Vol 5. No. 4, June 2012 ISSN 2006-1781
African Journal of Computing & ICT

www.ajocict.net

Context-Aware Stemming Algorithm for Semantically Related Root Words

[1]K.K. Agbele, [2]A.O. Adesina, [3]N.A. Azeez , & [4]A.P. Abidoye
[1, 2, 3, 4] University of the Western Cape, Computer Science Department
Private Bag X17, Bellville 7535, Cape Town, South Africa
[1, 2, 4] Machine Learning and Intelligent Systems Research Group
[3]Grid Computing and Security Research Group
agbelek@yahoo.com, inadesina@gmail.com, nurayhn1@gmail.com, ademaola.abidoye@gmail.com

[1]Corresponding Author

ABSTRACT
There is a growing interest in the use of context-awareness as a technique for developing pervasive computing applications that are flexible and adaptable for users. In this context, however, information retrieval (IR) is often defined in terms of location and delivery of documents to a user to satisfy their information need. In most cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of IR applications. Consequently, document indexing will also be more meaningful if semantically related root words are used instead of stems. The popular Porter's stemmer was studied with the aim to produce intelligible stems. In this paper, we propose Context-Aware Stemming (CAS) algorithm, which is a modified version of the extensively used Porter's stemmer. Considering only generated meaningful stemming words as the stemmer output, the results show that the modified algorithm significantly reduces the error rate of Porter's algorithm from 76.7% to 6.7% without compromising the efficacy of Porter's algorithm.

Keywords- Context-awareness; information retrieval; stemming, precision; recal.

1. INTRODUCTION

Stemming is a step in processing textual data preceding the tasks of information retrieval, text mining, and natural language processing. Thus, it is an important feature supported by present day indexing and searching systems. Stemming algorithms reduce different morphological variants to their base form (the stem). Stemming is used to enable matching of queries and documents in keyword-based information retrieval systems. This assumes that morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of IR applications. It is for this reason, that stemming *algorithms,* or s*temmers*, have been developed, which attempt to reduce a word to its *stem* or root form.

Thus, the keyword terms of a query or document are represented by their stems rather than by the original words, during the matching stage of information retrieval. However, it is often the case that morphological variations of stems can have unrelated contexts. For example, the word *"deliberation"* often refers to a logical discussion that is presumed to lead to a common understanding or position on a given issue.

African Journal of Computing & ICT Reference Format:

K.K. Agbele, A.O. Adesina, N.A. Azeez , & A.P. Abidoye (2012). Context-Aware Stemming Algorithm for Semantically Related Root Words. Afr J. of Comp & ICTs. Vol 5, No. 4. pp 33-42

Thus, the stem term of *"deliberation"*, "*deliberate"* means the act of performing a *"deliberation"*, but, it can also mean an act that is done intentionally and not by accident, or one that is done cautiously. Conversely, many stem word have other different, but semantically related terms. Many retrieved documents do not often convey the semantic context of the original query. For these reasons, stemmers are used in IR to reduce the size of index files. 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. In this context, the capability of search engines to find morphologically related terms with respect to every term would greatly enhance the IR effectiveness.

However, it is clear that the retrieval performance can go down drastically when stemming results in unrelated words being conflated to a single stem. For this reason, it is required that stem produced is very close to their root morphemes. Considering the fact that linguistic correctness of stems may become critical to effective retrieval in future, design of an intelligible root word has been proposed in this study in terms of context-awareness. This intelligible root word stemmer is an enhanced version of the popular affix stemmer, developed by [3]. The rule-based approach of Porter's stemmer has been considerably enhanced in context so as to give meaningful stems as output in as many cases as possible. Many studies have been conducted to evaluate the efficiency, accuracy and performance of the different types of stemming algorithms. The proposed Context-Aware Stemming (CAS) algorithm in context is intended for reducing the morphological variation of a given stream of query term through conflation; commonly called stemming.

The rest of the paper is organized as follows. *Section 2* presents and describes the existing stemming approaches in IR. *Section 3* gives the background of Porter's stemming algorithms. *Section 4* discusses the notion of context in IR. *Section 5* presents and discusses the proposed context-aware stemming algorithm. *Section 6* illustrates some evaluation of the results and we conclude the paper in *Section 7*.

2. CLASSIFICATION OF STEMMERS APPROACHES IN INFORMATION RETRIEVAL

Among the diverse approaches to stemming, the notable ones are Affix removal [1, 2, 3, 4, & 5], n-gram stemmers [6], HMM stemmer [7], YASS stemmer [8], Corpus-based stemmer [9], Context Sensitive Stemmer [10]. However, in this paper, we have used a rule-based (affix removal) to implement our CAS algorithm.

