ISSN:0975-9646

Modified Porter Stemming Algorithm

Atharva Joshi[1], Nidhin Thomas[2], Megha Dabhade[3]

[1,2,3]*M.Tech, Department of Computer Science and Engineering*
Vellore Institute of Technology
Vellore, India

***Abstract*— Stemming is a critical component in the pre-processing stage of Text Mining. It is an inter-disciplinary process which finds applications in Natural Language Processing and Information Retrieval systems, especially search engines. It is used to relate index and search terms that are morphologically similar. It provides reduction in the size of indexing files and improved information retrieval effectiveness by increasing the recall rate and giving us the most relevant results. In stemming, different forms of a word like its noun, adjectives, verb, adverbs are reduced to its root form. There are several different approaches to stemming; table lookup, affix removal, successor variety, and n-gram, each having their own advantages and limitations. In this paper, we propose a modified version of the Porter stemmer in an effort to overcome some of its limitations and equip it with features that will make it more useful in information retrieval.**

Keywords*— Porter Stemmer, Stemming Algorithms, Text Mining, Information Retrieval, Text Pre-processing

I. INTRODUCTION

With the dawn of the digital age, there has been an exponential increase in data, the majority of which is in text form. This calls for sophisticated information retrieval and text mining algorithms to enable one to filter out the relevant information from the junk, within the large collection of available data. However, the most crucial factors in effective information retrieval are speed and precision (relevance). The huge demand for improvement in this area has driven the development of linguistic morphological techniques such as stemming and lemmatization.

Stemming, in its simplest definition is the process of reducing an inflated or derived word to its stem, root or base form by removing the attached affixes. [1][14] This process is also referred to as conflation. The stem is obtained after applying a set of rules on the word, as opposed to lemmatizing which deals with the complex process of dealing with the part of speech of a word and its context within the sentence to obtain the lemma. Stemming makes it possible to reduce words with same roots into a single stem, thereby drastically improving the effectiveness of information retrieval and text mining by reducing the indexing size by up to 40-50%. Two key points to be considered while implementing a stemming algorithm are:

1. Morphological variants of a word are presupposed to have similar meanings and should be mapped to the same stem
2. Words that that are etymologically similar but sharply differ in meaning should not be mapped to the same stem

Thus, stemming errors can be mainly classified into two. Under-stemming in which words that refer to the same concept are not reduced to the same stem and Over-stemming in which words are converted to the same stem even when they have distinctly different meanings. [2][3][9][11] A heavy stemming algorithm might aggressively pursue the removal of affixes, resulting in incorrect stems. On the other hand, a light stemming algorithm, in an attempt to avoid over-stemming might end up causing several under-stemming errors. Designing an efficient stemming algorithm, is often a question of finding the perfect balance between these two extremes.

II. RELATED WORK

Stemming algorithms follow several different approaches such as truncation, statistical methods and inflectional/derivational techniques, with several algorithms under each approach. [1]

Truncation Stemmers: They create a stem by removing the affixes associated with it.

1. The Porter Stemmer: It is one of the most commonly used truncation stemmers. It removes affixes from a word over a number of iterations until all the rules/conditions are considered. As it operates without a lexicon and does not consider word meanings, it is subject to certain errors. [4][5][10][15] Words with different meanings are reduced to the same stem (E.g.: “generic” and “generation” are stemmed to “gener”), while words with similar meanings may not be reduced to a common stem at all (E.g.: “recognition” and “recognize”). Besides, the produced stem may not be a valid word. In spite of these issues, the analysis of the Porter stemmer has shown that its performance is one of the best in terms of IR recall/precision.
2. Lovins Stemmer: The first popular and effective stemmer proposed by Lovins in 1968. 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. [6] It removes the longest suffix from a word, is very fast and can handle removal of double letters in words but it consumes a lot of time and is highly unreliable.
3. Paice/Husk Stemmer: It is an iterative algorithm with one table containing about 120 rules indexed by the last letter of a suffix. [8] It tries to find an applicable rule which specifies either a deletion or replacement of an ending over each iteration. If no such rule is found, it terminates. The advantage is its simple and every iteration taking care of both deletion and replacement as per the rule applied. But being a very heavy algorithm and over stemming may occur.

Statistical Stemmers: They remove affixes after subjecting it to some statistical procedure.

