A Survey of Common Stemming Techniques and Existing Stemmers for Indian Languages

Vishal Gupta
UIET, Panjab University, Chandigarh, India
Email: vishal@pu.ac.in

Gurpreet Singh Lehal
Department of Computer Science, Punjabi University, Patiala, India
Email: gslehal@gmail.com

***Abstract*—Stemming is an operation that relates morphological variants of a word. The purpose of stemming is to obtain the stem or radix of those words which are not found in dictionary. If stemmed word is present in dictionary, then that is a genuine word, otherwise it may be proper name or some invalid word. Stemming is the process for reducing inflected or sometimes derived words to their stem, base or root form, generally a written word form. The stem need not be identical to the morphological root of the word, it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Stemming is used in Information Retrieval systems to improve performance. The design of stemmers is language specific, and requires some to significant linguistic expertise in the language, as well as the understanding of the needs for a spelling checker for that language. A stemmer's performance and effectiveness in applications such as spelling checker vary across languages. A typical simple stemmer algorithm involves removing suffixes using a list of frequent suffixes, while a more complex one would use morphological knowledge to derive a stem from the words. In this paper a survey of common stemming techniques and existing stemmers for Indian languages have been presented.**

***Index Terms*—stemmer, stemming techniques, Indian stemmers, suffix removal**

I. Introduction

Stemming is an operation that relates morphological variants of a word. The term 'conflation' is used to denote the act of mapping variants of a word to a single term or 'stem'. Stemming is used in Information Retrieval systems to improve performance. For example, when a user enters the query word stemming, he most likely wants to retrieve documents containing the terms stemmer and stemmed as well. Thus, using a stemmer improves recall, i.e., the number of documents retrieved in response to a query. Also, since many terms are mapped to one, stemming serves to decrease the size of the index files in the IR system.

The purpose of stemming is to obtain the stem or radix of those words which are not found in dictionary. If stemmed word is present in dictionary, then that is a genuine word, otherwise it may be proper name or some invalid word. stemming is the process for reducing inflected or sometimes derived words to their stem, base or root form, generally a written word form. The stem need not be identical to the morphological root of the word, it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. A stemmer for English, for example, should identify the string cats and possibly catlike, catty etc. as based on the root cat, and stemmer, stemming, stemmed as based on stem. A stemming algorithm reduces the words fishing, fished, fish, and fisher to the root word, fish. Stemming is an operation that conflates morphologically similar terms into a single term without doing complete morphological analysis.

Stemming [1] is used in information retrieval systems to improve performance. Additionally, this operation reduces the number of terms in the information retrieval system, thus decreasing the size of the index files. The design of stemmers is language specific, and requires some to significant linguistic expertise in the language, as well as the understanding of the needs for a spelling checker for that language. A stemmer's performance and effectiveness in applications such as spelling checker vary across languages. A typical simple stemmer algorithm involves removing suffixes using a list of frequent suffixes, while a more complex one would use morphological knowledge to derive a stem from the words. Not much work has been reported for stemming for Indian languages compared to English and other European languages.

A. Common Stemming Techniques

Porter (1980) proposed an algorithm for suffix stripping, is perhaps the most widely used algorithm for English stemming. Removing suffixes by automatic means is an operation which is especially useful in the field of information retrieval. It is rule based and is best

doi:10.4304/jetwi.5.2.157-161

suited for less inflectional languages like English. The suffix stripping process reduces the total number of terms in the IR system, and hence reduces the size and complexity of the data in the system, which is always advantageous. The algorithm does not remove a suffix when the stem is too short. The length of the stem is given by its measure, m. There is no linguistic basis for this approach. The resulting vocabulary of stems contained 6370 distinct entries. Usually the suffix stripping process reduces the size of the vocabulary by about one third. [2]

