Design of Lightweight Stemmer for Odia Derivational Suffixes

Dhabal Prasad Sethi
Lecturer in Computer Science & Engineering
Government College of Engineering, Keonjhar, Odisha, India

Abstract: Stemming plays an important role in morphological analyzer, information retrieval system and others. Stemming is the process of reducing all forms of word to its base form or stem. Derivational stemming is the process of removing the derivational suffixes from its derived word and get the stem/root word. Using stemmer the information retrieval system becomes faster. In this article, I designed the lightweight stemmer algorithm for Odia derivational suffixes and later that algorithm compared with simple suffix stripping algorithm.

Keywords: Derivational Suffixes, Stemmer, Lightweight, Krudanta, Tadhita, Information Retrieval

I. INTRODUCTION
Stemming is a process that conflates morphological forms into a single common form called stems or roots without doing complete morphological analysis. The term conflation means mapping variants of a word to a single term or stem. The most important tool used in information retrieval system (IRS) and morphological analyzer is a stemmer who reduces a word to its stem form to improve the system performance. In Information Retrieval (IR), it doesn’t mean stem generated are genuine words or not. The word “computation” might be stemmed to “comput” rather than “compute”. “Comput” is the stem form of a word where “compute” is the lemmatize form of word. An algorithm which converts a word form to its linguistically correct root is called lemmatize.

Stemming algorithm can be classified into two categories 1) stem-based2) root-based. In stem-based algorithm removes prefixes and suffixes from words where root-based algorithms reduce stems to root. The researches named Al-Jlayl and Frieder [5] shown that stem-based retrieval is more effective than root-based. In root-based algorithm many surface word variants don’t have similar semantic interpretation. These surface words are different in meaning; they originate from the same root. So it increases the word ambiguity.

In IR System, when the user enters the query word “fishing” as input, he actually wants to retrieve documents containing the related term “fisher” and “fished”. Thus using stemmer the system improves the recall rate i.e. the number of documents retrieved in response to a query. It also decreases the size of index file on IRS (Information Retrieval System), since many related terms are mapped to one. The applications of stemmer are machine translation, document summarization and text classification.

The rest part of this paper is organized as: Section II describes its related work, Section III describes the Odia derivational morphology, Section IV describes as Odia lightweight stemmer algorithm, Section V describes as result and experiment, Section VI describes as conclusion and future work.

II. RELATED WORK
First the stemmer algorithm is developed by Lovins in 1968 for English language. His algorithm is based on dictionary based and the suffix list, root words are stored in dictionary. when word is checked for suffix removal, right-hand end of word is matched with suffix dictionary, if found then removed the suffix and then matched with root dictionary. The Lovins algorithm contains 294 suffixes and 19 context-sensitive rules to remove the suffixes. Then in 1980 porter stemmer developed suffix stripping algorithm for stemmer. That algorithm gives better result than Lovins because minimal rules are applied e.g. 5 steps are applied for word transformation, approximately 60 rules are used for programmer. That onwards all the foreign language and Indian languages used the same technique for their language...
and get better result for stemming. In India not much work has done about stemmer. But now in India stemmer for Gujarati, Bengali, Hindi, Tamil, Odia have developed. Among the Indian stemmer algorithm are implemented in inflectional or derivational words. Rare works have done both the stemmer for inflectional and derivational types.

For Indian languages: Juhi Ameta, Nisheeth Josi, Iti Mathur [4] presented a paper named “A light weight stemmer for Gujarati”. They have shown the creation of rules for stemming and morphology that Gujarati possess. Md. Zahurul Islam, Md. Nizam Uddin and Mumit khan [2] presented a paper named “A light weight stemmer for Bengali and its use in spelling checker”. They have used the suffix stripping algorithm according to the longest suffix. Ananthakrishnan Ramanathan, Durgesh D Rao [1] presented a paper named “A light weight stemmer for Hindi”. They have removed the suffix from each word by longest possible suffix.

Kartik Subha.Dipti Jiangani, Pushpak Bhattacharyya [8] presented a paper named Hybrid inflectional stemmer and Rule-based derivational morphology for Guajarat. They have used the rule-based technique for derivational suffixes and POS based and suffix-stripping algorithm for inflectional suffixes.