1. N-gram Stemmer: Words are conflated to their stem using a string-similarity approach. N-gram is a set of n consecutive characters extracted from a word. Words that are similar will have a high proportion of n-grams in common hence they can be converted to same stem. It is language independent and hence very useful in many applications but it requires a significant amount of memory and storage for creating and storing the n-grams and indexes. [1]
2. HMM Stemmer: This stemmer is based on the concept of the Hidden Markov Model (HMMs) [7] which are finite-state automata where transitions between states are ruled by probability functions. This method is based on unsupervised learning hence it is language independent. Disadvantage is complex and may over stem words.
3. YASS Stemmer: It stands for yet another Suffix Striper. This is a category of statistical as well as corpus based. The clusters are created using hierarchical approach and distance measures. [12] It can be used for any language without knowing its morphology. Difficult to decide threshold for creating clusters. Requires significant computing power.

Inflectional/Derivational Stemmers: Inflectional methods relate the different forms of words to their tense, gender, case etc., while derivational methods deals with relating the variations of words to the part-of-speech (POS) of a sentence where the word occurs.

1. The Krovetz or K-STEM algorithm: It addresses many of the problems with the Porter stemmer using a digital dictionary and well defined rules for inflectional and derivational morphology. As it heavily depends upon the contents of its dictionary, its conflation might end up being conservative. [13] The constant "look-up" it has to perform during the execution significantly slows down its speed, thereby debilitating retrieval performance.
2. Xerox Stemmer: The linguists at Xerox Corporation created a lexical database for English which helps to identify the base word using morphological analysis of the word in the lexicon. [5] It works well with a large documents and removes the prefixes with a valid stem. On the other hand, its dependence on the lexicon makes it a language dependent stemmer.
3. Corpus Based: Conflation classes are automatically modified to overcome problems in Porter algorithm. The significance of word form co-occurrence can be determined by the statistical measure [15] given by Em (a, b) = nab / (na + nb) where, a and b are a pair of words, na and nb are the number of occurrences of a and b in the corpus, nab is the number of times both a and b fall in a text window of size win in the corpus. This method can potentially avoid making conflations that are not appropriate for a given corpus and the result is a real word stem. It is complex and hence processing time increases.

III. METHODOLOGY

Porter stemming algorithm is based on the idea that the suffixes in the English language are mostly made up of a combination of smaller and simpler suffixes. It has six steps, and within each step, rules are applied until one of them passes the conditions. If a rule is accepted, the suffix is removed accordingly, and the next step is performed. The resultant stem at the end of the sixth step is returned. The main drawback of the Porter algorithm is that the resultant stems produced are not always real words.

Another stemming algorithm, the Krovetz stemmer tries to overcome this drawback in the following way. It first removes the suffix and then puts the word through a dictionary checking process for any potential recoding before returning the stem. 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. However, this process is tedious, consumes a lot of memory, creates unnecessary overheads and slows down the stemming process significantly. On the other hand, converting a stem to a real word does not have any significant real word advantages, besides making it easier for the user at the back end to get a better grasp of the stemming process.

A better approach would be to maintain a relationship table comprising of the inverse relation between stems and the input words. So each stem will return all the words which have resulted in that stem. Thus whenever the algorithm returns a stem that is vague or ambiguous, all one needs to do is look it up with the original words from which the stem has been derived. This eliminates the need of storing thousands of words in a dictionary file and comparing the stem with each of those. Our method has substantial advantages in terms of speed and memory requirement compared to the dictionary look-up method. Additionally, our algorithm will also return the word count of the input words, which can be used in the ranking process (Term Frequency [TF]/Inverse Document Frequency [IDF]) of search engines.

Shown below is the traditional Porter stemming algorithm:

