http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 5 (4): 685-688, 2006
ISSN 1812-5638

STANS Algorithm for Root Word Stemming

[1]S. Srinivasan and [2]P. Thambidurai
[1]Department of Computer Science and Engineering,
Sathyabama Institute of Science and Technology, Old Mamallapuram Road, Chennai-600 119, India
[2]Department of CSE and IT, Pondicherry Engineering College, Pillaichavady,
Pondicherry-605 014, India

Abstract: Information Retrieval (IR) is essentially a matter of deciding which documents in a collection should be retrieved to satisfy a user's need for information. In most cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of IR applications. Document indexing will also be more meaningful if intelligible root words are used in place of stems. Popular stemming algorithms do not produce intelligence stems from the input. In this study, we propose STANS algorithm, which is a modified version of the widely used Porter's stemming algorithm. Considering only meaningful root words as the stemmer output, the modified algorithm reduces the error rate of Porter's algorithm from 21 to 3% without compromising the efficiency.

Key words: Conflation, deterministic finite state automation, precision, recall, stemming

INTRODUCTION

Conflation is the process of merging or lumping together non identical words which refer to the same principal concept. Conflation can be either manual or automatic. Programs for automatic conflation are called stemmers (Paice, 1994; Porter, 1980). Stemmers are used in IR to reduce the size of index files. Since a single stem typically corresponds to many full terms, compression factors of over 50% can be achieved by storing stems instead of terms, especially in the case of affix-removal stemmers. Terms can be stemmed either at indexing time or at search time. The advantages of stemming at indexing time are efficiency and index file compression. Capability of search engines to find morphologically related terms with respect to every term would generally enhance the IR performance.

It is clear that retrieval performance can go down significantly when stemming results in unrelated words being conflated to a single stem. It is required that the stem produced is very close to their root morphemes. Most stemmers today do not always provide root words as their outputs. Taking note of the fact that linguistic correctness of stems may become critical to effective retrieval in future, design of a root word stemmer has been proposed in this paper. This root word stemmer is an improved version of the popular affix stemmer, developed by Porter (1980).The rule base of Porter's stemmer has been considerably enhanced so as to give meaningful steams as output in as many cases as possible.

CLASSIFICATION OF STEMMERS

Among the different approaches to stemming, the prominent ones are Table look-up, Affix removal, Successor variety and n-g stemmers. We have used a rule-based approach to implement our algorithm.

Table look-up: Stemming can be done by storing all index terms in a table, against their respective stems. Terms from queries and indices could then be stemmed by table look-up. An example is shown in Table 1.

Affix removal stemmers: Most stemmers currently in use are iterative longest match stemmers. Such a stemmer removes the longest possible string of characters from a word according to a set of rules. This process is repeated until no more characters can be removed (Paice, 1994; Porter, 1980). The Lovins stemmer (Frakes *et al*., 1982) removes the longest-matching suffix whereas the Porter's algorithm iteratively removes suffixes from a pre-defined set. The most popular one among these is the Porter's Suffix Stripping algorithm (Porter, 1980), which is more compact than the stemmers of Lovins (1971). We observe that all the approaches to stemming mentioned above not only ignore word meanings, but also operate in the

Corresponding Author: S. Srinivasan, Department of Computer Science and Engineering,
Sathyabama Institute of Science and Technology, Old Mamallapuram Road, Chennai-600 119, India

Table 1: Terms and stems

Term	Stem
Connecting	Connect
Connected	Connect
Connection	Connect
Connect	Connect

absence of any lexicon at all. The design goals of all of them seem to be better retrieval effectiveness and compression performance and not the production of a linguistically correct root word. Our aim is to maximize the proportion of the meaningful stems (root words) in the stemmer output for a given set of input words, without compromising the other performance measures.

CRITERIA FOR JUDGING STEMMER PERFORMANCE

Stemmer Evaluation measures are discussed by lovins (1971), Paice (1994) Kraaij and Pohlmann (1996). Correctness, retrieval effectiveness and compression performance are the major performance measures of stemmers. The degrees of overstemming and understemming are two other measures that indicate how incorrect a stemmer can be. Stemmers can also be judged by their retrieval effectiveness, usually measured by recall and precision, as well as the speed and size. This involves substituting different stemmers to see which gives the best precision and recall. As a third measure, they can be judged by their compression performance. We use a new evaluation measure in which the ability of the stemmer to output intelligible stems is the only performance measure. We would like to derive meaningful stems from the words, which imply that the derived stems are linguistically correct.