For Odia languages: Sampa Chauatnaik, Sohang Sunder Nanda, Sanghamitra Mohanty [9] presented a paper named “A suffix stripping Algorithm for Odia stemmer”. They have used the suffix stripping algorithm to remove the inflectional suffixes from noun, and verb.

R.C Balbantray, B Sahoo, M.Swain, D.K Sahoo [10] presented” IIIT-Bh FIRE submission: MET Track Odia”. They have used the affix removal technique.

III. ODIA DERIVATIONAL MORPHOLOGY

a) Odia Morphology:
Odia morphology deals with analysis, identification and description of structure of morpheme. In Odia, morphemes are called Rupeme (ରୁପେମେ). Morpheme (Rupeme) is the smallest component/unit of Odia language which carries and conveys a unique meaning and is appropriated by grammar.

For example the word BALAKAMANE the morphemes are BALAKA, MANE. Morpheme is not necessary to form a meaningful word in Odia. Every morpheme is a base/root word, prefix or suffix. Morphemes are classified into five types:1) free morpheme,2) bound morpheme,3) complex or combined morpheme,4) mixed morpheme,5) marker morpheme. Here I have explained only free and bound morpheme. That morpheme which are independent called free morpheme. These morphemes are stand alone without other. It does not need to add with other to create a word. Example

Here the morpheme BHATA can stand alone without morpheme (KU).

Those morphemes added with another morpheme and give meaningful word are called bound morpheme. In Odia languages most of morphemes are bound type.

b) Study of Odia Derivational (Tadhit and Krudanta) Words and Suffixes (Prataya) Those Odia words are derived from Sanskrit verbal root with addition of suffixes and are used in Odia language; these words are called “TATSAMA KRUDANTA” word. Example Darshana is derived form drush dhatu; Patha is derived from path dhatu. Those Odia words are derived from Odia verbal root and the Odia verbal root are derived from Sanskrit verbal root, these Odia words are called TATABHABA KRUDANTA WORD. Example KANDANA is derived from Odia dhatu KANDA which is derived from Sanskrit KRANDA dhatu. The Odia words which are derived from Odia verbal root with addition of suffixes that words are called DESAJA KRUDANTA WORD. Example the Odia word KHASANDA word is derived from Odia dhatu KHASA with addition of suffix ADA. Those suffixes are added with root word to create new word, these suffixes are called TADHITA SUFFIXES and newly generated words (which are different part of speech) are called TADHITA WORD. The tadhita word KULINA is derived from root word KULA with addition of suffix (INA), KOULIKA word is derived from root word KULA with addition of IKA.

c) Odia Derivational Morphology:
Derivational morphology deals with the addition of derivational suffixes with word stem to form word of different class (different part-of-speech).

Like English, Odia derivational suffixes are added with root word to form different part-of-speech. They are

1) Noun to Adjective
Noun to adjective means when the derivational suffix added with noun word it changes to adjective category. It changes its part of speech. Here is the example

RUPA+ELI=PUPELI

2) Adjective to Noun
Adjective to noun means derivational suffixes added with adjective word to form noun word (which is the derived form).

ADHUNIKA+TA=ADHUNIKATA

3) Adjective to Adjective
Adjective to adjective means when some adjective words which when added with derivational suffixes it changes its part of speech e.g. adjective word+ suffix=adjective word(derived form).
SARU+A=SARUA

4) Verb to Adjective Verb to adjective means when derivational suffixes added with verbal root word it changes its meaning. E.g. verbal word+ suffixes=Adjective word (derived form).
PAN+IYA=PANIYA

5) Verb to Noun Verb to noun means some verbal word which when added with some derivational suffix, it changes its part-of-speech e.g. it converted verb to noun word.
BINCH+ANA=BINCHANA
KHEL+ALI=KHELALI

In derivational stemming, words that are derived, either by adding affixes to that stems or by performing changes at the morpheme boundary, are reduced to their stem form [8].

IV.ODIA LIGHTWEIGHT STEMMER ALGORITHM

The simple suffix stripping algorithm for Odia derivational suffixes is:

START
Step1: Input the derived word to be stemmed
Step2: If derived word is matched with one of suffix list, remove the suffixes
Step: Get the stem
STOP

Implementing the above algorithm I found many over-stemming and under-stemming errors. Over-stemming means those words are not morphological variants are conflated. Under-stemming means those words are morphological variants are not conflated. Example of over-stemming, if the user wants to remove the suffix UA before removing UA, because in the two suffixes the character AA is matched.