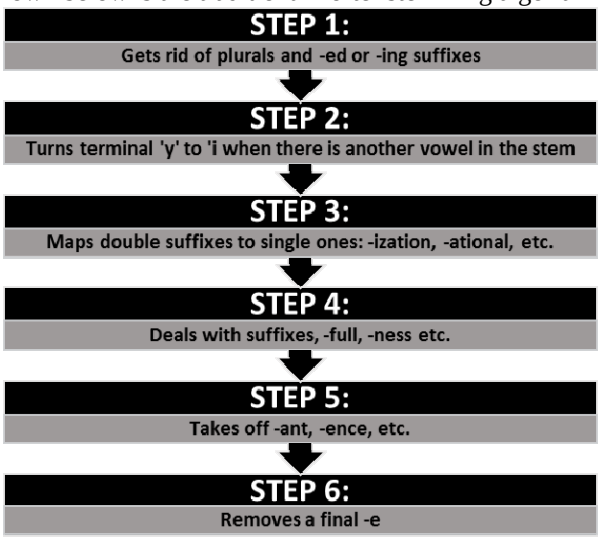

Fig. 1. Porter stemming algorithm

The removal of stop-words and duplicates is another function that the current Porter stemming algorithm does not address. Stop-words make up a significant portion of most text corpora and stemming these would only slow down the stemming process without contributing anything towards indexing or query expansion. Thus the prelude to our algorithm would be to define a strict entry condition for input words to prevent the stop words and duplicates of a given word from being considered for stemming.

Stemming algorithms in general have not kept up with the current linguistic trends. No rules have been defined to stem hyphenated words, email addresses and website names. We have devised rules to address these issues as well.

Thus to summarize, the following improvements have been made to the Porter stemmer:

1. A word count feature, which gives the number of occurrences of a word in the input text file.
2. Function to ensure that the duplicate of a given word will not be stemmed.
3. A reverse relation table showing the words from which the stems have been derived.
4. Function to remove the stop words from the input text file.
5. Addition of the rules to address the following:
    a. Hyphenated words
    b. Words with apostrophe
    c. Email addresses
    d. Websites

IV. Results and Discussion

Data Set: The text corpus considered to test the effectiveness of the new algorithm has been taken from the Leipzig corpora collection maintained by Wortschatz University, Leipzig. It is a collection of news stories that have been randomly selected from the year 2010. It contains about 30K words stored in plain-text format.

```
1     Intrigued I started to work it out.
2     The third installment of "The Twilight Saga: Eclipse,"
proved to be a small step-up from the first two movies.
3     Lowrie atoned for an error he made in the top half of the
inning, when he dropped a foul popup while playing first base.
4     But she couldnâ€™t sleep, knowing that her mother was
sitting alone somewhere in a big foreign airport that had gone
into crisis mode.
5     BRIDGE: Call the Minot YWCA at 838-1812 to reserve your spot
for Beginning Duplicate Bridge Monday evenings.
6     Plucking the track from her 2009 LP, I Am Sasha Fierce,
BeyoncÃ©'s "Halo" has become one of her signature songs.
7     Wow. Sound Smart should change his name to "Really Dumb".
8     It dabbles in several facets of the financial services
industry, from its JPMorgan investment banking, Chase credit
cards, and various banking operations.
9     The developing industrial applications for silver are
exciting and I expect long-term growth here.
10    Rick Morgan, 55, has been fishing Lower Manitou since he was
a kid growing up in Hibbing in the late 1960s.
```