Paice (1994) proposed an evaluation method for stemming algorithms. This method outlines an approach to stemmer evaluation which is based on detecting and counting the actual under and over stemming errors committed during stemming of word samples derived from actual texts. This permits the computation of a 'stemming weight' index for each stemmer, as well as indices representing the under and over stemming error rates and the general accuracy. The method involves manually dividing a sample of words into conceptual groups, and referring the actual stemming performance to these groups. Though not used for stemming in real systems, the algorithm provides a good baseline for other stemming algorithms evaluation. [3]

Jenkins and Smith (2005) proposed conservative stemming for search and indexing This stemmer is designed to stem conservatively to orthographically correct word forms and recognizing words which do not need to be stemmed, such as proper nouns. Similarly to other stemmers, it operates on a set of rules which are used as steps. There are two groups of rules: the first group is to clean the tokens, and the second to alter suffixes. The first group of rules first avoids a small list of six frequent problem words. Second, possessive apostrophes are removed and contractions are expanded. All hyphens are removed and tokens containing digits are left untouched. Strings which are all upper case and digits are left untouched unless there is a lower case terminal 's' (i.e. transforming plural forms of acronyms to singular forms). Proper nouns should not usually be stemmed, except to remove possessives. If the text is untagged the stemmer uses the simple heuristic that any capitalized token not preceded by sentence breaking punctuation is a proper noun. The second group of rules contains 139 suffix rules, each testing for a specific type of suffix. The rules are set in a particular order so that the longest suffix applicable is used rather a shorter one which could lead to nonsense words and more words not stemmed entirely to their root form. [4]

Paice (1990) proposed another stemmer which is an iterative algorithm with one table containing about 120 rules indexed by the last letter of a suffix. On each iteration, it tries to find an applicable rule by the last character of the word. If there is no such rule, it terminates. It also terminates if a word starts with a vowel and there are only two letters left or if a word starts with a consonant and there are only three characters left. Otherwise, the rule is applied and the process repeats. [5]

John (1974) proposed suffix removal and word conflation which follows the longest match process and has perhaps the most comprehensive list of English suffixes (along with transformation rules) – about 1200 entries. The suffixes are stored in the reversed order indexed by their length and last letter. The rules define if a suffix found can be removed (for example, if the remaining part of the word is not shorter than N symbols; or if the suffix is preceded by a particular sequence of characters). It seems that the algorithm didn't gain popularity due to its complexity and lack of a standard reusable implementation. [6]

Mayfield and McNamee (2003) proposed single N-gram stemming which demonstrates that selection of a single n-gram as a pseudo-stem for a word can be an effective and efficient language-neutral approach for some languages The idea is to analyze distribution of all N-grams in a document (with some rather high value for N like 4 or 5, selected empirically). Since morphological invariants (unique word roots) will occur less frequently than variate parts (common prefixes and suffixes, for example, "ing" or "able"), a typical statistics like inverse document frequency (IDF) can be used to identify them. [7]

Massimo and Nicola (2003) proposed a novel statistical method for stemmer generation based on hidden Markov models. It doesn't need a prior linguistic knowledge or a manually created training set. Instead it uses unsupervised training which can be performed at indexing time. HMMs are finite-state automata with transitions defined by probability functions. Since probability of each path can be computed, it is possible to find the most probable path in the automata graph. Each character comprising a word is considered as a state. The authors divided all possible states into two groups (roots and suffixes) and two categories: initial (which can be roots only) and final (roots or suffixes). Transitions between states define word building process. For any given word, the most probable path from initial to final states will produce the split point (a transition from roots to suffixes). Then the sequence of characters before this point can be considered as a stem. The authors considered three different topologies of HMM in their experiments. Using Porter's algorithm as a baseline, they found that HMM had a tendency to over stem the words. [8]

Xu and Croft (1998) proposed an approach, which allows correcting "rude" stemming results based on the statistical properties of a corpus used. The basic idea is to generate equivalence classes for words with a classical stemmer and then "separate back" some conflated words based on their co-occurrence in the corpora. It also helps preventing well-known incorrect conflations of Porter's algorithm, such as "policy/police" since chances of these two words co-occurrence are rather low. Using Porter's

and trigram matching algorithms on three English corpora and one Spanish corpus, the authors showed significant improvement in retrieval efficiency (though it should be noted that separating conflated entries back almost canceled the results of stemming). [9]

