Development and Integration of an Odia Stemmer in Dspace for Odia Search Engine

Gouranga Charan Jena, Siddharth Swarup Rautaray

Abstract: Stemmer is used for reducing inflectional or derived word to its stem. This technique involves removing the suffix or prefix affixed in a word. It can be used for information retrieval system to refine the overall execution of the retrieval process. This process is not equivalent to morphological analysis. This process only finds the stem of a word. This technique decreases the number of terms in information retrieval system. There are various techniques exists for stemming. Here a new hybrid stemmer has developed named as “Mula” for Odia Language. It is a combination of brute force and enhanced suffix stripping approach for Odia language. The new born stemmer is both computationally inexpensive and domain independent. We have integrated this stemmer in existing Dspace for Odia text retrieval System. The results are commendable and suggest that the new stemmer can be used effectively in Odia Search Engine. The proposed stemmer also handles over-stemming and under-stemming effectively.

Keywords: Derivational Suffixes, inflectional words, Odia Stemmer, Information Retrieval, Brute force, DSpace etc.

I. INTRODUCTION

Stemming is a technique for removing inflectional ordered word to its stem. This technique is used to remove the suffix or prefix affixed in a word. This process finds stem of a word. Stemmer is essential for information retrieval system to refine the performance of the system. It technique is not equivalent to morphological analysis. Its primary objective is to decreases the number of terms in an information retrieval system. Stemming technique can be used in information retrieval to decrease as many related words to a common form that is not in base form. For example, the English word “Computation” has different inflections such as “Comput”, “Compute”, “Computing”, “Computes” etc. In this case stemmer can be used to reduce derived words into its root or stem word. Many stemmers had been developed for different languages, which reduce a word to its root/stem form. It ultimately reduces the index file size in an information retrieval system. In this way we can improve recall (i.e. the number of documents retrieved in response to a query.) of an IR system by effectively using stemmer in the background. Since many derivational words are mapping into one word i.e. root or stem. It ultimately reduces the volume of the index files in the IR system. There are several types of stemming algorithms exists and it differs in respect to their performance and accuracy. There are various algorithms used to find a stem of a word i.e. (a) Brute-force algorithms: It uses a lookup table that contains derived words with their corresponding roots.

To find the root/stem of a word, the table is queried to find a matching inflection word. If a matching inflection word is found, then corresponding root returned. Otherwise, it fails. (b) Suffix-stripping Approach: It does not depend on any lookup table that consists of derived or inflected words and their root word relations. It simply uses a set “rules” which drives the algorithm. It finds the root/stem of the given input word based on that rules. (c) Lemmatization algorithms: This technique also called as text normalization. In Lemmatization root word is called Lemma. The POS is first identified of that language and then an attempt will be made to find the stem. The stemming rules will change based on a word’s POS of that language. Lemmatization process ensures that the root or stem of the inflected words belongs to that language (d) Stochastic algorithms: It is based on probability method to detect the root form of a word. This trained on a table of root words to inflected words relations to develop a probabilistic model. (e) Hybrid approach: This technique combines more than two methods as discussed above. It may merge the rule-based technique along with the probability method. (f) N-Gram Modeling: Many stemming methods used in the n-gram technique of a word to select the correct stem for a word.

Stemming plays a vital role to handle the vocabulary mismatch problem of anIR system. In this said problem, the query words mismatch with the document words. For example, when a user input a query word and the word does not existin the vocabulary of the document then it may cause unreliable result. To avoid this problem we have developed a new web-based hybrid stemmer using brute force with enhanced suffix stripping algorithm union that can be adopt in the Odia information retrieval system. The new stemmer is both computationally fasteras well as domain-independent.

