A Rule Based Lightweight Bengali Stemmer

Souvick Das
University of Calcutta
Kolkata, India
souvik.cs@hotmail.com

Rajat Pandit
West Bengal State University
Kolkata, India
rajatpandit123@gmail.com

Sudip Kumar Naskar
Jadavpur University,
Kolkata, India
sudip.naskar@gmail.com

Abstract

In the field of Natural Language Processing (NLP) the process of stemming plays a significant role. Stemmer transforms an inflected word to its root form. Stemmer significantly increases the efficiency of Information Retrieval (IR) systems. It is a very basic yet fundamental text pre-processing task widely used in many NLP tasks. Several important works on stemming have been carried out by researchers in English and other major languages. In this paper, we study and review existing works on stemming in Bengali and other Indian languages. Finally we propose a rule based approach that explores Bengali morphology and leverages *WordNet* to achieve better accuracy. Our algorithm produced stemming accuracy of 98.86% for Nouns and 99.75% for Verbs.

1 Introduction

Information retrieval is a very essential process to extract relevant data or documents systematically from big data collections. Inverted index is a crucial data structure used in almost all modern IR systems. All the words in the entire data collection is stemmed first before the inverted index is built. Thus stemming plays a very important role in IR (Kowalski, 2007).
Stemming is the process of reducing inflectional or derived variant forms of one word to its root form. Two major components of IR task is basically indexing and retrieval. Stemming not only enhances the recall rate of IR task but also reduces the index size significantly. Thus, it increases the efficiency of the information retrieval system.

Depending on the linguistic rules of a particular language, words of any natural language can be inflected in many ways. Bengali is one of the most morphologically decorated languages. Generally inflected words are generated from the root word by adding some suffixes. It is observed that one root word in Bengali may have more than 20 morphological variants. Another challenge is in finding root word from a compound word i.e a word can be formed by conjunction of more than one root words. A large number of notable works have been done on stemming techniques for different languages from the last couple of decades. Most of the approaches were first applied in English language and subsequently adapted in other languages. Different stemming approaches involve different techniques such as longest suffix matching, dictionary based look-up, co-occurrence computation etc. Several works (Faili and Ravanbakhsh, 2010; Urmi et al., 2016; Makhija, 2016) have been done based on these techniques in different languages. However, a very few papers utilize these techniques in Bengali language but failed to achieve over 95% of accuracy.

In this paper, we propose a *rule based technique* that utilizes rich volume of Bengali Grammar rules also involves *Bengali WordNet* (Dash et al., 2017) to achieve better accuracy. It is worth mentioning that we also verify the extracted stem word from our *rule based algorithm* with the help of modified *WordNet*. The sole purpose of the modification of the *WordNet* here is to reduce the complexity of the approach by circumventing the unnecessary prefix suffix removal of the word. The verification process and *WordNet* modification thus help us to achieve next level of higher accuracy. The following example helps to understand the proper usage of the *WordNet* in this approach. Suppose a word সরাইখানা (*Inn*) is the root word and within this word one of the suffixes খানা is present. If we remove the suffix from the word, it will mean something else. Similar example can be seen for the word দুর্বার (*Strong*) where one of the suffix set র is present within the word. In Bengali stemming, filtering of root word and inflected word was a immense challenge. We overcome this problem by introducing the *WordNet* in the methodology. In the subsequent section, we present several cases where the modification of *WordNet* plays an important role.

The rest of the paper is structured as follows: Section 2 states the related works in this literature and brief review on them. Formation of different types of inflections in Bengali languages are analyzed in section 3. Section 4 elaborates different techniques involved in the proposed methodology and the entire methodology. Different resources that are used in the evaluation of the algorithm is described in section 5. Experiment and dataset used in the experiment is detailed out in section 6. Evaluation and results are presented in section 7. Finally we conclude this paper in section 8.

2 Related Work

There are two types of approaches in stemming mechanism. The first approach is called stemming in which affixes and suffixes are removed in order to extract the root word. The second approach is called *lemmatization*. Lemmatization requires a sound knowledge of the researcher in the particular language. As an example the word “Good” in English language has its variants “Better” and “Best”. In order to get the *lemma* (root word) “Good” we need lemmatization. Lemmatization involves dictionary look up to solve this kind of unusual case of extracting root words.

Stemming technique for English language is well studied and several techniques have been developed. The very first stemming technique Lovins stemmer was proposed by Julie Beth Lovins in 1968 (Frakes and Fox, 2003). Lovins stemmer removes a suffix just in two major steps. It maintains 294 endings, 29 conditions and 35 transformation rules. Later, well known rule based Porters stemming (Porter, 1980) technique was proposed in 1980 which is basically a rule based algorithm. In its 8 steps it handles different cases of morphological and inflexional endings in different steps. An extension of the Lovins algorithm is known as Dawson stemmer (Jivani et al., 2011) was proposed which covers wide range of list of 1200 suffixes.