PORTER'S STEMMING ALGORITHM

Porter's Stemmer is a conflation stemmer and is based on the idea that the suffixes in the English language are mostly made up of a combination of smaller and simpler suffixes. This stemmer is a linear step stemmer. It has five different steps, at each of which rules are applied. If a suffix rule gets matched to a word, then the conditions attached to that rule are tested for the resulting stem as if that suffix were removed, as per the rule. One such condition may be that the number of vowel characters followed by a consonant character in the stem must be greater than one, for the rule to be applied. Porter's algorithm does not remove a suffix when the stem is too short, i.e. when the number of vowel-consonant pairs in a word is zero. The number of vowel-consonant pairs, denoted by m, is used as the decision making variable. We now show how this value may be computed for an arbitrary word. In order to count the number of vowel-consonant pairs in a word, disregarding initial consonants

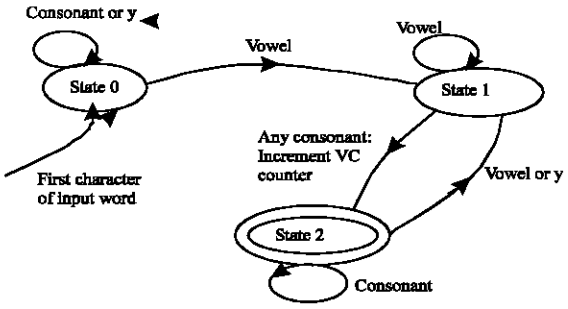

Fig. 1: The deterministic finite state automation

and final vowels, we use a finite state technique, which is demonstrated through Fig. 1. The character y is considered as a consonant at the beginning of a word or if it has a vowel ahead of it. Otherwise, it is treated as a vowel.

In the initial state 0, the first character is checked. If it is a vowel, then the machine changes to state 1, which is the last character was a vowel state. If the next character is a consonant or y, then it changes to state 2, the last character was a consonant state. In state 1, a y is treated as a consonant (since it follows a vowel), but in state 2, y is treated as a vowel (since it follows a consonant). The result counter is incremented on the transition from state 1 to state 2 , since this transition occurs only after a vowel-consonant pair which is what we are counting. Porter (1980) algorithm may remove a few characters of the word being stemmed. These characters may be restored, in a later step. Each of the subsequent steps transforms the partial stem, depending on the rule applicable for the given value of m. When a rule fires, the suffix is removed and the control moves to the next step. The rules in a step are tested successively until either a rule from that step fires and control passes to the next step, or there are no more rules in that step, in which case control moves to the next step. This process continues at all the five steps, until the stem is returned by the stemmer. Note that, in a particular step, it is sufficient if exactly one rule matches. It then proceeds to the next step.

We list some of the drawbacks of Porter's algorithm, from the viewpoint of getting intelligible stems as output. Consider Table 2, which shows two lists A and B, with m value greater than 0. The condition for stripping the suffix-ATE is:(m>0) ATE '-', an empty right hand side in the rule indicates the removal of the suffix indicated on the left hand side. Here, the suffix -ATE is removed from the words of list B, whereas the suffix E is removed from the words of list A. Complex suffixes are removed letter by letter in different steps. Thus, the word generalizations is first stripped to get generalization (Step 1); then to generalize (Step 2); then to general (Step 3) and then finally to GENER (Step 4), which is not an intelligible word.

Table 2: Words and the measure m

List A	List B
Relate	Derivate
Probate	Activate
Conflate	Demonstrate
Pirate	Necessitate
Prelate	Renovate

The main feature of this algorithm is that it does not make use of a stem dictionary (lexicon). As per the algorithm, a list of suffixes is considered and with each suffix, there is a criterion under which it may be removed from a word during the formation of a valid stem. The main merits of this algorithm are that it is small, fast and reasonably simple.

The algorithm feature often produces stems, which are unintelligible. For example, the stems result for the word serious is seriou. This is one of the cases in which the suffixs has been identified as denoting the plural form of the given word. We also get stems like Microscop, Secur and Envi for Microscope, Secure and Envy respectively, which are clearly meaningless. As another example, if Sand and sander get conflated, so do wand and wander. The error committed here is that the -er part of wander has been treated as its suffix when in fact it is part of the stem. However, the absence of a lexicon results in improper conflations and an associated loss of precision. Converting words like general into gener and iteration to iter make it difficult to relate them to dictionary entries. It also complicates the process of query enhancement.