II. RELATED WORKS

In the study of information retrieval, researchers find stemming plays an important role. Stemming is not a new concept. Stemming techniques had invented since 1968. The first stemming algorithm was designed by Julie Beth Lovins[1]. After that many researchers continued investigating various approaches to this area of study and proposed several algorithms to improve its performance. Another stemmer in English was written by Martin Porter [2] in the year 1980. As compared to European languages as well as English,
Development and Integration of an Odia Stemmer in Dspace for Odia Search Engine

a few researches have been discovered in Indian Language.

A Hindistemmer [3] was proposed by Rao, Durgesh et al. based on suffix stripping approach.

A Bengali Morphological analyzer [4] was developed by Dasgupta et al. based on suffix stripping approach. Stemming is the process by which the user inputs an inflected word to the trained model and the model produces the root/stem word according to its rule set. In this paper we have developed a Stemmer based on Hybrid Approach.

III. LITERATURE SURVEY ON ODIA STEMMER

| Reference          | Key Findings                                                                                     |
|--------------------|--------------------------------------------------------------------------------------------------|
| Sampa et al. [5]   | Published a paper on Stemmer for Odia language. They used the suffix stripping approach to remove the inflectional suffixes. The limitation of this algorithm was it only predicts 88% accuracy. |
| Balbantray, R.C. et al. [6] | Published a paper FIRE 2012 Submission: MET Track Odia. They had used the affix removal algorithm. The system reads input text files from the folder. Firstly, it removes stop words from the input files against the stop word dictionary then matched the token with the root word dictionary. After that the input matched the suffix dictionary then removes the suffix and match with the root word. If the root word found then there is no further processing required. |
| Balabantaray, R.C. et al. [7] | Presented a paper on Odia Text Summarization.                                                                 |
| Sethi, Dhabal Prasad [8] | Published a paper on Lightweight Stemmer for Odia Derivational Suffix. He used suffix stripping method to find the stems. |

IV. ODIA DERIVATIONAL MORPHOLOGY

a) Odia Morphology:

The formal variants of a morpheme are called allomorphs of that morpheme. The variant may be phonologically or morphologically conditioned. A morpheme may be a free or a bound form. Alternatively we can say that a word consist of one or more than one morpheme. From the point of view of its internal structure, a word may consist of (i) a root morpheme only (ii) a root and one or more non root morpheme or (iii) more than one root morpheme. The non-root morphemes are bound forms and are generally referred to as affixes. Roots enter into further morphological constructions and form a base while non-roots do not. [9]

Odia morphology deals with the analysis, identification and description of structure of morpheme. Morphology deals with the structure of words. The basic unit is the focus of study in morphology is morpheme.

Example: The word ରାମ ଭାତ ଖାଉଛି the morphemes are ରାମ, ଭାତ, ଖାଉଛି. Morpheme is not always conveying a meaningful word in Odia. Any morpheme in Odia should be a root word, prefix or suffix. Morphemes are divided into five categories shown in fig I.

Fig. 1. Types of Morpheme

The morpheme which are independent called free morpheme. Those morphemes are standalone in nature. It does not need to add with other to create a word. Examples offree and bound morpheme discussed below.

The morpheme ରାମ is a stand-alone morpheme and morpheme (ଭାତ) is a suffix. Most of the morphemes are bound type in Odia language.

An affix is a morpheme that may come at the beginning (Termed as Prefix) or the end (Termed as Suffix) of a base morpheme. In Odia, prefixes are bound morphemes are affixes that come before a base morpheme. For example: ରାମଭାତଖାଉଛି = /ରାମ/ + /ଭାତ + /�ାଉଛି/

b) Odia Derivational Morphology:

Odia morphemes of different types such as nouns, verbs, affixes, etc. combine to form new words. The suffixes can added with root word to form POS shwonin the below Table- I.

| Categories      | Examples                                                                 |
|-----------------|--------------------------------------------------------------------------|
| Noun to Adjective | Noun word + Derivational suffix = Adjective category. ସରୁଆଧୁନିକତା + ରୁପେଲି = ସରୁଆଧୁନିକତାରୁପେଲି |
| Adjective to Noun | Adjective word + Derivational suffix = noun words ଏଶରୁଆଧୁନିକତା + ରୁପେ + କୂଳ = ଏଶରୁତରୁପେ + କୂଳ |
| Adjective to Adjective | Adjeotive word + Suffix= Adjective word ସରୁରସୁଡ଼ଅସୁବ + ଏଶରୁ = ସରୁଏଶରୁ
In the suffix-stiffing approach of stemming, we remove suffixes or affixes attached to that stem. As Odia is being influenced by Sanskrit. So Odia language has strong inflectional system.

