Analysis of Popular Steaming Algorithms Supporting Information Retrieval System

Madhurima V , Prof. T.Venkat Narayana Rao , L. Sai Bhargavi
Student B.Tech, Final Year, C.S.E, Guru Nanak Institutions Technical Campus
Professor, Computer Science and Engineering, Guru Nanak Institutions Technical Campus
Assistant Professor, Computer Science and Engineering, Guru Nanak Institutions Technical Campus
R.R. District, Ibrahimpatnam, A.P, India

Abstract: Information retrieval is the activity of gaining information resources significant to an information need from a collection of information resources. Searches can be based on metadata . Computerized information retrieval systems are used to decrease what has been called "information overload". Several universities and public libraries use Information Retrieval systems to offer access to journals, books, other documents. Web search engines are the most visible Information retrieval applications. There are plenty of IR algorithms such as Stemming algorithms which process the text for reducing sometimes derived words to their stem, base or root form - generally a written word form. The use the term conflation, meaning the act of fusing, as the general term for the process of matching physiological term alternatives. Conflation can be:1. manual--using some kind of regular statements 2. automatic, via programs called stemmers. Algorithms for stemming have been studied in computer science since 1968 .Stemming programs are generally referred to as stemming algorithms or stemmers. This paper focus on some popular algorithms and also tender the comparative study with analysis on Stemming algorithms.

Keywords: stem, stemmers, conflation, conflation methods, n-grams.

I.INTRODUCTION

The stem need not be identical to the physiological root of the word; it is usually sufficient that related words map to the similar stem, even if that stem is not in itself a valid root. One of the best technique for improving Information Retrieval performance is to provide searchers with ways of finding physiological variants of search terms. For an example, if a searcher enters the term *stemming* as part of a query, it is likely that user will also be interested in such alternatives as *stemmed* and *stem*. We use the term *conflation*, as the general term for the process of matching morphological (physiological) term variants. Stemming is also used in IR to decrease the size of index files [1]. As a single stem typically corresponds to many full terms, by storing stems in place of terms, compression factors of more than 50% can be achieved.

Figure 1. shows types for stemming algorithms. There are four automated approaches:

- Affix removal algorithms eliminates suffixes and prefixes from terms leaving a *stem*. These algorithms also transform the resultant stem. The name stemmer is derived from this method, which is common.
- Successor variety stemmers use the frequencies of letter sequences in the text body as the basis of stemming.
- n-gram method conflates terms based on the number of n-grams or digrams they share. Terms and their corresponding stems can be stored in a table.

- Stemming is then done via lookups in the table.These methods are described below.

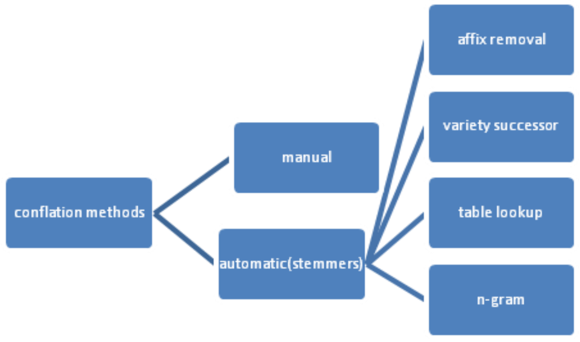

Figure 1: Conflation methods

There are several criteria for judging stemmers:compression performance,retrieval correctness and effectiveness. There are two ways stemming can be not correct:

(i).Overstemming and (ii).Under stemming. When a term is overstemmed, too much of it is eliminated. Overstemming can cause unrelated terms to be combined. The effect on IR performance is retrieval of non related documents. Under stemming is the removal of too little of a term. It will prevent related terms from being combined. The effect of under stemming on IR performance is that relevant documents will not be retrieved. Stemmers can also be determined on their retrieval effectiveness-usually measured with recall and precision as explained on their size,speed and so on.Finally, they can be rated on their compression performance. Stemmers for Information Retrieval are not usually judged on the basis of syntactical correctness, though the stems they produce are usually very similar to root morphemes, as described below.Affix removal is further divided as shown in the fig.1(a)