2.1. Affix Removal Approaches

This sub-section describes an overview of the state-of-the-art in the area of stemming algorithms. It covers basic ideas of ''classical'' (endings removal) techniques of inflectional and derivational suffixes. A number of stemming algorithms have been described in literature, noteworthy among them being [1, 2, 3, 4 and 5]. Lovins stemmer is a single pass and context-sensitive, which removes ending based on the longest-match principle. Dawson stemmer keeps the longest match and single pass nature of Lovins, and replaces the recoding rules, which were found to be unreliable, using instead an extension of the partial matching procedure also defined within the Lovins stemmer, while the Porter stemmer is a rule-based stemmer that eliminates the endings from a word based on a set of conditions in definite number of rules. Paice-Husk stemmer is a simple iterative method that removes the endings from a word in an indefinite number of steps. Finally, the Krovetz stemmer is a linguistic lexical validation stemmer. It is a very complicated low strength algorithm due to the processes involved in linguistic morphology and its inflectional nature. The most popular one among them is [3], which is more compact than [1]. We observe that all the approaches to stemming mentioned above not only ignore word meanings out of contexts, but also operate in the absence of any lexicon at all. The design goals of all of them seem to be better efficient and effective retrieval and compression performance and not the production of a linguistically correct root word. Our major objective is to maximize the proportion of the meaningful stems (root words) in our modified stemmer output for a given set of input words in context, without compromising the other performance measure.

2.2 Statistical Approaches

2.2.1. N-Gram Stemmer: This is a very interesting method and it is language independent. The string-similarity approach is used to convert word conflation to its stem. An n-gram is a set of n consecutive characters extracted from a word. The main idea behind this approach is that, similar words will have a high proportion of n-grams in common. For n equals to 2 or 3, the words extracted are called digrams or trigrams, respectively. There are n+1 such diagrams and n+2 such trigrams in a word containing n characters. The N-gram matching technique as one of the most common approaches to stemming was described by [2]. Besides, n-grams have been used in the automatic spelling correction on the assumption that the problems of morphological variants and spelling variants are similar. This stemmer has an advantage that it is language independent and hence very useful in many applications. The disadvantage is it requires a significant amount of memory and storage for creating and storing the n-grams and indexes and hence is not a very practical approach.

2.2.2. HMM Stemmer

This approach is based on the concept of the Hidden Markov Model (HMMs) which are finite-state automata where transitions between states are ruled by probability functions. At each transition, the new state emits a symbol with a given probability. This model was proposed by [7]. This method is based on unsupervised learning and does not need a prior linguistic knowledge of the dataset. In this method the probability of each path can be computed and the most probable path is found using the Viterbi coding in the automata graph. In order to apply HMMs to stemming, a sequence of letters that forms a word can be considered as the result of a concatenation of two sub sequences: a prefix and a suffix. A way to model this process is through an HMM where the states are divided into two disjoint sets: initial can be the stems only and the later can be the stems or suffixes. Transitions between states define word building process based on the assumptions made. 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 advantage of this method is it is unsupervised and hence knowledge of the language is not required. The disadvantage is it is a little complex and may over stem the words sometimes.

2.2.3. YASS Stemmer

This is an acronym for Yet Another Suffix Striper. This approach was proposed by [8]. According to the authors the performance of a stemmer generated by clustering a lexicon without any linguistic input is comparable to that obtained using standard, rule-based stemmers such as Porter's. This stemmer comes under the category of statistical as well as corpus based. It does not rely on linguistic expertise. Retrieval experiments on English, French, and Bengali datasets show that the proposed approach is effective for languages that are primarily suffixing in nature. The clusters are created using hierarchical approach and distance measures. Then the resulting clusters are considered as equivalence classes and their centroid as the stems. A study by [8] highlighted the details in the edit distance and YASS distance calculations for two string comparisons. The advantage of this method is based on corpus method and can be used for any language without knowing the morphology. The disadvantage is difficult to decide a threshold for creating clusters and requires a significant computing power.

2.3 Hybrid Approaches

2.3.1. Linguistic Lexical Validation Stemmer

A study by [5] presented the linguistic lexical validation stemmer. With the intention to decrease stemming errors and increase the stemmer's accuracy, Krovetz added a dictionary to validate the correctness of stems. The Krovetz stemmer performs a dictionary lookup after the removal of suffixes in a specific order. First, plural forms are converted to singular forms. Next, past tense words are converted to present. Finally, the "ing" suffix is removed. Besides, the Krovetz stemmer attempts to increase accuracy and robustness by treating spelling errors and meaningless stems. Spelling errors and meaningless stems are transformed into the closest word. Although this weak stemmer produced highly accurate results and a huge reduction in stemming errors, the addition of the dictionary lookup increased the complexity of the stemmer. 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.

2.3.2. Corpus Based Stemmer

This method of stemming was proposed by [9]. The authors suggested an approach which tries to overcome some of the drawbacks of Porter stemmer. For example, the words 'policy' and 'police' are conflated though they have a different meaning but the words 'index' and 'indices' are not conflated though they have the same root. Porter stemmer also generates stems which are not real words. Another problem is that while some stemming algorithms may be suitable for one corpus, they will produce too many errors on another. Corpus based stemming refers to automatic modification of conflation classes – words that have resulted in a common stem, to suit the characteristics of a given text corpus using statistical methods. The basic hypothesis is that word forms that should be conflated for a given corpus will co-occur in documents from that corpus. Using this concept some of the over stemming or under stemming drawbacks are resolved. A study by [9] used statistical measure to determine the significance of word form co-occurrence. The advantage of this method is it can potentially avoid making conflations that are not appropriate for a given corpus and the result is an actual word and not an incomplete stem. The disadvantage is that you need to develop the statistical measure for every corpus separately and this increases the processing time.

2.3.3. Context Sensitive Stemmer

This is a very interesting method of stemming unlike the usual method where stemming is done before indexing a document, for a Web Search, context sensitive analysis is done using statistical modelling on the query side. This method was proposed by [10]. 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. 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.