According to the Panini grammar rules, we are representing some rule in below example.

For example ‘ଜଣପଦିତି’ here stem ଜଣିତ and suffix is ପଦିତି.

The nominal suffix and verbal suffix of Odia language are described below. (Table – 1&2) [10].

### Table- II: List of Nominal Suffixes in Odia

| Inflection | Singular (Case) | Plural (Case) | Relationship |
|------------|----------------|---------------|--------------|
| ପଦିତି (1st inflection) | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି |
| ପଦିତି (2nd inflection) | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି |
| ପଦିତି (3rd inflection) | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି |
| ପଦିତି (4th inflection) | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି |
| ପଦିତି (5th inflection) | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି |
| ପଦିତି (6th inflection) | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି |
| ପଦିତି (7th inflection) | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି, ପଦିତା, ପଦିତ, ପଦିତାନା | ପଦିତି |

### Table- III: Odia Verbal Suffix

| (Tense) | (Person) | (Singular Suffix) | (Plural Suffix) |
|---------|----------|-------------------|----------------|
| ଏଭାନ୍ତି (Present Tense) | ସାଧନଯୁର୍ଧବ | ସାଧନଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ, ସାଧନଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ | ସାଧନଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ |
| ଏଭାନ୍ତି (Past Tense) | ସାଧନଯୁର୍ଧବ, ସାଧନଯୁର୍ଧବ, ସାଧନଯୁର୍ଧବ | ସାଧନଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ | ସାଧନଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ |
| ଏଭାନ୍ତି (Future Tense) | ସାଧନଯୁର୍ଧବ, ସାଧନଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ | ସାଧନଯୁର୍ଧବ, ସାଧନଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ | ସାଧନଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ, ସାଧନ୍ଯୁର୍ଧବ |

There are some prefixes which are attached to noun only in Odia language. There is 20 such prefixes exist in Odia. These are inherited from Sanskrit. These are as follows (Table -3):

### Table- IV: List of Odia Prefixes

| Prefix | Prefix | Prefix | Prefix |
|--------|--------|--------|--------|
| ପ୍ରେକ୍ଷଣ | ପ୍ର୭ୟୋଜନ | ପ୍ରୁକ୍ତ | ପ୍ରେକ୍ଷଣ |
| ପ୍ରେକ୍ଷଣ | ପ୍ର୭ୟୋଜନ | ପ୍ରୁକ୍ତ | ପ୍ରେକ୍ଷଣ |
| ପ୍ରେକ୍ଷଣ | ପ୍ର୭ୟୋଜନ | ପ୍ରୁକ୍ତ | ପ୍ରେକ୍&<br>
character from the derived words.

**Fig.2. Flowchart of Proposed Odia Stemmer Algorithm**

Brute force search is called as exhaustive search. It searches all the possible solution from the data. Here it searches the root words present in the database. This technique uses a lookup table which contains inflected words and root words mapping. In this technique [13] we create and store maximum possible inflected words along with their corresponding root word in a database table. When we give input to the system then brute force search is carry out and it inspects that whether the derivational words exist in the database. If the word is present in the table then it will give its corresponding stem or root word. If the word is not present in the table then it will go for suffix removal method to handle those words. Suffix removal is a rule based approach in which certain rule set is defined. By applying those rule set suffixes are removed from the inflected or derived word, to find the stem/root. [14] The new enhanced approach of suffix stripping algorithm is:

**Start**

Step 1: Enter derivational word that to be stemmed

Step 2: The system removes the 3 character suffixes, 2 character suffixes and 1 character suffixes from the derivational word if word length greater than three, and two respectively recursively.

