CS7030: Assignment 1 – Stemming Algorithms

Introduction:

Stemming is the process of conflating morphologically similar terms into a single term without actually caring about morphology. In informational retrieval systems this is especially useful since it will help to build a more generalised index for terms.

The errors in case of stemming process are either over stemming – wherein the resultant stem may be so small in length that it loses its significance, e.g., nation[al] → nat (*removed even -ion*)

On the other hand it could be under-stemming wherein the resultant stem may still be left with a suffix or letters that prevent it from being a good root, e.g.,[un]national[ist][ic] → unnationalist

Lovins Stemmer:

Proposed by Julie Beth Lovins in 1968 [1], this was the first stemming algorithm for any language. It is classified as a table lookup approach since it uses large vocabulary set which was mostly technical in the original version. It consists of 294 endings, 29 conditions and 35 transformation rules [2].

The algorithm has 2 major steps: in the first step the longest ending is identified and removed. In the second step, 35 rules are used to transform the ending. Each rule is accompanied by a stem length condition. The rules are sometimes described to be used in stages, wherein Rule 1 is used in stage 1 and remaining rules used in stage 2 [3].

Although it is much bigger in terms of the number of rules, it results in much better data reduction. Since it has an extensive predefined endings' list, it is much better in terms of speed.

Examples:-

Rub[ing] → rub, embed[ed] → embed

Believ[e] → belief

Absorpt[ion] → absorb, etc.

Porter stemmer:

The original Porter algorithm [4] was given by Martin Porter, in the year 1980. It has 60 suffixes, 2 rules for transformation and a context based rule to determine whether to remove a suffix. The algorithm works as a 5-step procedure wherein the first step removes inflectional suffixes and the second through fourth remove derivational suffixes. The fifth step performs recoding. This algorithm is classified as an affix stripping approach.

The original Porter stemming algorithm was found [4, 6] to be at least as effective as others but more complicated. Porter Algorithm was found to make fewer errors than Lovin's Algorithm by Paice [7]. The algorithm has since been modified into the Porter2 or Snowball stemming algorithm [5] which is accompanied by a framework to work with non-english languages as well.

Examples:-

Reviv[e] → reviv

Allow[ance] → allow

Gyroscop[ic] → gyroscop

Comparison of stemming algorithms:

The strength of a stemming algorithm may be estimated by the over-stemming and under-stemming indices [7]. Porter stemmer was found to have a tendency of performing more under-stemming in comparison to other stemmers, whereas Lovins stemmer might perform over-stemming in certain cases. However, Lovins stemmer may have lower utility due to its dependence on a vocabulary set which is limited in its scope.

The following table serves to compare the two algorithms.

Stemmer	Advantages	Weakness
Lovins Stemmer	Fast	Not all suffixes are available
	Handles double letter words such as getting to get	
	Handles irregular plurals such as mice to mouse	
Porter Stemmer	Best output compared to other stemmers (Porter2)	Stems may not be valid words
	Low error rate	Time consuming
	Light stemmer (no table lookup)	

Stemming for Indic-Languages:

Amongst others, following are some properties that make working with Indic-Languages challenging [8].

1. Free word order: words in a sentence may change their order without affecting overall meaning, however in general they have the order - SOV
2. More than one suffix may be attached to a root. Number of prefixes is much less than number of suffixes.
3. Verb changes with person and TAM (tense, aspect and modality) and not with gender and number.

Accompanying this report is a source code for a simple rule based suffix stripping approach for Hindi language.

Bibliography:

[1] Popat, Pratikkumar Patel Kashyap, and Pushpak Bhattachatyya. “Hybrid stemmer for gujarati.” In 23rd International Conference on Computational Linguistics. p. 51. 2010.

[2] Lovins, Julie B. Development of a stemming algorithm. Cambridge: MIT Information Processing Group, Electronic Systems Laboratory, 1968

[3] Jivani, Anjali Ganesh. “A comparative study of stemming algorithms.” Int. J. Comp. Tech. Appl 2, no. 6 (2011): 1930-1938

[4] Porter, Martin F. “An algorithm for suffix stripping.” Program: Electronic Library Inform. Syst. 14, 3, p. 130-137. 1980

[5] Porter, Martin F. “Snowball: A language for stemming algorithms.” (2001)

[6] Lennon, Martin, David S. Pierce, Brian D. Tarry, and Peter Willett. “An evaluation of some conflation algorithms for information retrieval.” In Document retrieval systems, pp. 99-105. Taylor Graham Publishing, 1988

[7] Paice, Chris D. “An evaluation method for stemming algorithms.” In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 42-50. Springer-Verlag New York, Inc., 1994.

[8] Saharia, Navanath Sharma, Utpal and, Jugal Kalita. “Stemming Resource-Poor Indian Languages.” ACM Trans. Asian Lang. Inform. Process. 13, 3, Article 14 (2014)

[9] Ramanathan, A and, Durgesh D Rao. “A lightweight stemmer for Hindi.” (2003)