3. THE PORTER'S STEMMING ALGORITHM

Porter's Stemmer [3] uses suffix stripping in English language for stemming an input word. The algorithm operates in five steps. At each step the input word is transformed based on a list of rules. This stemmer is a linear step algorithm. 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 measure *m*, is used as the decision making variable.

Porter algorithm [3] 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 successfully 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. It is important to note that, in a particular step, it is sufficient if exactly one rule matches. It then proceeds to the next step.

Our work, instead, focuses on getting reasonable stemmer in terms of context-aware. We highlight some of the drawbacks of Porter's algorithm, from the perspective of getting understandable stems as output. Consider Table 1, which displays two list 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 understandable word in context. The stemmed word (GENER) is completely out of semantic context to the query (GENERAL).

Table 1: Words and the measure *m*

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

The main drawback 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 strengths 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 anxious is *anxiou*. This is one of the cases in which the suffix 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 to *gener* and iteration to *iter* make it difficult to relate them to dictionary entries. It also complicates the process of query enhancement. We observe that all the approaches to stemming mentioned not only ignore word meanings, but also operate in the absence of any lexicon at all. The design goals of these approaches seem to be better retrieval and compression performance and not the generation of a linguistically correct root word that is in context. Our main goal is to maximize the proportion of the meaningful stems (words) in the stemmer output for a given set of input words, without compromising the other performance measures.

4. CONTEXT IN INFORMATION RETRIEVAL

Some definition of context is greatly dependent on the field of applications, as shown by the analysis of 150 different definitions by [11]. In order to use context in developing context aware applications, the notion of context has to be well understood. A widely adopted definition of context for ubiquitous computing – which is also considered with context-aware IR defines context as any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves [12].

The central aspects in this definition are identity (user), activity (interaction with an application), location and time (as the temporal constraints of a certain situation). In the same article, Dey defines context awareness as a system that is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task [12]. Dey's definition of context points to potential sources of contextual information on a generic level. His definition of context awareness, however, is directly related to the view on context for IR taken in this study, which defines an IR task's context as any information whose change modifies the task's outcome, for instance (*stemming reduces any given query term(token), to a common base*). To achieve this goal, we develop an approach for context-aware IR that enables a system to provide relevant information to the user.

Research activities on context-aware IR have increased remarkably in recent years. The ubiquitous and pervasive computing communities have developed numerous approaches to automatically provide users with information and services based on their current situation [13] for an overview. Existing context-aware applications range from smart spaces [14] over mobile tour guides [15] to generic prototyping platforms [16]. Context has been successfully employed in a number of ways for personalization [17 & 18] and to improve retrieval from the Web [19 & 20], from email archives [21], from Wikipedia [22], as well as for ontology-based alert services [23], intention recognition [24]. Diverse approaches based on vector spaces have been proposed to enable context-aware information retrieval [25]. Maeda identifies context as one of his laws of simplicity, stating that "what lies in the periphery of simplicity is definitely not peripheral" [26] transferred to IR, this statement is a clear hint that the context of a retrieval task needs to be taken into account to make completing the task as simple as possible for the user.

Lawrence [27] provides an overview of the different strategies to make Web search context-aware. Yahoo introduced context-aware tools [20] that automatically extend the user's query by text from the Web site the user is currently visiting or from the file she is working on. Dumitrescu and Santini [28] present a context-aware IR method for documents. Contexts are traces of documents the user has been working on, which they represent as self-organizing maps [29]. The authors point out that the advantage of this approach is that context does not need to be represented by logical means. Semantic contexts are therefore an inevitable requirement to reason about contextual information.

5. CONTEXT AWARE MODIFICATIONS APPROACH TO PORTER'S ALGORITHM

We examine that the addition of more rules in order to increase the performance in one part of the vocabulary may cause deprivation of performance elsewhere. To achieve the goal, after a meticulous analysis, new suffixes and the context in which they must be removed have been identified and suitable changes have been made to the original Porter's stemming algorithm. Altogether, 31 modifications comprising of 15 additions, 11 replacements and 5 deletions have been carried out so that the Porter's stemming algorithm now consists of 65 rules compared to 60 in the original algorithm. The modifications were carried out in each step without changing the order suggested by Porter. The Porter's algorithm is a basic building block for the proposed context-aware stemmer. The suffix stripping method is rule-based. The following is the summary of modification steps used by the CAS algorithm.

Step 1 Rules: The first modified rule US → US, in step 1 of rule 1 (Table 2) of the stemmer helps to avoid the stripping of the suffix '*S*' from words like *generous, enormous,* and *anxious* and more.

Besides, another modification CEED → CEES was carried out on the same step so that stems given by the stemmer for words like *succeed* (as *success*), and *exceed* (as *excess*). In order to compensate for the earlier changes for words like *breed*, *greed* etc., and a new rule EED → EED was introduced. Next, another rule IES → Y replaces IES → I in the original Porter's algorithm to transform words like *Ponies* to *Poni*. The rule IED → Y was added to transform words like *tried* and *intensified* to *try* and *intensify*. Words of the form (*V*) ING → have been modified to (*V*) ING → E. So that the stems for words such as *scoring* → *score* and *guiding* → *guide* are stemmed accordingly. The rule (*V*) Y → I is deleted to transform words like happy to *happi*. A new rule ED → E is added to generate stems for words like *guided* (*guide*), and *demonstrated* (*demonstrate*) and more.