STANS ALGORITHM

We observe that the addition of more rules in order to increase the performance in one section of the vocabulary may cause degradation of performance elsewhere. After a detailed analysis, several new suffixes and the context in which they must be removed have been identified and appropriate changes have been made to the Porter's algorithm.

In all, 31 modifications comprising of 15 additions, 11 replacements and 5 deletions have been carried out so that the modified Porter's algorithm now consists of 65 rules compared to 60 in the original algorithm. The modifications are listed in Table 3, without changing the order suggested by Porter.

The first modified rule US→US, in step1.1 of the algorithm helps to avoid the stripping of the suffix S from words like serious, various and so on. Another modification was done to the same step so that stems given by the algorithm for words like succeed as success.

In order to compensate for the former changes for words like greed, breed, etc., a new rule EED→EED was introduced. Another rule IES→Yreplaces IES→I in the Porter's algorithm to transform words like Ponies to Poni. The rule IED→Y was added to transform words like modified, tried to modify, try. The stems for gerunds such as scoring→score, words of the form (*V*) ING→ have been modified to (*V*) ING→E. The rule (*V*) →I is deleted to transform words like happy to happi. A new rule ED→E is added to generate stems for words like provided (to generate provide).

The modifications in step 2 are described as follows: The rule ANCI→ANCE was modified into ANCY→ANCE. This rule was used to generate stems for medicancy to medicance. The rule ENCI→ENCE is assumed to be replaced by ENCY→ ENCY. The other modification in step 2 is ABLI→ABLE that is replaced with ABLY→ABLY.

Similarly the rule OUSLI→OUS has been modified to OUSLI→OUS. As a result, seriously is stemmed into serious. Word like captivity can be stemmed into captive using rule 14, IVITY→IVE. On modifying BILITI→BLE as BILITY→BLE, a word like capability is stemmed into capable. A new rule FULLY→FUL was added to stem words like carefully to careful. The rule FUL→ removes the said suffix from the word completely. For example, careful is stemmed into care. The rule LESSLY→LESS stems words like carelessly to careless. A word like possibly is stemmed into possible using BLY→BLE. There are only 2 changes in step 3. A new rule LESS→ is added to remove suffix less from worthless. For example, this rule remove the suffix less and generate worth. The rule ICITI → IC is modified into ICITY → IC to consider cases like publicity (public), electricity (electric). Changes in step 4 are described below:

The rule AL → is modified into AL → E. This rule is to transform revival to revive. A new rule IABLE → Y was added to generate stems for words like modifiable (modify). The rule ANCE → is discarded because it results in meaningless stems like vari for variance. The stems for words ending in scopic, such as microscopic (microscope), are obtained by applying the rule SCOPIC → SCOPE. Removal of IC suffix in done away with, so that on stemming words such as gyroscopic do not end up on gyroscope. The rule ATE→ is discarded to avoid unpredictable results for words like activate. The rule FYE→FY was added to convert words like Purifying into their root form Purify. FYE→FY have been added. The e gets added as a result of a rule in step1 replacing ing with e. The rule TLY→T takes care of words like conveniently (convenient) which is not done in original algorithm. The change in step 5 is the rule which removes the trailing E from words such as possible has been removed (Table 3).

From Table 4, it can be seen that the modifications have led to the generation of better stem words.

