Volume 6, Issue 2, February 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Applications of Stemming Algorithms in Information Retrieval- A Review

Rakesh Kumar[*], Vibhakar Mansotra
Department of Computer Science & IT, University of Jammu,
India

Abstract— Stemming is process that provides mapping of related morphological variants of words to a common stem /root form. The main purpose of stemming is to get root word of those words that are not present in dictionary/Wordnet. Stemming is very important approach for those languages that are rich in morphology. It has many application in NLP and Information Retrieval. In Information Retrieval systems stemming improves performance in terms of recall and precision. It also reduces the size of index file during indexing by conflating morphological variant to a common term/stem. In this paper different stemming Algorithms for Information retrieval and its applications in IR have been presented.

Keywords— Stemming, Information Retrieval(IR), Corpus, Morphology

I. INTRODUCTION

Stemming is process that provides mapping of different related morphological variants of words into their base or common form. The main purpose of stemming is to obtain the stem or radix of those word that are not found in the dictionary/ Wordnet. Almost all natural languages are morphologically inflected. In these languages several words that are derived from same root can occur in a document. So stemming can be used to conflate all these words that are inflected or derived from the same root to its base form(stem). This application of stemmer can be utilized efficiently in Information Retrieval System for improving performance by increasing recall and precision [1]. Stemming also helpful in NLP tools like Spellchecker, Machine translation. There are mainly two errors in stemming i.e. Under stemming and Over stemming. Under stemming occurs when having same root but does not conflates to same root. This is termed as a false negative. Over stemming is when two words having different stems are stemmed to the same root. This is also termed as a false positive [2]. In the next sessions we will discuss different approaches/techniques of stemming in brief and related work employing these techniques.

II. STEMMING APPROACHES

In stemming, different variants of word stemmed to a common root. Various Stemming Approaches used are shown in figure 1. We can do stemming either manually with the help of regular expressions or automatic . Automatic technique branched into four categories like successor, affix removal, n-gram and table lookup. Affix removal further divided into longest match and simple removal methods[4].

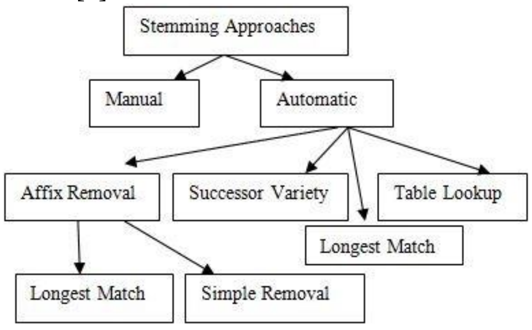

Fig.1 Stemming Approaches

A. Affix Removal Method

This method removes prefix or suffix from a word to produce a common word. These algorithms works on two principles- first one is iterations: starting at the end of a word and going towards beginning of word by removing strings against each order class one at a time, only one match is allowed in a single order class. The second one is longest match in which within any given class of endings, longest match is eliminated if more than one ending gives a match [4].

B. Successor Variety Method

In successor variety method [5], determination of word and sequences in a body of text as the basis of stemming. The successor variety of a string is the number of different characters that follows it in word in some body of text. When this process is applied on a large body of text the successor variety of the substring of term will reduces as more character are added until a segment boundary is reached. So this idea is used to get the stem.

C. Table Lookup Method

This method works by looking at the table where words and their morphological variants are stored. Words from queries and indexes are searched in lookup table and a corresponding root is returned[6]. We prefer to use B-tree or hash table lookup for fast searching, but there is always a storage overhead for such table.

D. N-gram Method

Adamson and Boreha [7] gives a method in 1974 to stem terms known as shared digram. The digram is a pair of two consecutive letters. We can also use trigrams and hence it is called n-gram method [8]. With this approach, an association measures are calculated between pairs terms based on shared unique digrams. Example for word Numeric and numerical

Numeric:	nu um me er ri ic
unique digrams:	nu um me er ri ic
Numerical:	nu um me er ri ic ca al
unique digrams:	nu um me er ri ic ca al

Similarity= 2C/(A+B) =2*6/(6+8)=0.85

we use determines Dice's coefficient [4] to find the similarity between a word and its root word. Here in the above example

A denotes unique digram of word numeric and B denotes unique digrams of word 'numerical' and C denotes unique digrams shared by both.

