DOI: http://dx.doi.org/10.26483/ijarcs.v9i5.6308

ISSN No. 0976-5697

Volume 9, No. 5, September-October 2018

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

REVIEW ON STEMMING TECHNIQUES

Prabhjot Kaur
Department of Computer Science and Engineering
Sant Longowal Institute of Engineering and Technology
Longowal, Sangrur (Punjab) – 148106, India

Preetpal Kaur Buttar
Department of Computer Science and Engineering
Sant Longowal Institute of Engineering and Technology
Longowal, Sangrur (Punjab) – 148106, India

Abstract: Stemming is a method of deriving root word from the inflected word. The stemming process is often called conflation and is done by stemmers or stemming algorithms. The stemming algorithm is the process that reduces all the words of the same basis in a common form. The algorithm is basic building block for the stemmer. The development of stemmer is based on language and requires specific language knowledge and spell checking for that language. This paper, presents an overview of different stemming techniques and algorithms which have been used by the researchers for stemming in different languages.

Keywords: Stemming; Stemming techniques; Survey

I. INTRODUCTION

1.1 Stemming: Stemming is a process that reduces the comparative morphological variation of the words into a single term called stem or root word, without performing a complete morphological analysis [1]. Stemming is the process of removing the affixes from inflected words to their original, basic or root form [2]. The main objective of the stemmer is to find the root word from the inflected words. Stemming is finished by removing attached prefixes and suffixes from words. Stemming will cut the inflections from the word and get the root word as a result. For example, a stemming algorithm stem the words "applied", "applies" and "applying" to the root word "apply". The example of stemming shown in figure 1.1. Various languages like English, Hindi, Marathi, Nepali, Bengali used stemming in their information retrieval systems. The first English stemmer was distributed in 1968. It was written by Julie Beth Lovins [3]. A later stemmer was written by Martin Porter in 1980 [4].

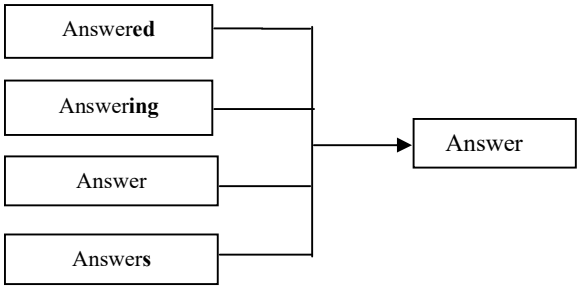

Figure 1.1 Stemming Example

The derived words answered, answer, answering and answers are converted to the root word **answer**, through which not only retrieval performance improves as well as capacity can be enhanced in some particular applications.

Stemming can be used for indexing and search system. In order to develop any application in NLP like text extraction, machine interpretation, document arrangement, topic tracking, text outline, etc., stemmer is required as a basic linguistic resource for any language in the world to attain high accuracy [5].

1.2 Stemmer: Stemmer is a system whose input is an inflected word and it provides the output in the form of a root word. The inflected word can be singular, plural or it may contain some other affixes. The root word is the correct word that contains some dictionary meaning [1].

1.3 Stemming Algorithms: A stemming algorithm is a technique which is used by stemmer [1]. The stemming algorithms may include pattern matching algorithms, stochastic algorithms, hybrid algorithms, porter stemming algorithm, dictionary-based algorithms, rule-based algorithms, corpus-based algorithms, n-gram techniques etc. A stemming algorithm can be characterized as context-sensitive or context-free[1]. In a context-free algorithm, no restriction is placed on the removal of suffix and the matched ending is stripped if any ending matches. In a context-sensitive algorithm, various restrictions are placed when we are removing the suffixes from the words. For the removal of suffixes from the word, these algorithms have to construct some rules and some dictionary sets. Rules tell us which suffix is to be removed and how; and dictionary tells us whether the word is present or not. Porter stemmer is the context-sensitive; Rule-based stemmer that is widely used.

1.4 The purposes of a stemming algorithm

A stemming algorithm, or stemmer has two main purposes:

1) The first one comprises of words with the same term being clustered into single class which reduces dictionary size. It helps to reduce the storage space.
2) Secondly, the base word form is matched with the variant forms of words in documents and queries, which solves the problem of mismatch in vocabulary [6].

1.5 Stemming Errors:

Two types of errors occur in the stemming process one is over-stemming and another is under-stemming.

Over-stemming occurs when the word refers to distinct concepts even though they are converted to the same stem. For example, "compute" and "compile" getting stemmed to "comp".

Under-stemming is occurred when the two words which have similar root are not reduced to the same stem. For example, "compiling being stemmed to "compil" and "compile" to "comp"[7].