Table 3: Modified rules

Step No.	Rule No.	Operation	Porter's algorithm	STANS algorithm
1	1	Add	--	US→ US
	2*	Add	--	CEED→ CESS
	3*	Add	--	EED→ EED
	4	Modify	IES→I	IES→Y
	5	Add		IED→Y
	6	Modify	(*V*)ING→	(*V*)ING→ E
	7	Delete	(*V*)Y→I	
	8	Add		ED→ E
2	9	Modify	ANCI→ANCE	ANCY→ ANCE
	10	Modify	ENCI→ENCE	ENCY→ ENCY
	11	Modify	ABLI→ABLE	ABLY→ABLY
	12*	Modify	OUSLI→OUS	OUSLY→ OUS
	13*	Modify	ALITI→AL	ALITY→ AL
	14*	Modify	IVITI→IVE	IVITY→IVE
	15*	Modify	BILITI→BLE	BILITY→ BLE
	16	Add		FULLY→FUL
	17*	Add		FUL→
	18	Add		LESSLY→ LESS
	19	Add		BLY→BLE
3	20	Add		LESS→
	21*	Modify	ICITI→IC	ICITY→ IC
4	22	Modify	AL→	AL→ E
	23	Add		IABLE→ Y
	24	Delete	ANCE→	
	25	Add		SCOPIC→ SCOPE
	26	Delete	IC→	
	27	Delete	ATE→	
	28*	Add		FYE→ FY
	29*	Add		ALLY→AL
	30*	Add		TLY→T
5	31	Delete	E→	

Table 4: Comparison of stems generated by Porter's algorithm and STANS algorithm

Word	Stems by original	Stem by STANS
Probate	Probat	Probate
Deadly	Deadli	Deadly
Microscopic	Microscop	Microscope
Possibly	Possibli	Possible
Serious	Seriou	Serious
Verifiable	Verifi	Verify
Carefully	Carefulli	Care
Carelessly	Carelessly	Care
Purifying	Purifye	Purify
Typically	Typic	Typical
Succeed	Succee	Success
Capability	Capability	Capable
Captivity	Captivity	Captive
Conveniently	Conveniently	Convenient
Electricity	Electricity	Electric
Scoring	Scor	Score
Happy	Happi	Happy
Provided	Provid	Provide
Seriously	Seriously	Seriousi
Archaeology	Archaeologi	Archaeology

EVALUATION

We measured the performance of the STANS stemmer's output by finding the proportion of meaningful stems produced by it and comparing that with the output of the original Porter's algorithm. We used a set of 30000 words excluding stop words. Our algorithm produced meaningful stems in 97% of the cases on an average, while the original algorithm was successful in 79% cases as shown in Fig. 2.

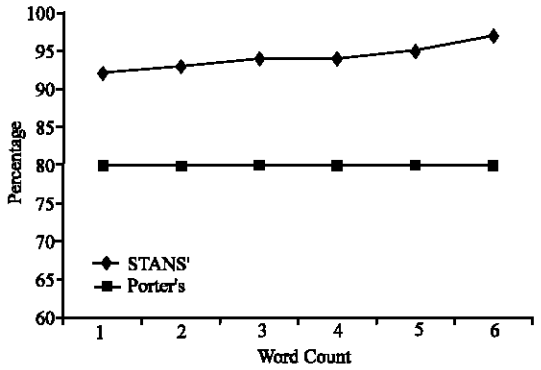

Fig. 2: Performance comparison

Thus the error rate of our algorithm is 3% against 21% due to Porter's algorithm. This improvement has been achieved without losing the efficiency of Porter's algorithm.

CONCLUSIONS

The popular Porter's stemming algorithm was studied with an aim to produce intelligible stem as output, in order to enhance the effectiveness of IR system. By modifying the rule based used by Porter's algorithm, the accuracy was improved from 79 to 97% in terms of proportion of meaningful stems produced. The STANS stemmer can be effectively used in the preprocessing stages of text summarization system and text classification systems in the content of information retrieval.

ACKNOWLEDGMENT

The authors wish to thank Aarthi Jerome and Nithya for their help at implementation stage.

REFERENCES

Frakes, W.B. and R. Baeza-Yates, 1992. Information Retrieval: Data Structures and Algorithms, Englewood Cliffs, NJ, Prentice-Hall Inc.

Kraaij, W. and R. Pohlmann, 1996. Viewing stemming as recall enhancement. Proceedings of the 17th ACM SIGIR Conference, Zurich, August 18-22, pp: 40-48.

Lovins, J.B., 1971. Error evaluation for stemming algorithms as clustering algorithms. J. Am. Soc. Inform. Sci., 22: 28-40.

Paice, C.D., 1994. An Evaluation Method for Stemming Algorithms in Croft, W.B. and C.J. van Rijsbergen, (Eds.). Proceedings of the 17th ACM SIGIR conference, Dublin, July 3-6, pp: 42-50

Porter, M., 1980. An Algorithm for Suffix Stripping. Automated Library and Information Systems, pp: 130-137.