Step 2 Rules: The modifications in step 2 are described in the following. The rule ANCI → ANCE was modified into ANCY → ANCE. This rule was used to generate stems for *mendicancy* to *medicance*. The rule ENCI → ENCE is assumed to be replaced by ENCY → ENCY. This rule was used to consider cases like *urgency*, *frequency*, and *exigency*. The other modification in step 2 is ABLI → ABLE that is replaced with ABLY → ABLE to generate stems for words like creditably (as *creditable*), *miserably* (as *miserable*), and *notably* (as *notable*) and more. Likewise the rule OUSLI → OUS has been modified to OUSLY → OUS. Consequently, *callously, seriously,* and *dubiously* are stemmed into *callous, serious,* and *dubious* respectively. Words like *sensitivity*, *captivity*, and *productivity* can be stemmed into *sensitive*, *captive*, and *productive* using rule 14, IVITY → IVE. On modifying BILITI→BLE as BILITY→BLE, a word like *capability* is stemmed into *capable.*

Step 3 Rules: A new rule FULLY → FUL was added to stem words like *thankfully* to *thankful.* The rule FUL → removes the said suffix from the words completely. For example, *thankful* is stemmed into *thank*. The rule LESSLY → LESS stems words like *carelessly* to *careless* and *aimlessly* to *aimless*. A word like *possibly* is stemmed into *possible* using BLY → BLE. In this case, there are only two changes in step 3. A new rule LESS → is added to remove suffix '*less'* from *worthless* and *priceless*. For example, this rule removes the suffix *less* and generates worth and price. The rule ICITI → IC is modified into ICITY → IC to consider cases like *publicity* (*public*), *electricity* (*electric*) and *ethnicity* (*ethnic*).

Step 4 Rules: Changes in step 4 are described in the following. The rule AL → is modified into AL → E. This rule is to transform *agricultural* to *agriculture*, *arrival* to *arrive*, and *revival* to *revive* respectively. A new rule IABLE → Y was added to generate stems for words like *variable* (*vary*) and *identifiable* (*identify*). The rule ANCE → is discarded as it results in meaningless stems like *vari* for variance, and *appli* as appliance etc. The stems for words ending in scopic, such as *telescopic* (*telescope*) are obtained by applying the rule SCOPIC → SCOPE. Removal of IC suffix is 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*. The 'e' gets added as a result of a rule in step1 of original Porter's stemmer replacing 'ing' with 'e'. The rule TLY → T takes care of words like *diligently* (*diligent)* and *silently* (*silent*) etc which is not done in original Porter's stemming algorithm.

Step 5 Rules: The change in step 5 is the rule which removes the trailing E from words such as *possible* and *avoidable* have been removed. This is shown in Table 2. From Table 3, it can be seen that the modifications to Porter's stemming algorithm have led to the generation of better stem words in context. By context, we refer to the circumstances or situation in which a computing task takes place.

Table 2: Modified Rules

Step No.	Rule No.	Process	Porter's algorithm	Context-aware stemmer
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 → ABLE
	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 3: Comparison of stems generated by Porter's algorithm and CAS algorithm

Word (Input)	Porter stemmer (Output)	CAS (Output)
Probate	Probat	Probate
Gladly	Gladli	Gladly
Microscopic	Microscop	Microscope
Possibly	Possibli	Possible
Anxious	Anxiou	Anxious
Identifiable	Identifi	Identify
Thankfully	Thankfulli	Thank
Carelessly	Carelessli	Care
Purifying	Purifye	Purify
Biblically	Biblic	Biblical
Exceed	Excee	Excess
Capability	Capabiliti	Capable
Festivity	Festiviti	Festive
Diligently	Diligentli	Diligent
Ethnicity	Ethniciti	Ethnic
Guiding	Guid	Guide
Happy	Happi	Happy
Demonstrated	Demonstrat	Demonstrate
Callously	Callousli	Callous
Arrival	Arriv	Arrive
Effective	Effect	Effect
Falling	Fall	Fall
Generalization	Gener	General
Conditional	Condit	Condit
Archaeology	Archaeologi	Archaeology
Appointment	Appoint	Appoint
Allowance	Allow	Allow
Adoption	Adopt	Adopt
Formalize	Formal	Formal
Adjustable	Adjust	Adjust

5.1. Our Proposed (CAS) Algorithm

In this sub-section, we describe our Context-Aware Stemming (CAS) algorithm in detail. CAS algorithm was proposed to lessen the problems of traditional stemming approach that performs blind transformation of all query terms without considering the context of the stemmed word for effective search with regard to context awareness. The application flow involves modified *rule-based* approach that will add, replace, and remove query term matching endings in generating new stem words are illustrated in Figure 1.

The modified CAS algorithm uses a list of rules to reduce any given query term (token), such as word, to a common base. The rules are applied iteratively by matching a suffix string against query term endings. The rule is conditioned so that it may or may not apply to the query term depending on if the query term has already been modified by a previous rule. Each rule specifies removing, replacing, or adding characters to the end of the query term.