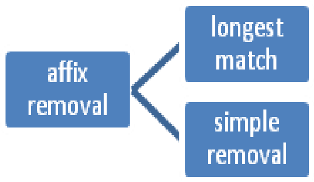

Fig.1(a):Affix removal

I.a:Example of Stemmer Use in Searching :

To illustrate how a stemmer is used in searching, take the below example from the ‘catalog system’ (Frakes 1984, 1986). In this system terms are stemmed at search time rather than at indexing time. catalog prompts for queries with the string "Look for:". At the prompt, the user types in one or more terms of interest [2].

For example: Look for system users will cause catalog to attempt to find documents about system users. catalog takes each term in the query, and tries to determine which other terms in the database might have the same stem. If any possibly related terms are found, catalog presents them to the user for selection. In the case of the query term "users," for example, catalog might respond as follows: Search Term:

S.no.	**Term**	**Occurrences**
1.	purchaser	15
2.	purchasers	1
3.	purchased	3
4.	purchasing	2

Which terms (0 = none, CR = all):
The user selects the terms by entering their numbers. This method of using a stemmer in a search period provides a naive system user with the advantages of term conflation while requiring some knowledge of the searching methodologies. It also allows experienced searchers to focus their attention on other search complications. Since stemming may not always be convenient, the stemmer can be offend by the user. Having a user select the terms from the set found by the stemmer also reduces the possibility of false matches.

II.KINDS OF STEMMING ALGORITHMS

There are many approaches to stemming[5]. One way of stemming is to store a table of all index terms and their related stems. For example:

Term	**Stem**
Engineering	Engineer
Engineered	Engineer
Engineer	engineer

Terms from queries and indexes could then be stemmed via lookups in the table. Using a B-tree or hash table, such lookups would be fast.There are complications with this advent. The first thing is that there is no such content for English. Even if there were, many terms found in databases wouldn’t be represented, as they are domain dependent that isn’t standard English. For these terms, some other stemming approach would be required and the second is that storage overhead for such kind of table, though trading size for time is sometimes guaranteed. Storing precomputed data, as opposed to computing the data values on the fly, is useful when the computations are expensive or frequent. For example, reports cases such as chess computations where storing precomputed results gives significant performance improvements.

A.Successor Variety

Successor variety stemmers are based on work in structural linguistics which attempted to determine word and morpheme boundaries based on the distribution of phonemes in a large body of utterances. The stemming method rooted on this work uses letters in place of phonemes, and text body in place of phonemically transcribed utterances.Hafer and Weiss formally defined the technique as follows:Let α be a word of length n; α_i, is a length i prefix of α. Let D be the corpus of words. $D\alpha i$ is defined as the subset of D containing those terms whose first i letters match α_i exactly. The successor variety of α_i, *denoted* $S\alpha i$, is then defined as the number of distinct letters that occupy the i + 1st position of words in $D\alpha i$. A test word of length n has n successor varieties $S\alpha i_{,}\ S\alpha 2$, . . . , $S\alpha n$. In less formal tenure, the successor kind of a string is the number of different characters that follow it in words in some body of text[8]. Consider a body of text consisting of the following words, for example.able, axle, accident, ape, about. To determine the successor varieties for "apple," for example, the following process would be used. The first letter of apple is "a." "a" is followed in the text body by four characters:

"b," "x," "c," and "p." Thus, the successor variety of "a" is four. The next successor variety for apple would be one, since only "e" follows "ap" in the text body, and so on.When this process is carried out using a large body of text the successor variety of substrings of a term will decrease as more characters are added until a segment boundary is achieved. At this point, the successor variety will sharply increase. This information is used to identify stems.Once the successor varieties for a given word have been derived, this data must be used to fragment the word.

To illustrate the use of successor type stemming, take the example below where the task is to determine the stem of the word BEATABLE. Test Word: BEATABLE Corpus: ABLE, APE, BEATABLE, FIXABLE as shown in Table 1. .