Peng et al. (2007) suggested context sensitive stemming for web search [68]. In their work corpus analysis is used to find word distributional similarity. Then a few morphological rules from Porter's stemmer are applied to the similarity list to find stemming candidates, some of which are finally selected based on the handling purpose, for example, pluralization. Obtained forms are used to expand a search query on non-transformed index. For example, considering word "develop", Applying stemming rules retains "developing, developed, develops, development, development" and for Pluralization purposes only "develops" is selected. Hence, the user's query "develop" is expanded to "develop OR develops". [10]

Goldsmith (2001) proposed an algorithm for the morphology of a language based on the minimum description length (MDL) framework which focuses on representing the data in as compact manner as possible. This study reports the results of using minimum description length (MDL) analysis to model unsupervised learning of the morphological segmentation of European languages, using corpora ranging in size from 5,000 words to 500,000 words. A set of heuristics are proposed that rapidly develop a probabilistic morphological grammar, and use MDL as primary tool to determine whether the modifications proposed by the heuristics will be adopted or not. The resulting grammar matches well the analysis that would be developed by a human morphologist. [11]

Creutz and lagus (2005) used probabilistic maximum a posteriori (MAP) formulation for morpheme segmentation and described the first public version of the Morfessor software, which is a program that takes as input a corpus of unannotated text and produces a segmentation of the word forms observed in the text. The segmentation obtained often resembles a linguistic morpheme segmentation. Morfessor is not language-dependent. The number of segments per word is not restricted to two or three as in some other existing morphology learning models. [12]

B. Existing Stemmers for Indian Languages

Not much work has been reported for stemming for Indian languages compared to English and other European languages. Ramanathan and Rao (2003) proposed a lightweight stemmer for Hindi which has used a hand crafted suffix list and has performed longest match stripping. Light stemming refers to stripping of a small set of either prefixes or suffixes or both, without trying to deal with infixes, or recognize patterns and find roots. This lightweight stemmer proposed for Hindi is based on the grammar for Hindi language in which a list of total 65 suffixes is generated manually. Terms are conflated by stripping off word endings from a suffix list on a `longest match' basis. Noun, adjective and verb infections have been discussed and based on that 65 unique suffixes are collected. The major advantage of this approach is as it is computationally inexpensive. Documents were chosen from varied domains such as Films, Health, Business, Sports and Politics. The collection contained 35977 unique words. Under stemming and over stemming errors calculated in this methodology were 4.68% and 13.84% respectively. No recall/precision-based evaluation of the work has been reported; thus the effectiveness of this stemming procedure is difficult to estimate. [13]

Islam et al. (2007) proposed a light weight stemmer for Bengali and its use in spelling checker with similar approach as proposed by Ramanathan and Rao (2003) [13]. The proposed algorithm strips the suffixes using a predetermined suffix list, also on a `longest match' basis. A total of 72 suffixes for verbs, 22 for nouns and just 8 for adjectives for Bengali language have been found. The proposed stemming algorithm is primarily for handling inflections – it does not handle derivational suffixes, for which one would need a proper morphological analyzer. Reducing derivationally related terms to the same stem would lead to overconflation in some cases, potentially affecting the precision of information retrieval applications, other than spelling checkers. [14]

Majumder et al. (2007) developed statistical approach YASS: Yet Another Suffix Stripper, which uses a clustering based approach based on string distance measures and requires no linguistic knowledge. They concluded that stemming improves recall of IR systems for Indian languages like Bengali. YASS is based on string distance measure which is used to cluster a lexicon created from a text corpus into homogenous groups. Each group is expected to represent an equivalence class consisting of morphological variants of the single root word. Graph-theoretic clustering algorithm which require a threshold Θ which was used as a parameter in the experiments. [15]