1.6 Classification of Stemming Techniques:

Stemming process contains rich writing, and various stemmers have been produced since few years. Stemming technique may run from simple methodologies, for example the elimination of plural and participle present and passed to complex methodologies that expel a lot of suffixes and incorporate a dictionary. Current stemming algorithms categorized are: rule based, statistical, or hybrid which have their own normal way to find the stems of the variation word form. The order of these stemming techniques is displayed in figure 1.2.

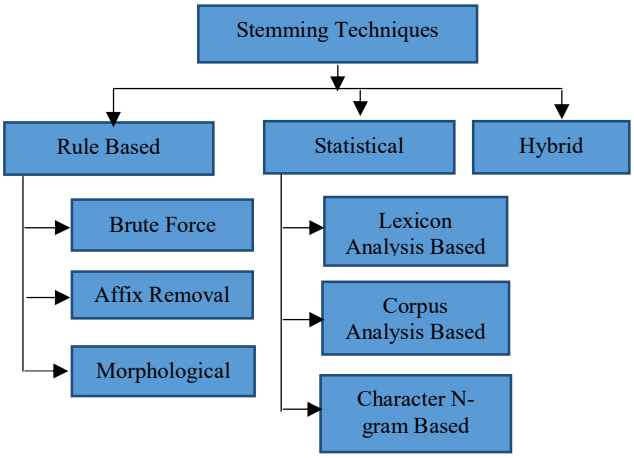

Figure 1.2 Categorization of stemming techniques.

1.6.1 Rule based stemmers

Rule based stemmers are also known as language-specific stemmers because they convert variant word forms into their stem. The language-specific rules are created if these are measure experience in language. These Stemmers are better than Statistic Stemmers in the implementation of complex language code rules [8] [6]. These stemmers are divided into three categories:

1) **Brute Force Algorithms:** In order to get the root of the word, brute force algorithms use a search table. The search table contain list of root words and their inflected words. The table is searched to find the appropriate inflection, as well as the root form of the word associated with it. These stemming techniques also known as dictionary-based or table lookup techniques. These stemmers can consider the inflected word forms that does not properly adopt language rules, for example; suffix removal algorithms can stem from the word "eating" to "eat", so it does not stem the infrequent inflection "ate".
2) **Affix Removal Algorithms:** These algorithms remove prefixes and/or suffixes from inflected words. These stemmers used context-sensitive rules and suffix/prefix list to get the stem. The disadvantage of these stemmers is that the stem created after removing the suffix is not the real word of the language.
3) **Morphological Stemmers:** To perform stemming, morphological stemmers contain inflectional and derivational morphological analysis. These stemmers prefer dictionaries in special languages with word combinations organized using grammatical and morphological variations [9]. Inflectional analysis will recognize the forms of words because of gender, mood, time, event, status, number, face. Derivational analysis will recognize the changes in the word portion (POS) and to minimize the surface to mold forms which it produces. Such as "*advancement*" is stemmed to "*advance*" however "*department*" is not stemmed to *"depart"* because each form has a totally different linguistics. These stemmers serve morphological correct roots and can evaluate different exceptional cases and process roots from lexicon, using rules and as a lexicon.

1.6.2 Statistical Stemmers

The stemmers use semi-supervised or unsupervised learning to find out stemming rules of the language [6]. They cluster morphologically connected words using the nearby set of documents, thus avoiding the use of language experts or additional language resources. Therefore, statistical stemmers are also known as *corpus-based or language independent* stemmers. A number of studies [8-13] showed that statistical stemmers used are reasonable substitutes for a specific language stemmer, particularly for languages in which language resources are not completed [6].

1) **Lexicon Analysis-Based Stemmers:** Lexicon analysis based stemmers evaluate cluster of words attained from the lexicon to group related lexical words. They find possible suffixes and stems by different techniques such as calculating distances [12], the frequency of substrings [14], etc.
2) **Corpus Analysis-Based Stemmers:** Corpus analysis based stemmers group morphologically familiar words, and then analyze their context or appearance in the lexicon. They confirmed the fact that the words used inside the corps are the best representative for inclusion, then words that do not occur together [12].
3) **Character N-Gram-Based Stemmers:** These stemmers recall the rules of stemming by the frequency of n-grams derived from vocabulary words. They can manage morphological differences in alphabetic language [11].