The ordering of the rules is important because a rule may be designed to act upon changes made by the previous rule. Each rule may be of arbitrary length and comprises of one or more instances of the five CAS algorithm rule components. The list of rules is processed iteratively until each has had a chance to apply itself to the query term. When considering the conditioned of the second or greater instance in a rule, the condition only applies to changes made to the rule.

Within each step, if a suffix rule matched to a word, then the conditions attached to that rule is tested on what would be the resulting stem, if that suffix was removed, in the way defined by the rule? Once a rule passes its condition and is accepted the rule applies and suffix is removed/replaced/added and control moves to the next step.

The CAS will enforce the following boundary conditions. The input must be greater than two characters in length and must contain at least one vowel and one consonant. The output must be greater than two characters in length. For example, the query word '*filing*'. After performing stemming process, the *-ING* ending is stripped such that the remaining word contains consonant-vowel-consonant (cvc) pattern i.e. '*fil*', using step 1 rule 6, we simply add '*e*', so that, '*fil*' becomes '*file*'. The new stemmed word '*file*' can be used in different contexts. It could be a tool used in smoothing, polishing or grinding. Moreover, it could be a folder or box that houses objects such as papers or cards. In this context, for this reason, the semantic of the original query (*filing*) has changed.

5.2. CAS Algorithm Rule Format

1. A suffix string (token endings) of 1 or more characters.
2. A condition ('Y' or 'N') indicating if the rule can be applied to the query term. If the condition is 'Y' and the query term has already been modified by another rule or this same rule (for rule with multiple instances) then rules cannot be applied to the token.
3. The number of characters to remove.
4. The string to add to the query term (token) consisting of 0 or more characters.
5. The number of characters to replace

5.3. The CAS Algorithm Description

In this section, we describe step by step method of our CAS algorithm as follows:

Step 1 Initialization:

Input the query term

Step 2 Select relevant text ending:

Examine the final letters of the query term;
Consider the first rule in the relevant ending for the input query term, and to indicate the first query term among stemming candidates.

Step 3 Check applicability of rule:

If the final letters of the query term do not match the ending rule, output stem, then terminate;
if the final letters of the query term and the ending rule matches, then goto 4;
if the final letters of the query term and the rule matches, and matching ending acceptability conditions are not satisfied, then goto 5;

Step 4 Apply rule:

Delete from the right end of the token the number of characters specified by the remove rules;
if there is an add string, then add it to the end of the query term specified by the rule;
if there is a replace string, then replace the number specified to the end of the query term;
if the condition specified is "no applicable rule" output the stem, then terminate;
if the condition specified is "match ending found" then take output to the next rule to access;
Otherwise goto 2.

Step 5 Search for another rule:

Go to the next rule in the rule engine database;
if the endings of the query term has changed, output stem, then terminate;
Otherwise goto 3.

Step 6 Termination Condition:

If matching endings acceptability conditions are satisfied, and then terminate the stemming process