To solve this over-stemming problem, I designed a prototype of Odia lightweight stemmer that removes only derivational suffixes. I identify a set of suffixes that should be removed based on the grammatical functions of suffixes. I generate three derivational lists of suffixes which consist of one character, two characters and three characters respectively. In the Odia derivational suffix list, I identify 18 one character, 55 two character and 7 three character suffixes that should be removed recursively in stemming. To remove the prefixes in word, the algorithm works non-recursively. Since my work is to remove only derivational suffixes from words, it uses recursive technique. The new enhanced version of suffix stripping algorithm is

START
Step1: Enter the derived word to be stemmed
Step2: Recursively removes the three character suffixes in the order of presentation if word length greater than three before removing the suffixes.
Step3: Recursively remove the two character suffixes in the order of presentation if the word length greater than two before removing the suffix.
Step4: Recursively remove the one character suffixes in the order of presentation if the word length greater than two before removing the suffixes.
STOP

Again I explained the above new algorithm clearly. When the user enters the derived word as input, it removes longest length suffix (three characters), two character and then one character recursively and gets the stem.

V.RESULT AND EXPERIMENT

I have taken 20 derived words for each suffix in testing. There are 80 derivational suffixes in Odia language approximately e.g. 1600 derived words are taken for testing out of which using simple suffix stripping algorithm predicted 1060 words (53suffixes) correct stem, 300words (15suffixes) are over-stemming, 240words (12suffixes) did not stem due to difficulty word construction. So the accuracy of stemming for derived word is 66.25%, over-stemming error percentage is 18.75% and 15%words did not stem. Using lightweight stemming algorithm the accuracy of correct stem is 85%, because this algorithm solves over-stemming problem and 15%did not stem. Fig2 show the percentage of correct stemming using simple suffix-stripping algorithm where fig3 shows the percentage of stemming using lightweight algorithm.
VI. CONCLUSION

In this article I designed a lightweight stemmer algorithm for Odia which removes derivational suffixes from derived word. The algorithm removes the suffixes recursively first three character, then two characters and last one character which is a new enhanced approach of simple suffix removal technique. Implementing the simple suffix stripping algorithm for derived word gives the result 66.25%, whereas the proposed algorithm predicts 85% words correct stem approximately. The future work is to use this algorithm in different stemmer.

ACKNOWLEDGEMENT

I would like to thanks my school teacher named Padmanava Rout who was teaching the Odia language, has given his valuable time to prepare the article. I would like to thank all my well wishers.

REFERENCES

[1] A lightweight stemmer for Hindi” developed by Ananthakrishan Ramanathan from national center for software technology, rain tree marg, sector 7, CBD Belapur, Navi Mumbai and Durgesh D Rao from DR system, S-27, Lane 1, Sector 9, CBD Belapur, Navi Mumbai 400614, India.


[6] “Building an Arabic stemmer for information retrieval” by Aitao chan, school of information management and systems, university of California at Berkeley, CA 94720-4600, USA and Fredric Gey, UC Data Archive & Technical Assistant, University of California at Berkeley, CA 94720-5100, USA.

[7] “Design and Development of stemmer for Tamil Language: Cluster Analysis” by M. Thangarasu, Dr. R. Manavalan, Department of Computer Science and Application, K. S. Rangasamy college of Arts and Science, Tiruchengode, India.


BIography

Dhabal Prasad Sethi is working as a lecturer in Computer Science & Engineering at Government College of Engineering, Keonjhar, Odisha. He has completed his Bachelor of Engineering in Computer Science & Engineering from BIET, Bhubaneshwar in 2006. He has completed his Master of Engineering in Computer Science & Engineering with specialization in Knowledge Engineering from PG Department of Computer Science and Application, Utkal University, Bhubaneswar in 2011. He has more than two years teaching experience. He has presented two numbers of papers in international journals. This is his third numbers of his own credit. His research area of interest is Natural Language Processing, Information Retrieval, Data Mining and Software Engineering.