**End**

The inflected word is processed by the stemmer in three steps. The steps are shown below.

**Input:** The inflected Odia word/paragraph is entered as an input to the web-based system. Here “ଉଳାପିତ୍ତି” is given as an Input word.

**Processing:** Derivational/inflected word is searched by brute force method. It matches with the user searched word with the words exist in the database table. If the matching word is exist then it will provide the stem of the word as output. If mismatch found then it search for the alternate method called suffix stripping method i.e. the algorithm removes the suffixes recursively first 3 characters, then 2 characters and last 1 character with a condition that the inflected word must be greater than the suffix to find the stem/root of the word.

**One Character derivational Suffix:** ଉଳାପିତ୍ତି, ଉଳାଷ୍ଣୁ, ଉଳାରା, ସନ୍ତୁ‌ରା

**Two Character derivational Suffix:** ଉଳାପିତ୍ତିକାର, ଉଳାଧିକାର, ଉଳାରତବ୍ୟ, ଉଳାତମତ, ଉଳାନ୍ତାକ, ଉଳାକର୍ତ୍ତନ, ଉଳାନାକାଲ, ଉଳାକାତମ, ଉଳାକାନ୍ତତ୍ତ୍ତ୍ତ୍ତ୍ତ୍ତ୍ତ୍ତ୍ତ୍ତ୍ତ୍

**Three Character derivational Suffix:** ଉଳାପିତ୍ତିକାରା, ସନ୍ତୁ‌ରାକାର, ସନ୍ତୁ‌ରାକାଲ, ସନ୍ତୁ‌ରାକାତମ

**Output:** The output after processing is “ଉଳାପିତ୍ତି.”
VI. RESULT & DISCUSSION

We have evaluated the stemmer by taking different set of words i.e. 100 words, 200 words, 300 words and so on to calculate the time taken to extract the root words. We have not compared our Odia stemmer with any of the existing stemmer available for Odia language. Nowhere had we found the existing result to compare with the proposed stemmer.

Table 4: Time Taken to Extract Odia Root Words

| Set No | No of Words | Time Taken (Sec.) |
|-------|-------------|-------------------|
| Set-1 | 100         | 5.07              |
| Set-2 | 200         | 8.92              |
| Set-3 | 300         | 13.27             |
| Set-4 | 400         | 17.18             |
| Set-5 | 500         | 20.29             |
| Set-6 | 600         | 24.84             |
| Set-7 | 700         | 29.11             |
| Set-8 | 800         | 32.85             |
| Set-9 | 900         | 37.66             |
| Set-10| 1000        | 40.48             |

Fig. 3. Evaluation graph

VII. CONCLUSION AND FUTURE WORK

This stemmer can be used effectively to improve the query translation performance for information retrieval system [15]. Finally, we have integrated the new stemmer in the existing Dspace for Odia text retrieval system. The results are commendable and implicates that this can be used effectively in Odia Search Engine. It also handles the problem of under stemming and over stemming in some extent. In future we can integrate this stemmer for Odia text summarization purpose. As a future scope this system can be enhanced by including some more inflected word and corresponding to their stem mapping in to the database.

ACKNOWLEDGMENT

We are thankful to all the Language Researcher who directly and indirectly help to design and develop this web based Odia Stemmer.

REFERENCES

1. Lovins, J.B., "Development of a stemming algorithm", Mechanical Translation and Computational Linguistics, 1968, Vol. 11 Nos. 1 and 2, pp. 22-31.
2. Porter, M.F. (1980), "An algorithm for suffix stripping", 1980, Program, Vol. 14 No. 3, pp. 130-137.