III. CLASSIFICATION OF STEMMERS

The simplest technique to design a stemmer involves removing suffixes by using a list of frequent suffixes, and more complicated technique uses the morphological structure of the words to derive a stem. All these techniques differ from other in terms of accuracy and performance. Broadly, stemming algorithms can be classified in three groups: truncating methods, statistical methods, and mixed methods[2].

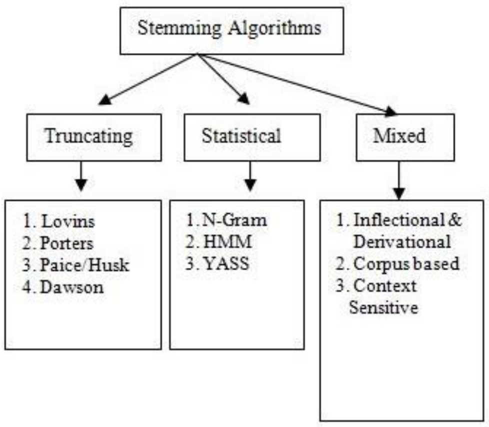

Fig. 2 Classification of Stemmers

A. Truncating Methods(Affix Removal Methods)

As the name clearly suggests these methods are related to removing the suffixes or prefixes of a word. Most famous stemming algorithms that employ this methods are as Lovins, Porter[9,10].

Lovins(1968) developed the first stemmer, it performs a lookup on a table of 294 endings, 29 conditions and 35 transformation rules, which have been arranged on a longest match principle. The Lovins stemmer removes the longest suffix from a word and once the ending is removed, the word is recoded using a different table that makes various adjustments to convert these stems into valid words. It always removes a maximum of one suffix from a word, due to its nature as single pass algorithm. The advantages of this algorithm is, it is very fast and can handle removal of double letters in words like 'getting' being transformed to 'get' and also handles many irregular plurals like – mouse and mice, index and indices etc. Drawbacks of the Lovins approach are that it is time and data consuming. Furthermore, many suffixes are not available in the table of endings. It is sometimes highly unreliable and frequently fails to form words from the stems or to match the stems of like-meaning words. The reason being the technical vocabulary being used by the author[9].

Porter (1980) proposed an algorithm for suffix stripping and is perhaps the most widely used algorithm for English stemming for removing suffixes by automatic means. It involves an operation which is especially useful in the field of information retrieval and is best 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 and there is no linguistic basis for this approach. The resulting vocabulary of stems contained 6370 distinct entries and is usually the suffix stripping process reduces the size of the vocabulary by about one third. [10]
Paice (1990) proposed 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 and 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. [11]
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. This method involves manually dividing a sample of words into conceptual groups and referring the actual stemming performance to these groups. Though it is not used for stemming in real systems but this algorithm provides a good baseline for other stemming algorithms evaluation[12] .
Jenkins and Smith (2005) proposed conservative stemming for search and indexing and the 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. 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 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 in which 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 than a shorter one which could lead to nonsense words and more words not stemmed entirely to their root form[13] .
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). 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[14].

B. Statistical Methods

These are the stemmers which are based on statistical analysis and techniques. Most of the methods remove the affixes but after implementing some statistical procedure. List of some stemmers employing statistical methods are: N-Gram Stemmer, HMM Stemmer, YASS Stemmer.
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 [15].
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[16].
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 Ө which was used as a parameter in the experiments[17].

C. Inflectional and Derivational Methods

This is another approach to stemming and it involves both the inflectional as well as the derivational morphology analysis. The corpus should be very large to develop these types of stemmers and hence they are part of corpus base stemmers too. In case of inflectional the word variants are related to the language specific syntactic variations like plural, gender, case, etc whereas in derivational the word variants are related to the part-of-speech (POS) of a sentence where the word occurs.

Krovetz (1993) by Robert Krovetz [18] and is a linguistic lexical validation stemmer. Since it is based on the inflectional property of words and the language syntax, it is very complicated in nature. It effectively and accurately removes inflectional suffixes in three steps:

1. Transforming the plurals of a word to its singular form
2. Converting the past tense of a word to its present tense
3. Removing the suffix 'ing'

The conversion process first removes the suffix and then through the process of checking in a dictionary for any recoding, returns the stem to a word. The dictionary lookup also performs any transformations that are required due to spelling exception and also converts any stem produced into a real word, whose meaning can be understood. The strength of derivational and inflectional analysis is in their ability to produce morphologically correct stems, cope with exceptions, processing prefixes as well as suffixes. Since this stemmer does not find the stems for all word variants, it can be used as a pre- stemmer before actually applying a stemming algorithm. This would increase the speed and effectiveness of the main stemmer. Compared to Porter and Paice / Husk, this is a very light stemmer. The Krovetz stemmer attempts to increase accuracy and robustness by treating spelling errors and meaningless stems.

