in corpus that imparts effects on the results. The significance of terms is defined by:

$$Sig(a,b) = \frac{nab}{na + nb} \tag{8}$$

*'a'* and *'b'* are the pair of terms being examined. *'na'* and *'nb'* are the number of occurrences of terms *a* and *b* in the corpus respectively. *'nab'* is the number of occurrences of both *a* and *b*.

This stemmer is not used standalone but in with the combination of Porter's Stemmer. First Porter's Stemmer is applied and the conflated terms are identified which then sent to the Corpus Based Stemmer that redefines the conflations. If there is no any suitable term is found, it returns the original return rather sending some impropriate stem. Also this stemmer has some drawbacks which involve the high processing time.

### *3.4 Context Sensitive Stemmers*

This is one type of stemmer that is very unique in its working and methodology. The trivial method is to analyze terms in the document but stemmer's approach is to analyze terms on query level before sending it to the search engine. The main idea is to do stemming process as part of query rather during indexing. From this process, after getting terms from the query, context sensitive matching of a document is done which not only merely reduce the time cost but also improves precision of resultant records.

This stemming algorithm is based on four steps:

- ***i.*** ***Query Segmentation:*** First, the long query is broken down into segments that are generally noun phrases from which the candidate terms are generated that goes in another process of stemming.
- ***ii.*** ***Candidate Generation:*** The step involves the identification of candidate terms fetched from the query that need to be analyzed under the algorithm. These are said to be candidate terms.
- ***iii.*** ***Context Sensitive Word Expansion:*** After getting selected candidates, the next step is to check the usefulness of the terms. A specific threshold is defined upon which the candidate value is checked. The main transformation is of plural terms.
- ***iv.*** ***Context Sensitive Document Matching:*** Term variants are matched from the documents in terms of context as in the given query.

## IV. CONCLUSION

With the exponentially increasing data on web and overall in information retrieval systems, there is a need to fetch accurate and efficient data from enormous corpus. In this paper, we have thrown some light on stemmers and the algorithms they use. It can be judged that stemming really put many advancements in information retrieval systems by improving the search efficiency with all types of stemmers that reduce terms to its common root, called stem. The stem determination is one of the key purpose to proceed stemming process. The mature languages which have been known from many decades use language dependent stemmers i.e. rule based whereas some languages for which complete structure is not defined need to use language independent stemmers i.e. statistical based stemmers. Such types of stemmers don't bother the language structure because they work on statistical basis. These stemmers also reduce the size of index files. The number of rules, number of suffix and strength of rule based stemmers are described in the below table:

Table 4.

| Stemmer | No. of Rules | No. of Suffix | Strength |
|---|---|---|---|
| Porter | 62 | 51 | Weak |
| Lovins | 29 | 294 | Strong |
| Dawson | - | 1200 | Strong |
| Paice/Husk | 115 | - | Strong |

## V. Future Recommendations

Although very efficient stemming algorithms have been developed that work in a very efficient way but still there is need for more improvement to better stemming algorithms that deal with syntactically and semantically. There are some stemmers that do *over-stemming (a process to product more stemmers than required)* or under-stemming *(a process to product less stemmers than required).* This improvement need also be added to stemming algorithms. Also there is also a need to produce correct words as there are some stemming algorithms that product incorrect words in some cases.

## References

[1] Deepika Sharma, Stemming Algorithms: *A Comparative Study and their Analysis, International Journal of Applied Information Systems (IJAIS) Foundation of Computer Science,* FCS, New York, USA September 2012 ISSN : 2249-0868 Volume 4– No.3

[2] Narayan L. Bhamidipati and Sankar K. Pal, *Stemming via distribution based word segregation for classification and retrival,IEEE Transaction on system,man,and cybernetics*. Vol 37, No.2 April 2007.

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

[4] M. F. Porter 1980. *"An Algorithm for Suffix Stripping Program"*, 14(3), 130-37.

[5] Sandeep R. Sirsat, Dr. Vinay Chavan and Dr. Hemant S. Mahalle, *Strength and Accuracy Analysis of Affix Removal Stemming Algorithms, International Journal of Computer Science and Information Technologies,* Vol. 4 (2) , 2013, 265 - 269.

[6] Porter M.F. *“Snowball: A language for stemming algorithms”*. 2001.

[7] Paice Chris D. *“Another stemmer”*. ACM SIGIR Forum, Volume 24, No. 3. 1990, 56-61.

[8] Frakes, W. *"Stemming Algorithms."* In Frakes, W. and R. Baeza-Yates, Information Retrieval: Data Structures and Algorithms. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[9] Eiman Tamah Al-Shammari, *“Towards An Error-Free Stemming”*, in Proceedings of ADIS European Conference Data Mining 2008, pp. 160-163.

[10] Melucci, M. & Orio, N. (2003). *“A novel method for stemmer generation based on hidden Markov models. In Proceedings of the Twelfth International Conference on Information and Knowledge Management”*, (pp. 131-138). New York, NY: ACM Press.

[11] J. Xu and W. B. Croft 1998. *“Corpus-based stemming using co-occurrence of word variants”*. ACM Trans. Inf. Syst. 16, 1, 61–81.

[12] D.W. Oard, G.A. Levow and C.I. Cabezas 2001. CLEF experiments at Maryland: *“Statistical stemming and back off translation”*. In Revised Papers from the Workshop of Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation (CLEF), Springer, London, 176–187.

[13] M. Bacchin, N. Ferro, and M. Melucci 2005. *“A probabilistic model for stemmer generation”*. Inf. Process. Manage. 41, 1, 121–137.

[14] WB Frakes, 1992,*“Stemming Algorithm “, in “Information Retrieval Data Structures and Algorithm”*,Chapter 8, page 132-139.

[15] JH Paik, Mandar Mitra, Swapan K. Parui, Kalervo Jarvelin, *“GRAS ,An effective and efficient stemming algorithm for information retrieval”*, ACM Transaction on Information System Volume 29 Issue 4, December 2011, Chapter 19, page 20-24

[16] Mayfield, J. & McNamee, P. (2003). *“Single n-gram stemming. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval”*, (pp. 415-416). New York, NY: ACM Press.

[17] P. McNamee, and J. Mayfield 2004. *“Character n-gram tokenization for European language text retrieval”*, Inf. Retr. 7(1-2), 73–97.

[18] Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, Gobinda Kole, Pabitra Mitra and Kalyankumar Datta. *“YASS: Yet another suffix stripper”*. ACM Transactions on Information Systems. Volume 2 , Issue 4. 2007, Article No. 18.

[19] A. K. Jain, M.N. Murthy, and P. J. Flynn 1999. *“Data clustering”*: A review. ACM Comput. Surv. 31, 3, 264–323.

[20] Majumder, P., Mitra, M., Parui, S. K., Kole, G., Mitra, P. & Datta, K. (2007). *“YASS: yet another suffix stripper. ACM Transactions on Information Systems”*, 25(4), paper 18.

## Author’s Profile

**Muhammad Haroon** is working as Associate Lecturer at University of Gujrat Lahore Sub Campus and student of MS Computer Science with the specialization of Database and Data Mining in Virtual University of Pakistan. His areas of interest are Database Development, Data Mining and Data Processing.

**How to cite this paper:** Muhammad Haroon, " Comparative Analysis of Stemming Algorithms for Web Text Mining", International Journal of Modern Education and Computer Science(IJMECS), Vol.10, No.9, pp. 20-25, 2018.DOI: 10.5815/ijmecs.2018.09.03