1.6.3 Hybrid Stemmers

To perform stemming, hybrid stemmers combine various different methods [6]. The combination of methods increased the effectiveness of the stemmer. The stemmers can be formed by combining various methods, for example, combination of different methods based on rules or combination of rule-based approaches with statistical methods. Such as, the suffix stripping algorithm can be further enhanced by table searches for non-common verb forms (such as ran / run) or singular / plural formulas [6]. A number of hybrid stemmers [15-19] have been developed for different languages.

II. LITERATURE SURVEY

Literature survey of stemming for English Language:

Julie Beth Lovins (1968) [3] discussed the practical and theoretical attributes of stemming algorithms. He proposed new version of the longest-match, context-sensitive stemming algorithm for English language that can be developed for use in a library information transfer system.

Porter (1980) [4] suggested suffix stripping algorithm which has been implemented as a short, fast program in BCPL. It performs better than a much more elaborated system with which it has been compared.

Paice (1994) [20] developed a method for evaluating stemming algorithms. The method is dealing with stemmer assessment which depends on counting and detecting actual errors and errors that

occur when creating sample words that are obtained from real texts. This makes it possible to calculate the index of "stemming weight" for each stemmer, and also describing the general accuracy and the error rates of over and under stemming. This method involves the separation of word samples manually into conceptual groups and into these groups that reference actual performance indicators.

Xu and Croft (1998) [21] suggested a methodology for the error in the results obtained from the statistical characteristics of the group is used. The main idea of the generation of equivalence of equations for words methods with conventional logs and "separate back" mixed multiple words based on their participation in the lexicon.

Mayfield and McNamee (2003) [22] developed an output of n-grams, which determines that when one n-gram in the form of a pseudo-word stem to be effective and neutral approach for some languages. Since different morphological parts (common suffixes and prefixes, such as “able” or “ing”) will occur more frequently than invariants (unique word roots), a typical statistic can be used to identify them.

Jenkins and Smith (2005) [23] suggested a conservative stemming algorithm for search and indexing. The algorithm is recognized by words that do not need to be derived, it works on rules that are also used as steps. It contains rules that are divided into two sets: the first set is used to clean the icons; second set is used to change the suffixes. The first set of rules avoids a small list of six common problem words. Second, the required upper joints are removed and the contractions expand. Contains 139 suffix rules that are used to test certain types of suffixes. The result of stemmer shows that it frequently meets these goals in approximately 85% or more of words that it stems.

Massimo and Nicola (2003) [24] developed a statistical method for generating a Hidden Markov Models (HMMs) stemmer. The approach based was on unsupervised learning without prior knowledge and it created training set manually. The HMM topology determines the number of states, the initial and final states, the states marking as belonging to one of two groups, and permitting transfers. The transition was composed by a probability function. In every transition, the new state expands a symbol and links possibilities. The set of symbols before division is considered a stalk, remaining as a suffix.

Peng et.al.(2007) [25] proposed a contextually sensitive web search. In the algorithm to determine the distribution of labor similarities, corpus analysis is used. Porter's morphological rules are then applied to the list of similarities in order to find the stemming candidates that come from, some of which are chosen based on the goal of dealing, for example, pluralization. In the non-converter index, the forms obtained are used to extend the search query. For example, in view of the word "present", the application of the rules applies to " presenting, presented, presents ". For the purposes of pluralization, only "presents" is selected. Thus, a current user request is expanded to "present" is expanded to “present” or “presents”.

Literature survey of stemming for Indian Languages:

Ramanathan et.al (2003) [26] developed a lightweight stemmer for Hindi language. In this stemmer, words conflated the terms by suffix removal for information retrieval. The proposed lightweight trunk of the Hindi language was based on Indian grammar, where a list of 65 total suffixes was generated manually. The accuracy of the lightweight stemmer for Hindi language was 88%.

Dasgupta and Ng (2006) [27] developed unsupervised morphological analysis of Bengali language. The algorithm is used to segment words into stems, suffixes and prefixes with no prior knowledge of the morphological rules of a particular language. This consists of two steps: (1) the activation of suffixes, prefixes and root vocabulary containing words taken from a large unknown group, (2) division of the words based on these induced segments. If calculated on a group of fragmented 4-110 phonetic words of Bengali language, the algorithm has the F-83% level, significantly superior to semantics, one of the most traditional unmonitored morphological analyst, with about 23%.

Zahurul et.al (2009) [28] proposed a lightweight stemmer for Bengali language spell checker. The lightweight stemmer was used to find the root of an input word. The authors reported the efficiency of the lightweight stemmer for Bengali language was 90.8%.