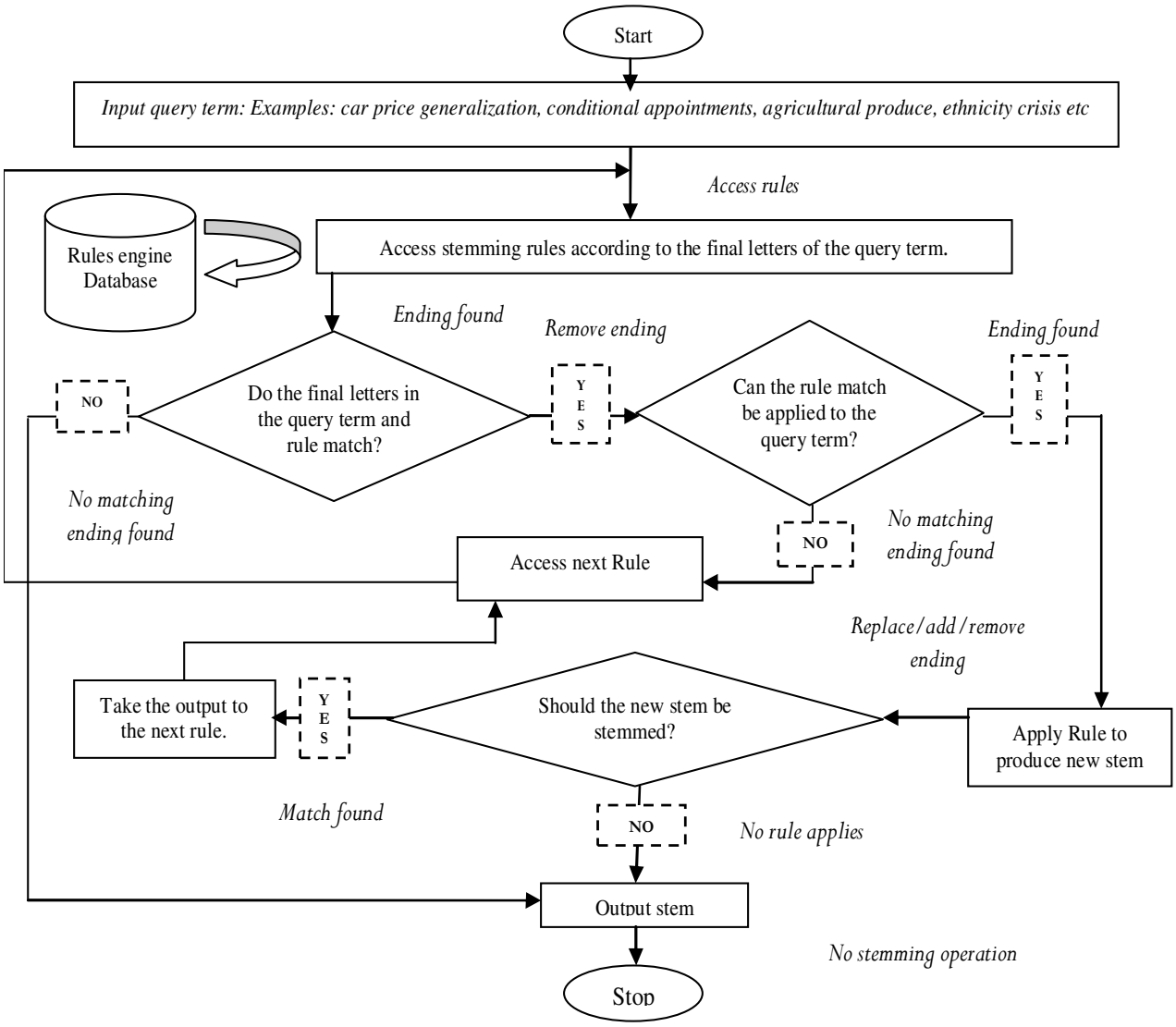


Fig 1: An Overview of the Proposed CAS Application Flow Architecture

6. EVALUATION OF RESULTS

Stemmer evaluations measures have been discussed and evaluated in literature [30, 31 & 32]. Among the notable criteria for judging stemmer performance, includes compression performance, retrieval performance, and correctness. The extents 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.

In this context, we derived meaningful stems from the words,which imply that the derived stems are linguistically correct when compared with most Porter's stem words that are obviously meaningless. We measured the performance of CAS algorithm output by finding the proportion (percentage) of meaningful stems generated by CAS algorithm and comparing that with the output of the original Porter's algorithm. Here we have taken a set of 30 English words used by Porter's algorithm as our working example excluding stop words. Our stem produced meaningful stems in 93.3% of the cases on an average while the original Porter's was successful in 23.3% cases as shown in Figure 2. Consequently, the error rate of our stemmer is 6.7% against 76.7% due to the Porter's algorithm. In this context, this improvement has been achieved without losing the efficiency of Porter's algorithm.

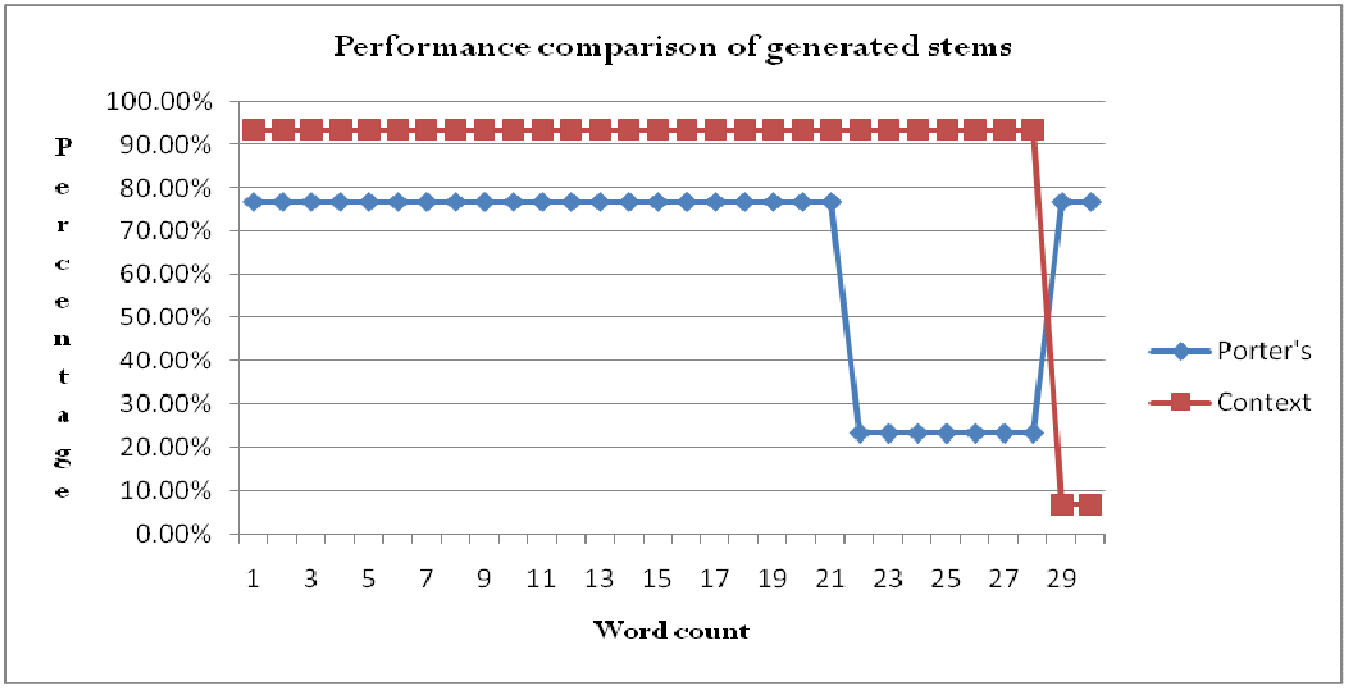


Fig. 2: Performance comparison

7. CONCLUSIONS

So far, none of the stemming algorithms give 100% output but is 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. The popular Porter's stemming algorithm was studied with an aim to generate understandable stem as output, in order to improve the effectiveness of information retrieval system.