Fig. 2. Sample text from Leipzig corpora

Fig. 2. Shows a sample input text taken from the Leipzig corpora. On feeding this input to our new and improved stemming algorithm, the result will be as follows:

```
Intrigued------->Intrigu-----1
started------->start-----1
work------->work-----1
third------->third-----1
installment------->instal-----1
Twilight------->Twilight-----1
Saga------->Saga-----1
Eclipse------->Eclips-----1
proved------->prove-----1
small------->small-----1
step-up------->step-up-----1
first------->first-----1
two------->two-----1
movies------->movi-----1
Lowrie------->Lowri-----1
atoned------->aton-----1
error------->error-----1
made------->made-----1
top------->top-----1
half------->half-----1
inning------->in-----1
dropped------->drop-----1
foul------->foul-----1
popup------->popup-----1
playing------->plai-----1
first------->first-----1
base------->base-----1
couldn't------->couldn't-----1
sleep------->sleep-----1
knowing------->know-----1
mother------->mother-----1
sitting------->sit-----1
alone------->alon-----1
somewhere------->somewher-----1
big------->big-----1
foreign------->foreign-----1
airport------->airport-----1
gone------->gone-----1
crisis------->crisi-----1
mode------->mode-----1
BRIDGE------->BRIDGE-----1
Call------->Call-----1
Minot------->Minot-----1
YWCA------->YWCA-----1
reserve------->reserv-----1
spot------->spot-----1
Beginning------->Begin-----1
Duplicate------->Duplic-----1
Bridge------->Bridg-----1
Monday------->Mondai-----1
evenings------->even-----1
Plucking------->Pluck-----1
track------->track-----1
Sasha------->Sasha-----1
Fierce------->Fierc-----1
Beyoncé's------->Beyoncé'-----1
Halo------->Halo-----1
become------->becom-----1
one------->on-----1
signature------->signatur-----1
songs------->song-----1
Wow------->Wow-----1
Sound------->Sound-----1
Smart------->Smart-----1
change------->chang-----1
name------->name-----1
Really------->Realli-----1
Dumb------->Dumb-----1
dabbles------->dabbl-----1
several------->sever-----1
facets------->facet-----1
financial------->financi-----1
services------->servic-----1
industry------->industri-----1
JPMorgan------->JPMorgan-----1
investment------->invest-----1
banking------->bank-----1
Chase------->Chase-----1
credit------->credit-----1
cards------->card-----1
various------->variou-----1
banking------->bank-----1
operations------->oper-----1
developing------->develop-----1
industrial------->industri-----1
applications------->applic-----1
silver------->silver-----1
exciting------->excit-----1
expect------->expect-----1
long-term------->long-term-----1
growth------->growth-----1
Rick------->Rick-----1
Morgan------->Morgan-----1
fishing------->fish-----1
Lower------->Lower-----1
Manitou------->Manit-----1
since------->sinc-----1
kid------->kid-----1
growing------->grow-----1
Hibbing------->Hib-----1
```

Fig. 3. Output of the stemming algorithm

As evident from the above figure, the output shows the stemmed word, the word from which the stem has been derrived and the number of times the original word occurs in the input text file.

Conclusion and Future Scope

Our proposed algorithm shows significant improvements in terms of scope and efficiency as compared to the current Porter stemming algorithm. With the semiconductor industry's shift to multi-core processors, parallel computing and more specifically high performance computing is emerging as the prevalent computing paradigm. However, current stemming algorithms have not kept up with this trend. A stemmer that is capable of leveraging the massive capabilities of parallel processing, making use of multiple CPU cores can perform efficiently with increased throughput, which will lead to a tremendous

increase in speed of the stemming process. One could also formulate additional rules based on the frequent exceptions and stemming errors encountered to improve the correctness and reliability of the algorithm. Our algorithm can be scaled to stem words in other Indo-European languages, provided one has a thorough understanding of their grammatical structure and lexicon.

REFERENCES

[1] Anjali Ganesh Jivani, *"A Comparative Study of Stemming Algorithms"*, in Int. J. Comp. Tech. Appl., Vol 2 (6), 1930-1938, 2011

[2] Eiman Tamah Al-Shammari, *"Towards An ErrorFree Stemming"*, in Proceedings of ADIS European Conference Data Mining 2008, pp. 160-163.

[3] Frakes William B. *"Strength and similarity of affix removal stemming algorithms"*. ACM SIGIR Forum, Volume 37, No. 1. 2003, 26-30.

[4] Harman, D. (1991). *"How effective is suffixing?"* Journal of the American Society for Information Science, 42(7), 7-15.

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

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

[7] Melucci Massimo and Orio Nicola. *"A novel method for stemmer generation based on hidden Markov models"*. Proceedings of the twelfth international conference on Information and knowledge management. 2003, 131-138.

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

[9] Paice Chris D. *"An evaluation method for stemming algorithms"*. Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. 1994, 4250.

[10] Porter M.F. *"An algorithm for suffix stripping"*. Program. 1980; 14, 130-137.

[11] Porter M.F. *"Snowball: A language for stemming algorithms"*. 2001.

[12] 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 25, Issue 4. 2007, Article No: 18

[13] Robert Krovetz. *"Viewing morphology as an inference process."* In Proc. of the 16th ACM/SIGIR Conference, pages 191-202, 1993.

[14] Toman Michal, Tesar Roman and Jezek Karel. *"Influence of word normalization on text classification"*. The 1st International Conference on Multidisciplinary Information Sciences & Technologies. 2006, 354-358.

[15] Xu Jinxi and Croft Bruce W. *"Corpus-based stemming using co-occurrence of word variants"*. ACM Transactions on Information Systems. Volume 16, Issue 1. 1998, 61-81