Table 1. Successor Stemming

Prefix	Successor variety	Letters
B	3	E,I,O
BE	2	A,D
BEA	1	D
BEAT	3	A,I,S
BEATA	1	B
BEATAB	1	L
BEATABL	1	E
BEATABLE	1	BLANK

Using the complete word segmentation method, the test word "BEATABLE" will be segmented into "BEAT" and "ABLE," since BEAT appears as a word in the corpus. The peak and plateau approach would give the identical result. After a word has been divided, the division to be used as the stem must be selected.Weiss and Hafer used the below rule:

if (1st division occurs in <= 12 words in corpus) 1st division is stem
else (2nd division is stem)

The check on the number of occurrences is based on the observation that if a division occurs in more than 12 words in the corpus, it is mostly a prefix. The authors report that because of the infrequency of multiple roots in English, no fragment beyond the second is ever selected as the stem. Using this rule in the example above, BEATwould be selected as the stem of BEATABLE.In summary, the successor variety stemming process has three parts: (i)determine the successor varieties for a word, (ii) use this information to segment the word using one of the methods above, and (iii) select one of the segments as the stem. The aim of “Hafer and Weiss” was to develop a stemmer that required little or no mankind processing. They point out that while affix removal stemmers work well, they require mankind preparation of suffix lists and discarding rules. Their stemmer requires no such preparation.

B.n-gram stemmers

A report was written on a method of conflating terms called the ‘shared digram method’. A ‘digram’ is a pair of consecutive letters. As trigrams, or n-grams could be used, we have called it the n-gram method[6] [5]. Though we call this a "stemming approach," this is a bit confusing since there is no stem is produced.

In this method, association measures are calculated between pairs of terms based on shared unique digrams. For example, the terms mathematics and mathematical can be broken into digrams as follows:

mathematics => ma at th he em ma at ti ic cs
unique digrams = th he em ti ic cs
mathematical => ma at th he em ma at ti ic ca al
unique digrams =th he em ti ic ca al

Thus, "mathematics" has ten digrams, sixof which are unique, and "mathematical" has eleven digrams, seven of which are unique. The two words share five unique digrams: th,he,em,ti,ic

Once the unique digrams for the word pair have been noticed and counted, a similarity measure based on them is computed [3]. The similarity measure used was Dice's coefficient, which is defined as

$$S = \frac{2C}{A + B}$$

where *A* is the number of unique digrams in the first word, *B* the number of unique digrams in the second, and *C* the number of unique digrams shared by *A* and *B*. Since Dice's coefficient is symmetric ($S_{ij} = S_{ji}$), a lower triangular similarity matrix can be used as in the example below.

	$word_1$	$word_2$	$word_3$. . .$word_{n-1}$
$word_1$			
$word_2$	S_{21}		
$word_3$	S_{31}	S_{32}	

◆

◆

$word_n$ S_{n1} S_{n2} S_{n3} $S_{n(n-1)}$

If such a similarity matrix is usable, terms are clustered using a single link clustering method[3] .The algorithm for calculating a digram similarity matrix follows.

```
#define MAXGRAMS 50
#define GRAMSIZE 2
void
digram_smatrix (wordlist, word_list_length, smatrix)
char *wordlist[];
int word_list_length;
double *smatrix[];
{
int i, j;
int uniq_in_wordl;
int uniq_in_word2;
int common_uniq;
char uniq_digrams_1 [MAXGRAMS] [GRAMSIZE];
char uniq_digrams_2 [MAXGRAMS] [GRAMSIZE];
int unique_digrams();
int common_digrams();
for ( i=0; i< word_list_length; ++i)
for (j=i+1; j <word_list_length ;++j)
 {
uniq_in_word1 = unique_digrams(wordlist [i],
uniq_digrams 1);
uniq_in_word2 = unique_digrams(wordlist [j],
uniq_digrams_2);
common_uniq = common_digrams(uniq_digrams_1,
uniq_digrams_2);
smatrix[i][j] =
(2*common_uniq)/(uniq_in_word1+uniq_in_word2);
}
}
```

C.Affix Removal Stemmers

Affix removal algorithms remove suffixes and/or prefixes from terms freeing a *stem*. These algorithms sometimes also transfer the resultant stem. A basic example of an affix removal stemmer is one that removes the plurals from terms[4]. A set of rules for such a stemmer is as follows :

If a word ends in "ies" but not "eies" or "aies"
Then "ies" -> "y"
If a word ends in "es" but not "aes", "ees", or "oes"
then "es" -> "e"
If a word ends in "s", but not "us" or "ss"
then "s" -> NULL

In this algorithm only the 1st applicable rule is used.