Dasgupta and Ng (2006) proposed unsupervised morphological parsing of Bengali. Unsupervised morphological analysis is the task of segmenting words into prefixes, suffixes and stems without prior knowledge of language-specific morphotactics and morpho-phonological rules. This parser is composed of two steps: (1) inducing prefixes, suffixes and roots from a vocabulary consisting of words taken from a large, unannotated corpus, and (2) segmenting a word based on these induced morphemes. When evaluated on a set of 4,110 human-segmented Bengali words, our algorithm achieves an F-score of 83%, substantially outperforming Linguistica, one of the most widely-used unsupervised morphological parsers, by about 23%. [16]

Pandey and Siddiqui (2008) [17] proposed an unsupervised stemming algorithm for Hindi based on

Goldsmith (2001) [69] approach. It is based on split-all method. For unsupervised learning (training), words from Hindi documents from EMILLE corpus have been extracted. These words have been split to give n-gram (n=1, 2, 3 … *l*) suffix, where *l* is length of the word. Then suffix and stem probabilities are computed. These probabilities are multiplied to give split probability. The optimal segment corresponds to maximum split probability. Some post-processing steps have been taken to refine the learned suffixes. It is evaluated on 1000-1000 words randomly extracted words (only) from Hindi WordNet1 data base. The training data has been constructed by extracting 106403 words extracted from EMILLE2 corpus. The observed accuracy is 89.9% after applying some heuristic measures. The F-score is 94.96%. The algorithm does not require any language specific information. [17]

Majgaonker and Siddiqui (2010) developed an unsupervised approach for Marathi stemmer. Three different approaches (rule based, suffix stripping and statistical stripping) for suffix rules generation has been used in unsupervised stemmer. The rule-based stemmer uses a set of manually extracted suffix stripping rules whereas the unsupervised approach learns suffixes automatically from a set of words extracted from raw Marathi text. The performance of both the stemmers has been compared on a test dataset consisting of 1500 manually stemmed word. The maximum accuracy observed is 82.5% for the statistical suffix stripping approach. This approach uses a set of words to learn suffixes. [18]

Suba et al. (2011) proposed two stemmers for Gujarati- a lightweight inflectional stemmer based on a hybrid approach and a heavyweight derivational stemmer based on a rule-based approach. The inflectional stemmer has an average accuracy of about 90.7% which is considerable as far as IR is concerned. Boost in accuracy due to POS based stemming was 9.6% and due to inclusion of the language characteristics it was further boosted by 12.7%. The derivational stemmer has an average accuracy of 70.7% which can act as a good baseline and can be useful in tasks such as dictionary search or data compression. The limitations of inflectional stemmer can be easily overcome if modules like Named Entity Recognizer are integrated with the system. [19]

Regarding Punjabi language, A stemmer for Punjabi nouns and proper names had been developed by Gupta and Lehal (2011) in which an attempt was made to obtain stem or radix of a Punjabi word and then stem or radix was checked against Punjabi noun morph and proper names list. From Punjabi news corpus various possible noun suffixes were identified like ੀਆਂ *īāṃ*, ਿਆਂ *iāṃ*, ੂਆਂ *ūāṃ*, ਾਂ *āṃ*, ੀਏ *īē* etc. and various stemming rules for nouns and proper names were generated. The efficiency of Punjabi language noun and Proper name stemmer is 87.37%, which is tested over 50 news documents of Punjabi news corpus containing 11.29 million words.

REFERENCES

[1] H. Harmani, Walid Keirouz and Saeed Raheel, "A rule base extensible stemmer for Information retrieval with application to Arabic", *The International Arab Journal of Information Technology*, Vol. No.3, Issue No.3, pp 265-272, 2006.

[2] M. Porter, "An Algorithm for Suffix Stripping Program", 14(3): 130-137, 1980.

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

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

[5] C. D. Paice, "Another stemmer". *ACM SIGIR Forum*, Volume 24, No. 3, 56-61, 1990.