These above mentioned algorithms work very well for English language but we are more interested in different stemming algorithms for Indian languages. Ramanathan and Rao (2003) proposed a lightweight stemmer for Hindi in 2004 which removes the suffixes based on the longest suffix matching from a list of suffixes. They also developed a suffix list in Hindi language to enhance the performance of the stemming. Akram et al. (2009) proposed an affix-exception list based stemmer for Urdu language. They omit prefix and suffix from the word based on looking up the exception list of prefixes or suffixes. This stemmer finds the stem word based on lexical look up method. A successful look up ignores the stripping off of the prefix and suffix of a word. Hussain et al. proposed a stemming mechanism for Urdu language based on n-gram stripping model (Durrani and Hussain, 2010). Kumar and Rana (2011) developed a brute force algorithm to strip off suffixes in order to find stem words in Punjabi language. They have overcome the problem of over-stemming and under-stemming. The suffix stripping is replaced sometimes by suffix substitution. Islam et al. (2007) proposed a lightweight stemmer which strips off suffixes and finds the stem word for Bengali. The fundamental idea of this algorithm is to remove suffixes based on the longest suffix matching. They also maintain a list of possible suffixes for Bengali language. Paik et al. (2011) reported a simple corpus based unsupervised stemmer for Bengali Language. The algorithm uses some statistics collected from corpus analysis based on the co-occurrences between word variants. They generate a graph where nodes are the variants of word and an edge between them represents a co-occurrence. Das and Mitra (2011) used the Porter stemming technique in Bengali language. Majumder et al. (2007) developed a stemming technique based on statistical clustering based approach to discover equivalence classes of root words using some set of distance

Table 1: Comparison of different stemming techniques in Indian languages

Year	Language	Method	Author(s)	Description	Accuracy
2004	Hindi	A lightweight stemmer	Ramanathan et al.	Strip off the words endings from a suffix list on a 'longest match' basis.	88%
2009	Urdu	Affix-exception list based stemmer	Qurat-Ul-Ain Akram et al.	Stems the Urdu words using lexical lookup method (Assas-band).	91.20%
2012	Urdu	Unsupervised approach to develop stemmer	Shahid Husain et. al.	n-gram stripping model	95.8%
2010	Punjabi	Brute Force Technique	Dinesh Kumar, Prince Rana	Suffix Stripping	80.73%
2007	Bengali	Yet Another Suffix Stripper	Majumder et al.	Statistical clustering based approach to discover equivalence classes of root words using some set of distance measures.	91.56%
2009	Bengali	A lightweight stemmer	Islam et. al.	Suffix Stripping	90.80%
2011	Bengali	A Fast Corpus-Based Stemmer	Jiaul H. Paik and Swapan Kumar Parui	A purely corpus based technique finds the equivalence classes of variant words in an unsupervised manner.	95%
2011	Bengali	Porter stemming technique	Suprabhat Das and Pabitra Mitra	Suffix Stripping	96.27%
2014	Bengali	Rule Based Bengali Stemmer	Redowan Mahmud. MD et. al.	Rule based suffix removal technique without using any list of suffixes.	88%
2016	Bengali	Bengali Lemmatizer(BenLem)	A. Chakrabarty and U. Garain	Reverse transformation based lemmatizer from surface words.	81.85%
2016	Bengali	A Neural Lemmatizer for Bengali	A. Chakrabarty et. al.	Neural network based lemmatizer using word2vec and CBOW	69.57%

measures. Mahmud et al. (2014) developed a rule based Bengali stemmer in 2014. This paper identify the occurrences of different inflections and their pattern. They developed rules to remove suffix from a inflected word without using any list of suffix list. Chakrabarty and Garain (2016) have designed a Bengali Lemmatizer (BenLem) which is able to handle both inflection and derivational morphology in Bengali language. In this approach, they used the FIRE Bengali News Corpus. It achieved 81.85% of accuracy in terms of resolving the inflected and derived words. Chakrabarty et al. (2016) proposed a neural network based lemmatizer which achieved 69.57% of accuracy. Thangarasu and Manavalan (2013) presented a literature review on stemming techniques for the Indian languages. Patel and Shah (2016) presented a literature review on unsupervised stemming techniques.

Table 1 shows different stemming techniques for Indian languages. In this comparison Das and Mitra (Das and Mitra, 2011) shows the highest accuracy of 96.27% for Bengali language with respect to other Indian languages and the lowest accuracy is 80.73% for Punjabi language proposed by Kumar and Rana (Kumar and Rana, 2011).