The Porter algorithm consists of a set of action rules[2]. The restrictions fall into three classes: restrictions on the stem, restrictions on the suffix, and restrictions on the rules.

There are many types of stem conditions.

(i). The *measure*, denoted *m*, of a stem is based on its alternate Vowel-Consonant effect. Vowels are [a, e, i, o, u] and [y] if preceded by a consonant. Consonants are all letters which are not vowels. Let *C* stand consonants, and *V* for vowels. The measure *m*, then, is defined as

$[C](VC)^m[V]$

The superscript *m* in the formulation, which is the measure, shows the number of *VC* sequences. Square brackets show an optional occurrence. Some examples of measures for terms follows as shown in table 2.

Table 2. Porter stemmer Example

Measures	Example
m=0	Shy,tree
m=1	cold
m=2	Enroll
m=3	Graceful,isotherm

(ii). * < X >--the stem ends with a given letter X
(iii). *v*--the stem contains a vowel
(iv). *d--the stem ends in a double consonant
(v). *o--the stem ends with a *CVC* sequence, where the last *C* is not w, x, or y.

Suffix conditions take the form:
(current_suffix == pattern).
Rule specifications take the form: (rule was used).
Actions are rewrite rules of the form:
old_suffix -> new_suffix

The rules are divided into steps. The rules in a step are examined[7] in sequence, and only one rule from a step can be applied. The longest possible suffix is always deleted because of the ordering of the rules within a step. The algorithm is as shown below.

```
{
step1a(word);
step1b(stem);
if (the second or third rule of step 1b was used)
step1b1(stem);
step1c(stem);
step2(stem);
step3(stem);
step4(stem);
step5a(stem);
step5b(stem);
}
```

The conditions for the steps of the stemmer are as shown below:

Table 3. Stemmer rules and Conditions

Step	Condition	Suffix	Replacement	Example
1a	NULL	sses	ss	Glasses->glass
1b	*v*	ing	NULL	Walking->walk
1b1	NULL	at	ate	Generated->generate
1c	*v*	y	i	Classy->classi
2	m>0	ality	al	Formality->formal
3	m>0	icat	ic	Duplicate->duplic
4	m>1	able	NULL	Favourable->favour
5a	m>1	e	NULL	Annihilate->annihil
5b	m>1	NULL	single	Animall->animal

D.YASS (Yet Another Suffix Striper) Algorithm
Conflation can be viewed as a clustering difficulty with a-priory unknown number of clusters. Usually such problems can be solved with a hierarchical algorithm having a distance measure function for cluster elements. Then the resulting clusters are considered as equivalence classes and their centroids as stems. This paper suggests several distance measures rewarding long matching prefixes and penalizing early mismatches. Apparently, a threshold must be chosen to distinguish elements belonging to a cluster and lying outside it. Unfortunately, it seems that this problem can be solved only experimental. Also, the approach requires significant computer power especially if it is applied to a essential inventory. On the other side, as most statistical algorithms, this one can be used for any language without knowledge of its structure. The authors also showed that inventory clustering enhances recall almost as well as Porter's [4][8].

III.RESULTS AND DISCUSSION

This paper summarizes the various studies of stemming for improving retrieval effectiveness. These studies must be viewed with caution. The failure of some of the authors to report test statistics, especially effect sizes, make interpretation difficult[6]. Since some of the studies used sample sizes as small as 5 their validity is questionable. Given these cautions, we offer the following conclusions.