Gupta and Lehal (2011) [2] suggested on Punjabi language noun and proper name stemming. In their approach, an attempt was made to get the root or stem the words of Punjabi language, and then to examine it against the Punjabi name and the correct name dictionary. An in-depth analysis of our Punjabi news group was conducted and the different rules of the name, correct name and possible different suffixes were identified, such as ੀਆਂ ੀਆਂ ਾਂ, ਿਆਂ iāṃ, ੀਏ ੀਏ etc. The authors reported the accuracy of stemmer is 87.37%.

Kumar and Rana (2011) [5] presented a design and development stemmer for Punjabi language, which used a brute force and suffix stripping technique. Brute force does not require text preprocessing. The authors used substitution with suffix striping, in order to avoid the problem of over-stemming and under-stemming. The authors reported the average accuracy of the stemmer as 80.73%. Sundra et.al (2010) [30] proposed a morphological analyzer for Tamil language. The analyzer was used to change the Tamil word to equal metrological Tamil words(lemmas). The accuracy of the stemmer was 91.7%.

Juhi et.al (2012) [29] developed a lightweight stemmer for Gujarati. In this proposed method the lightweight stemmer algorithm was used for stems the Gujarati words. The lightweight stemmer had an average accuracy of 91.5%.

Mishra et.al (2012) [19] developed MAULIK: An efficient stemmer for Hindi language. In this stemmer, MAULIK algorithm was used to stem the Hindi words by using Hybrid approach. The stemmer used the hybrid approach for stemming the Hindi words. The accuracy of the stemmer is 91.59%.

Suba et al. (2011) [31] suggested two stemmers of Guajarati language- (1) lightweight inflectional stemmer used the hybrid approach, and (2) heavyweight derivational stemmer used the rule-based approach. Accuracy of inflectional stemmer was 90.7% and was as large as IR and the accuracy of derivational stemmer was 70.7%.

Table 1: Existing stemmers for Indian languages

Reference	Method	Description	Accuracy
Kumar et.al, 2010 [5]	Brute Force algorithm	Truncate the derivational forms from a word.	80.7%
Gupta et.al, 2011 [2]	Rule-based	Rules used for stem the words of noun and proper names of Punjabi language	87.37%
Upendra Mishra et.al, 2012 [19]	MAULIK algorithm	Combination of Brute force and suffix removal algorithm	91.8%
Dasgupta and Ng, 2006 [27]	Rule-based	Segmenting words into suffixes, prefixes and stems.	83%
Ramanathan et.al, 2004 [26]	Lightweight algorithm	find the root of the word	88%
Zahurul Islam Md et.al, 2009 [28]	Lightweight algorithm	find the root of the word	90.8%
Juhi Ameta et.al, 2012 [29]	Lightweight algorithm	find the root of the word	91.5%
Suba et.al, 2011 [31]	Lightweight stemmer	find the root of the word	70.70%
Vijay Sundar et.al 2012 [30]	Morphological analyzer	Derivational form of a word	91.7%

III. CONCLUSIONS

Stemming plays an important role in information retrieval system and its impact is very large, compared with that found in the review of the various stemming algorithms. In this paper, we studied various stemming algorithms and their effectiveness in various Indian languages. This is not enough for an information retrieval system. So that in future, researchers will opt for more implementations of stemming algorithm method and their utilities for various information retrieval systems in Indian languages.

IV. REFERENCES

[1] P. Rana, "Stemming of Punjabi Words By Using Brute Force Technique," *Int. J. Eng. Sci.*, vol. 3, no. 2, pp. 1351–1358, 2011.

[2] V. Gupta and G. S. Lehal, "Punjabi language stemmer for nouns and proper names," *Proc. 2nd Work. South* Southeast Asian Nat. Lang. Process. (WSSANLP), IJCNLP 2011, pp. 35–39, 2011.

[3] J. B. Lovins, "Development of a stemming algorithm," Mech. Transl. Comput. Linguist., vol. 11, no. June, pp. 22–31, 1968.

[4] M. F. Porter, "An algorithm for suffix stripping," Program, vol. 14, no. 3. pp. 130–137, 1980.

[5] D. Kumar and P. Rana, "Design and Development of a Stemmer for Punjabi," Int. J. Comput. Appl., vol. 11, no. 12, pp. 18–23, 2010.

[6] Jasmeet Singh and V. Gupta, "Text Stemming: Approaches, Applications, and Challenges," ACM Comput. Surv. Vol. 49, No. 3, Article 45 pp. 1-46, 2016.

[7] J. Patel, P. Desai, and U. Bhagat, "A survey of different stemming algorithm," Int. J. Adv. Eng. Res. Dev., vol. 2, no. 6, pp. 1083–1088, 2015.

[8] Tom´aˇs Brychc´ın and Miloslav Konop´ık, "High precision stemmer," Inf. Process. Manag. 51, 1, pp. 68–91, 2015.