By modifying the rule-based used by Porter's algorithm, the accuracy was improved from 23.3% to 93.3% in terms of the proportion of meaningful stems produced by CAS algorithm. The stemmer was introduced to address the traditional blind transformation of all query terms in the context of IR. Thus, the CAS algorithm conveys the semantics context of related terms by the original query in diverse contexts. In this context, CAS algorithm uses semantic knowledge to reduce stemming errors. The CAS technique can be effectively used in pre-processing

stages of text summarization and text classification systems in the context of information retrieval.

Acknowledgement

The authors wish to thank and appreciate colleagues within the research group for their helpful comments and suggestions at the implementation stage.

REFERENCES

[1] J. B. Lovins. (1968). Development of a Stemming Algorithm, Mechanical Translation and Computational Linguistics, vol.11, no. 12, pp: 22-31.

[2] J. Dawson. (1974). Suffix removal and word conflation. ALLC Bulletin, vol. 2, no. 3, pp: 33-46.

[3] M. Porter (1980). An Algorithm for Suffix Stripping. Program, vol. 14, no. 3, pp: 130 – 137.

[4] D. Paice Chris. (1990). Another Stemmer. ACM SIGIR Forum, Volume 24, No. 3, pp: 56-61.

[5] R. Krovetz. (1993). Viewing morphology as an inference process. In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, PA, USA – June 27th –July 01, 1993, pp: 191-202.

[6] G.E. Freund and P. Willet. (1982), 'Online identification of word variants and arbitrary truncation searching using a string similarity measure'. Information Technology: Research and Development, vol. 1, pp: 177-187.

[7] M. Melucci and N. Orio. (2003), A novel method for stemmer generation based on hidden Markov models. Proceedings of the 12th international conference on Information and knowledge management, New Orleans, LA, USA, Nov 03 – 08, pp:131-138.

[8] M. Prasenjit, M. Mandar, K. Swapan K. Parui, K. Gobinda, M. Pabitra and D. Kalyankumar. (2007). YASS: Yet another suffix stripper. ACM Transactions on Information Systems. vol. 25, no. 4, article 18.

[9] J. Xu, W.B. Croft, (1998). Corpus-based stemming using co-occurrence of word variants, ACM Transactions on Information Systems, vol. 16, no. 1, pp: 61-81.

[10] P. Funchun, A. Nawaaz, L. Xin and L. Yumao (2007), Context sensitive stemming for web search. Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval Amsterdam, July 23 – 27, pp: 639-646.

[11] M. Bazire and P. Brézillon. Understanding Context Before Using It (2005). In A. K. Dey, B. N. Kokinov, D. B. Leake, and R. M. Turner, editors, Modelling and Using Context, *5*th International and Interdisciplinary Conference (CONTEXT *2005*), volume 3554 of Lecture Notes in Computer Science, pp: 29–40. Springer, July 2005.

[12] A. K. Dey. Understanding and Using Context (2001). Personal Ubiquitous Computing, vol. 5, no. 1, pp: 4–7.

[13] S. Loke. Context-Aware Pervasive Systems: Architectures for a New Breed of Applications. Auerbach Publications, 2006.

[14] X. Wang, J. S. Dong, C. Chin, S. Hettiarachchi, and D. Zhang (2004). Semantic Space: An Infrastructure for Smart Spaces. IEEE Pervasive Computing, vol. 3, no. 3, pp: 32–39.

[15] S. Chou, W. Hsieh, F. Gandon, and N. Sadeh (2005). Semantic Web technologies for context-aware museum tour guide applications. In Advanced Information Networking and Applications, *2005*. AINA *2005*. *19*th International Conference on, vol. 2.

[16] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen. Context-Phone (2005): A Prototyping Platform for Context-Aware Mobile Applications. IEEE Pervasive Computing, vol. 4, no. 2, pp: 51–59.

[17] H. R. Kim and P. K. Chan (2008). Learning implicit user interest hierarchy for context in personalization. Applied Intelligence, vol. 28, no. 2, pp: 153–166.

[18] C. Keßler, M. Raubal, and C. Wosniok (2009). Semantic Rules for Context-Aware Geographical Information Retrieval. In P. Barnaghi, editor, *4*th European Conference on Smart Sensing and Context, EuroSSC *2009*, volume 5741 of Lecture Notes in Computer Science, pp: 77–92. University of Surrey, Springer.

[19] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin (2001). Placing search in context: the concept revisited. In WWW '*01*: Proceedings of the *10*th international conference onWorldWideWeb, pp: 406–414, New York, NY, USA, ACM Press.

[20] R. Kraft, C. C. Chang, F. Maghoul, and R. Kumar (2006). Searching with context. In WWW '*06*: Proceedings of the *15*th international conference onWorldWideWeb, pages 477–486, New York, NY, USA, ACM Press.

[21] W. Weerkamp, K. Balog, and M. de Rijke (2009). Using Contextual Information to Improve Search in Email Archives. In Advances in Information Retrieval. *31*st European Conference on Information Retrieval Conference (ECIR *2009*), pp: 400–411.