- There is a chance that stemming can effect information retrieval performance, but the studies are doubtful. Where effects have been found, the majority have been positive, with the Hafer and Weiss stemmer in the study and the effect of the strong stemmer in the Walker and Jones study, being the exceptions. Otherwise there is no evidence that stemming will decrease the retrieval effectiveness.
- Stemming is as effective as manual conflation.Results indicate that the effect of stemming is dependent on the vocabulary nature used. A specific and homogeneous vocabulary may exhibit different conflation properties than will other kinds of vocabularies.It also appears that there is a little difference between the retrieval effectiveness of different full stemmers, with the exception .

IV.COMPARATIVE STUDY AND FUTURE SCOPE

Here we will compare the performance of various stemming approaches discussed till now. In this comparison we consider one rule-based approach and compare it with statistical approaches. The parameters used in this comparison are each stemmer's features, strengths and limitations required by each stemmer to compute the stem. Although a lot of research work has already been done in developing stemmers there still remains a lot to be done to improve recall as well as precision. There is a need for a method and a system for efficient stemming that reduces the heavy tradeoff between false positives and false negatives. A stemmer that uses the syntactical as well as the semantical knowledge to reduce stemming errors should be developed. Perhaps developing good lemmatizer could help in achieving the goal.

Table 3. Comparison of Stemming Algorithms

Algorithm	**Features**	**Strengths**	**Limitations**
Successor variety	Based on the structural linguistic	Simple form.	-Heavy algorithm and complex.
n-gram stemmer	Based on the concept of n-grams and string comparisions	Language independent	-Not time efficient and. requires more space for indexing n-grams. -Not very practical method
Porters /Affix removal stemmers	Based on removing the suffixes and prefixes from terms of a stem.	Produces the best output when compared to other algorithms. Less error rate.	-The stems produced are not always real words.
YASS algorithm	Stemmers created using hierarchical approach.	Based on Hierarchical clustering approach and distance measures. -It is also a corpus based method.Can be used for any language without knowing its morphology.	-Difficult to decide a threshold for creating clusters. -Requires significant computing power

V.CONCLUSION

As it can be seen from all the results of stemming algorithms that have been discussed so far, there is lot of similarity between the stemming algorithms and if one algorithm does better in one area, the other does better in some other area. In fact, none of them gave perfect results but are good enough to be applied to the text mining, NLP or IR applications. The main difference lies in using either a rule-based approach or a linguistic one. A rule based approach may not always give correct output and the stems generated may not always be correct words. As far as the linguistic approach is concerned, since these methods are based on a lexicon, words outside the lexicon are not stemmed properly. It is of utmost importance that the lexicon being used is totally exhaustive which is a matter of language study. A statistical stemmer may be language independent but does not always give a reliable and correct stem. The problem of over stemming and under stemming can be reduced only if the syntax as well as the semantics of the words and their POS is taken into consideration. This in conjunction with a dictionary look-up can help in reducing the errors and converting stems to words. However, no perfect stemmer has been designed so far to match all the requirements.

VI.REFERENCES

[1]. Frakes W.B. “Term conflation for information retrieval”. Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval. 1984,383-389.
[2]. Porter M.F. “An algorithm for suffix stripping”.Program. 1980; 14, 130-137.
[3]. Dawson John. “Suffix removal and word conflation”. ALLC Bulletin, Volume 2, No. 3.1974,33-46
[4]. Porter M.F. “Snowball: A language for stemming algorithms”. 2001. http://snowball.tartarus.org/texts/introduction.html
[5].Frakes,W.&BaezaYates,R.,eds(1992),InformationRetrieval:DataStructuresandAlgo- rithms,Prentice-Hall.
[6]. T.G. Rose, M. Stevenson, M. Whitehead. The Reuters Corpus Volume 1 - From Yesterday’s News to Tomorrow’s Language Resources. In Proc. LREC’02, 2002.
[7]. M. Braschler and B. Ripplinger. How effective is stemming and decompounding for german text retrieval? Information Retrieval, 7(3-4):291–316, 2004.
[8]. D. R. Morrison. PATRICIA—Practical Algorithm to Retrieve Information Coded in Alphanumeric. J. of the ACM, 15(4):514–534, October 1968.