If the input document size is large this stemmer becomes weak and does not perform very effectively. The major and obvious flaw in dictionary-based algorithms is their inability to cope with words, which are not in the lexicon. Also, a lexicon must be manually created in advance, which requires significant efforts. This stemmer does not consistently produce a good recall and precision performance.

Xerox Inflectional and Derivational Analyzer [19]

The linguistics groups at Xerox have developed a number of linguistic tools for English which can be used in information retrieval. In particular, they have produced English lexical database which provides a morphological analysis of any word in the lexicon and identifies the base form. Xerox linguists have developed a lexical database for English and some other languages also which can analyze and generate inflectional and derivational morphology. The inflectional database reduces each surface word to the form which can be found in the dictionary, as follows [19]:

nouns singular (e.g. children child) verbs infinitive (e.g. understood understand) adjectives positive form (e.g. best good) pronoun nominative (e.g. whom who) semantics.

For example, 'government' stems to 'govern' while 'department' is not reduced to 'depart' since the two forms have different meanings. All stems are valid English terms, and irregular forms are handled correctly. The derivational process uses both suffix and prefix removal, unlike most conventional stemming algorithms which rely solely on suffix removal.

D. Corpus Based Stemmer

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)[20] .

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 [21].

D. Context Sensitive Stemmer

This is a very interesting method of stemming unlike the usual method where stemming is done before indexing a document, over here for a Web Search, context sensitive analysis is done using statistical modeling on the query side. This method was proposed by Funchun Peng et. al. [22].

Basically for the words of the input query, the morphological variants which would be useful for the search are predicted before the query is submitted to the search engine. This dramatically reduces the number of bad expansions, which in turn reduces the cost of additional computation and improves the precision at the same time.

After the predicted word variants from the query have been derived, a context sensitive document matching is done for these variants. This conservative strategy serves as a safeguard against spurious stemming, and it turns out to be very important for improving precision.

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" [23] .

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 [24] .The advantage of this stemmer is it improves selective word expansion on the query side and conservative word occurrence matching on the document side. The disadvantage is the processing time and the complex nature of the stemmer. There can be errors in finding the noun phrases in the query and the proximity words.

IV. CONCLUSIONS

The algorithms we have discussed so far, it can be inferred that there is a lot of similarity between the stemming algorithms. Every algorithm has its negatives and positives. However, none of them give 100% accuracy but we can use it in NLP or IR applications by changing suitable parameters as per requirements. In future, we will design a hybrid stemmer by employing these existing approaches as a part of Information Retrieval System for improving performance of the system in terms of recall and precision.

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] A. G. Jivani, "A Comparative Study of Stemming Algorithms", International Journal of Computer Technology and Applications, Vol.2 (6), PP 1930-1938, NOV-DEC 2011.

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

[4] WB Frakes,1992, "Stemming Algorithm", "Information Retrieval Data Structures and Algorithm", Ist Edition, Prentice Hall,1992 ,page 132-139.

[5] M. Hafer and S. Weiss 1974. "Word Segmentation by Letter Successor Varieties," Information Storage and Retrieval, volume 10, Issues 11-12, pp 371-85,1974.

[6] Wessel Kraaij and Renee Pohlmann, Porter's stemming algorithm for Dutch, UPLIFT (Utrecht Project: Linguistic Information for Free Text retrieval) is sponsored by the NBBI,Philips Research, the Foundation for Language Technology.

[7] G. Adamson and J. Boreham 1974. "The Use of an Association Measure Based on Character Structure to Identify Semantically Related Pairs of Words and Document Titles," Information Storage and Retrieval, 10,253-60.

[8] 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

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

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

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

[12] 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.

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

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

[15] 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.

[16 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.

[17] 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, Volume 25 Issue 4 :18-38, 2007.

[18] Krovetz Robert. "Viewing morphology as an inference process". Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval. 1993, 191-202.

[19] Hull D. A. and Grefenstette,. " A detailed analysis of English Stemming Algorithms", XEROX Technical Report, http://www.xrce.xerox.

[20] 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.

[21] 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.

[22] Funchun Peng, Nawaaz Ahmed, Xin Li and Yumao Lu. “Context sensitive stemming for web search”. Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. 2007, 639-646.

[23] 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.

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