[22] A. Ukkonen, C. Castillo, D. Donato, and A. Gionis (2008). Searching the Wikipedia with Contextual Information. In CIKM '*08*: Proceedings of the *17*th ACM conference on Information and knowledge mining, pp: 1351–1352, New York, NY, USA, ACM.

[23] A. Leonidis, G. Baryannis, X. Fafoutis, M. Korozi, N. Gazoni, M. Dimitriou, M. Koutsogiannaki, A. Boutsika, M. Papadakis, H. Papagiannakis, G. Tesseris, E. Voskakis, A. Bikakis, and G. Antoniou (2009). AlertMe: A Semantics-based Context-Aware Notification System. In *33*rd Annual IEEE International Computer Software and Applications Conference, pp: 200–205. IEEE.

[24] P. Kiefer and C. Schlieder (2007). Exploring context-sensitivity in spatial intention recognition. In Proceedings of the Workshop on Behavior Monitoring and Interpretation (BMI'*07*), pp: 102–116.

[25] M. Melucci. A basis for information retrieval in context (2008). ACM Trans on Info Sys, Vol. 26, No. 3, pp: 1–41.

[26] J. Maeda. The Laws of Simplicity (Simplicity: Design, Technology, Business, Life). The MIT Press, August 2006.

[27] S. Lawrence. Context in Web Search (2000). IEEE Data Engineering Bulletin, vol. 23, no.3, pp: 25–32.

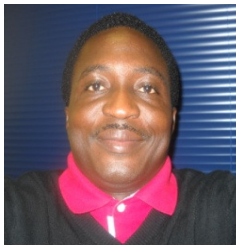
[28] A. Dumitrescu and S. Santini. Think locally, search globally; context based information retrieval (2009). In International Conference on Semantic Computing, pages 396–401, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[29] T. Kohonen. Self-Organizing Maps. Springer, 3rd edition, 2000.

[30] J. B. Lovins (1971). Error evaluation for stemming algorithms as clustering algorithms. Journal America Social of Information Science, vol. 22, pp: 28-40.

[31] C. D. Paice (1994). An Evaluation Methods 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.W. Kraaij and R. Pohlmann (1996). Viewing stemming as recall enhancement. Proceedings of the 17th ACM SIGIR Conference, Zurich, August 18-22, pp: 40-48

Authors' Briefs



Kehinde Kayode AGBELE is a Lecturer at Department of Mathematical Sciences (Computer Science Option), EKSU, Ado-Ekiti, Nigeria. AGBELE received B.Sc (Hons) degree in Computer Science from Ondo State University (now Ekiti State University), Ado in 1997, and M.Tech degree in Computer Science from the Federal University of Technology, Akure, in 2005. Currently, AGBELE is a Doctoral Research student at University of the Western Cape, Computer Science Department (Machine Learning and Intelligent Systems Research Group), Cape Town, South Africa. His research interests include Information Retrieval, Data Mining, Text Mining, Web Search Engine, Agent Technology, Pattern Classification and Clustering, Ubiquitous Healthcare, ICTs & Applications. He can be reached by phone on +27789345755 and through E-mail at agbelek@yahoo.com.

Ademola Olusola ADESINA, a lecturer of Computer Science in Lagos State University (LASU), obtained his First Degree in Computer Science from Ogun State University (now Olabisi Onabanjo University), Ago-Iwoye and Masters Degree in Computer Science from the University of Ibadan, Nigeria. He is presently a doctoral student at the University of the Western Cape, Department of Computer Science (Machine Learning and Intelligent Systems Research Group), Cape Town, South Africa. His research interests are in Mobile computing, Text Processing, Agent Technology, Information Retrieval, Web Search and Mobile security. Email: inadesina@gmail.com

Azeez Nureni Ayofe graduates with B.Sc. (Hons) degree in Computer Science with Second Class (Hons.) Upper Division, from the Federal University of Technology, Akure (FUTA), Ondo State, Nigeria in 2004. He proceeded in 2006 to the University of Ibadan, Oyo State, Nigeria, after completing his National Youth Service Corps (NYSC) programme, for his Masters Degree programme in Computer Science which he successfully completed in 2008. He is currently a PhD research student in Computer Science at the University of the Western Cape, South Africa. He was a lecturer in the Departments of Computer Science of Crescent University, Abeokuta, Ogun State, Nigeria and the Fountain University, Osogbo, Osun State, Nigeria between 2008 - 2010. His areas of research include security and privacy; Grid and Cloud computing, knowledge representations and Computer Education & Applications. He can be contacted on +277 3 899 1735; nurayhn@yahoo.ca; and 3008814@uwc.ac.za.

Abidoye Ademola Philip received B.Tech degree from Federal University of Technology Akure, Ondo State, Nigeria in 2001. He went further for his master's degree (M.Sc.) in Computer Science from University of Ibadan, Oyo State, Nigeria and completed it in 2006. He is presently a PhD student in Computer Science at University of the Western Cape, Cape Town South Africa. He is working as a Lecturer in the Department of Computer Science, Lagos State University, Ojo Nigeria. He has attended both local and international conferences, written many papers published in reputable international journals. His research areas include wireless sensor network, energy optimization, security, and mobile health. He can be contacted through Email: ademaola.abidoye@gmail.com