[9] Robert Krovetz, "Viewing morphology as an inference process," In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 191–202, 1993.

[10] JiaulH. Paik, Mandar Mitra, Swapan K. Parui, and Kalervo Jarvelin, "An effective and efficient stemming algorithm for information retrieval," ACM Trans. Inf. Syst. 29, 2011.

[11] Jiaul H. Paik, Dipasree Pal, and Swapan K. Parui, "A novel corpus-based stemming algorithm using co-occurrence statistics," In Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'11). ACM, New York, NY, pp. 863–872, 2011.

[12] Jiaul H. Paik, Swapan K. Parui, Dipasree Pal, and Stephen E. Robertson, "Effective and robust querybased stemming," ACM Trans. Inf. Syst. 31, pp. 2013.

[13] Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, Gobinda Kole, Pabitra Mitra, and Kalyankumar Datta, "Yet another suffix stripper," ACM Trans. Inf. Syst. 25, 2007.

[14] JiaulH. Paik and Swapan K. Parui, "A Fast corpus-based stemmer," ACMTrans. Asian Lang. Inf. Process. 10, 2011.

[15] David Weiss, "A hybrid stemmer for the Polish language," Institute of Computing Science: Poznan University of Technology Research Report. 2005

[16] Manish Shrivastava, Bibhuti Mohapatra, Pushpak Bhattacharyya, Nitin Agarwal, and Smriti Singh, "Morphology based natural language processing tools for indian languages," In Proceedings of the 4th Annual Inter Research Student Seminar in Computer Science, 2005.

[17] Giorgos Adam, Konstantinos Asimakis, Christos Bouras, and Vassilis Poulopoulos, "An efficient mechanism for

stemming and tagging: the case of Greek language," In Proceedings of the 14th International, 2010.

[18] Pratikkumar Patel, Kashyap Popat, and Pushpak Bhattacharyya, "Hybrid stemmer for Gujarati," In Proceedings of the 23rd International Conference on Computational Linguistics (COLING), 51, 2010.

[19] Upendra Mishra and Chandra Prakash, "MAULIK: An effective stemmer for Hindi language" Int. J. Comput. Sci. Eng. 4, pp. 711–717, 2012.

[20] C. D. Paice, "An Evaluation Method for Stemming Algorithms", Proceedings of 17th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 42-50, 1994.

[21] X. Jinxi and C. Bruce W., "Corpus-based Stemming Using Co-occurrence of Word Variants", ACM Transactions on Information Systems, Volume 16, Issue 1, pp. 61-81, 1998.

[22] J. Mayfield and P. McNamee, "Single N-gram stemming", Proceedings of the 26th annual international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 415-416, 2003.

[23] M. Jenkins and D. Smith, "Conservative Stemming for Search and Indexing", In Proceedings of SIGIR'05, 2005.

[24] M. Massimo and O. Nicola. "A Novel Method for Stemmer Generation based on Hidden Markov Models", Proceedings of the twelfth international conference on Information and knowledge management, pp. 131-138, 2003.

[25] F. Peng, N. Ahmed, X. Li and Y. Lu, "Context Sensitive Stemming for Web Search", Proceedings of the 30th annual international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 639-646.

[26] A. Ramanathan and D. D. Rao, "A Lightweight Stemmer for Hindi", Workshop on Computational Linguistics for South-Asian Languages, EACL, 2003.

[27] S. Dasgupta and V. Ng, "Unsupervised Morphological Parsing of Bengali", Language Resources and Evaluation, 40(3-4):311-330, 2006.

[28] Khan. 2007. "A light weight stemmer for Bengali and its Use in spelling Checker," Proc. 1st Intl. Conf. on Digital Comm. and Computer Applications (DCCA07), Irbid, Jordan, March 19-23.

[29] Juhi Ameta, Nisheeth Joshi and Iti Mathur, 2011, "A Lightweight Stemmer for Gujarati," 46th Annual National Convention of Computer Society of India. Organized by Computer Society of India Gujarat Chapter. Sponsored by Computer Society of India and Department of Science and Technology, Govt. of Gujarat and IEEE Gujarat Section.

[30] Vijay Sundar et.al, "Morphological Analyzer for Classical Tamil Texts," Workshop on Computational Linguistics for South-Asian Languages, 2012.

[31] K. Suba, D. Jiandani and P. Bhattacharyya, "Hybrid Inflectional Stemmer and Rule-based Derivational Stemmer for Gujarati", In proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), IJCNLP 2011, Chiang Mai, Thailand, pp.1-8, 2011