[6] D. John, "Suffix removal and word conflation", *ALLC Bulletin*, Volume 2, No. 3, 33-46, 1974.

[7] 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*, 415-416, 2003.

[8] 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*, 131-138, 2003.

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

[10] 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*, 639-646, 2007.

[11] J. A. Goldsmith, "Unsupervised Learning of the Morphology of a Natural Language", *Computational Linguistics*, MIT Press, 27(2):153-198, 2001.

[12] M. Creutz and K. Lagus, "Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora using Morfessor 1.0.", *Technical Report A81*, Publications in Computer and Information Science, Helsinki University of Technology, 2005.

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

[14] M. Z. Islam, M. N. Uddin and M. Khan, "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 2007.

[15] P. Majumder, M. Mitra, S. K. Parui, G. Kole, P. Mitra, and K. Datta, "YASS: Yet Another Suffix Stripper", *Association for Computing Machinery Transactions on Information Systems*, 25(4):18-38, 2007.

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

[17] A. K. Pandey and T. J. Siddiqui, "An Unsupervised Hindi Stemmer with Heuristic Improvements", *In Proceedings of the Second Workshop on Analytics For Noisy Unstructured Text Data*, 303:99-105, 2008.

[18] M. M. Majgaonker and T. J Siddiqui, "Discovering Suffixes: A Case Study for Marathi Language",

International Journal on Computer Science and Engineering, Vol. 02, No. 08, pp. 2716-2720, 2010.

[19] 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.

[20] V. Gupta and G. S. Lehal, "Punjabi Language Stemmer for Nouns and Proper Names", *Proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), IJCNLP 2011*, Chiang Mai, Thailand, pp. 35–39, 2011.

[21] V. Gupta and G. S. Lehal, "Preprocessing Phase of Punjabi Language Text Summarization", *International Conference on Information Systems for Indian Languages Communications in Computer and Information Science ICISIL2011*, Volume 139, Part 2, Springer-Verlag Berlin Heidelberg, pp. 250-253, 2011.

AUTHORS' INFORMATION

Vishal Gupta is Assistant Professor in Computer Science & Engineering department at University Institute of Engineering & Technology, Panjab University Chandigarh. He has done MTech. in computer science & engineering from Punjabi University Patiala in 2005. He is among University toppers. He secured 82% Marks in MTech. Vishal did his BTech. in CSE from Govt. Engineering College Ferozepur in 2003. He is also pursuing his PhD in Computer Science & Engineering. Vishal has written around thirty five research papers in international and national journals and conferences. He has developed a number of research projects in field of NLP including synonyms detection, automatic question answering and text summarization etc. One of his research paper on Punjabi language text processing was awarded as best research paper by Dr. V. Raja Raman at an International Conference at Panipat. He is also a merit holder in 10^{th} and 12^{th} classes of Punjab School education board.

Professor Gurpreet Singh Lehal received undergraduate degree in Mathematics in 1988 from Panjab University, Chandigarh, India, and Post Graduate degree in Computer Science in 1995 from Thapar Institute of Engineering & Technology, Patiala, India and Ph. D. degree in Computer Science from Punjabi University, Patiala, in 2002. He joined Thapar Corporate R&D Centre, Patiala, India, in 1988 and later in 1995 he joined Department of Computer Science at Punjabi University, Patiala. He is actively involved both in teaching and research. His current areas of research are- Natural Language Processing and Optical Character recognition. He has published more than 25 research papers in various international and national journals and refereed conferences. He has been actively involved in technical development of Punjabi and has to his credit the first Gurmukhi OCR, Punjabi word processor with spell checker and various transliteration software. He was the chief coordinator of the project "Resource Centre for Indian Language Technology Solutions- Punjabi", funded by the Ministry of Information Technology as well as the coordinator of the Special Assistance Programme (SAP-DRS) of the University Grants Commission (UGC), India. He was also awarded a research project by the International Development Research Centre (IDRC) Canada for Shahmukhi to Gurmukhi Transliteration Solution for Networking.