3 Inflections of words in Bengali language

Inflection of word is a process in which a word takes different forms. It may be based on tense, number and person. Different forms of the actual word are called inflected words. Bengali is one of the most morphologically decorated languages due to its wide range of inflected words. In this language inflections are mainly observed for verbs and nouns. Adjectives in Bengali can take only two suffixes - তর and তম, marking comparative and superlative adjectives, respectively, while adverbs in

Table 2: Possible Suffixes for Bengali Language

Tense	**1st Person & 2nd Person**	**2nd Person (Formally, Informally)**	**2nd Person (Informally for Junior Persons)**	**3rd Person (Formally)**	**3rd Person (Informally)**
Past Indefinite	লাম	লে, লেন	লি	লেন	লা, লো
Past Continuous	ছিলাম	ছিলে	ছিলি	ছিলেন	ছিল
Past Perfect	এছিলাম	এছিলে, এয়েছিলে, ইয়েছিলে এয়েছিলেন, এছিলেন, ইয়েছিলেন	ইয়েছিলি, এছিলি, এয়েছিলি	এছিলেন, এয়েছিলেন, ইয়েছিলেন	ইয়েছিল, এয়েছিল, এছিল
Present Indefinite	ই	এন	ইস	এন	এ
Present Continuous	চ্ছি, ছ	চ্ছ,ছ , চ্ছেন, এছেন	চ্ছিস, এছিস	চ্ছেন, এছেন	চ্ছে, এছে
Present Perfect	এছি	এছ, এছেন	এছিস	এছেন	এছে
Present Perfect Continuous	Not Applicable	এন	Not Applicable	উন	উক
Future Indefinite	ব (বো)	বে, বেন	বি	বেন	বে
Future Continuous	তেথাকব	তেথাকবেন	তেথাকবি	তেথাকবেন	তেথাকবে
Future Perfect	এথাকল	থাকবে	এথাকবি	এথাকবেন	এথাকবে
Future Perfect Continuous	Not Applicable	বেনওএন	তিস	বেন	বে
Habituatal Past	তাম	তে, তেন	তিস	তেন	ত

Bengali do not get suffixed. In this section we address the inflections of verbs and nouns. We also analyze the rules of inflections.

3.1 Inflections for Verbs

In Bengali language a verb is formed from the root-verb by joining some suffixes. As an example the root-verb of verb বলেছেন (*Told*) is বল and the suffix is এছেন. The inflections in verbs are varied according to tenses and the persons. The deviation of verb form according to the tense can be observed easily. For example the word বলছে (*Telling*), বলেছেন (*Told*) is deviation of actual root word বলা (*Tell*). Inflections are also noticed in case of informal and formal communications. For example the verb (Go) in the sentence "you (addressing younger one) are going" is presented as যাচ্ছিস and the verb (Go) in the sentence "you (addressing elder/ respected one) are going" is presented as যাচ্ছেন. Both of these words are infected form of যাওয়া (*Go*) and the root-verb is যা. One important point is essential to notice that the deviation of word যাওয়া (*Go*) is গিয়েছিলেন *(Went)* where there is no such linguistic interpretation. In such cases we maintain a mapping between root-verb and its possible deviations. We have listed out a number of possible suffixes in table 2 that are used to deviate a verb from its stem form. We apply rule based stemming mechanism to extract root-verb by omitting suffixes from the inflected word. A detailed procedure is illustrated in section 4. It is worth mentioning that the length of the root-verb in Bengali language is maximum 3. Root-verbs of different length is also presented in the table 3

3.2 Inflections for Noun

Noun inflections in Bengali language are limited. In case of verbal inflections the stem words can be changed sometimes but in case of Nominal inflection the stem word cannot be changed. Noun inflections are occurred to mention singular and plural forms of an object. Limited number of suffixes (Bhattacharya et al., 2005) such as `টি', `টা' ,`রা', `খানা', `খানি', `গুলো', `গুলি', `এরা', `রাজি', `রাশি', `পুঞ্জ', `সমূহ' etc. are added to the stem words. These suffixes are also added according to the representation of human being or other living things or non-living things. A number of noun inflections are presented in table 4. Example of different Noun inflections can be ছেলেটি (*The boy*), বইগুলি (*Books*), বৃক্ষাদি (*Trees*), পর্বতমালা (*Mountain range*) etc.

4 Proposed Methodology

We have discussed about various verb and noun inflections in Bengali Language till now. We also mentioned possible suffixes for deviation of a word and the different root-verb of different lengths of verbs. In this section, we illustrate our methodology to find stem of a verb.

It is easily observable that the number of rules for formation of different length of root-

Table 3: Different Types of Root-Verbs available in Bengali

Length	Category	Root-Verbs
1	Single Letter	হ, ক্ষ, ল, ঘ
1	Letter + আ	খা, ধা, পা ,যা, গা
1	Letter + ই	দি, নি
1	Letter +ু	শু, ধু, নু, etc
2	2 Letters	কর, কম, গড়, চল, পড়, জম etc around 100 root-verbs
2	letter + হ	কহ, সহ, বহ etc
2	্ is adddded at last	কাট্, গাঁথ্, চাল্, আঁক্, কাঁদ্, বাঁধ্, লিখ্, কিন্, জিত্, ঘির্, ফির্, ভির্, চিন্, উঠ্, শুন্, ফুট্, খুঁজ্, খুল্, উড়্ etc almost 200 root-verbs
2	হ্ is added at last	গাহ্, বাহ্, নাহ্, চাহ্ etc
2	া is added at last	চড়া, কাটা, লাফা, চরা, ছড়া, ছরা, আগা, চালা, নাহা, গাহা, ফিরা, ছিটা, শিখা, ঝিমা, পিটা, মিটা, লুকা, ঘুরা, কুড়া, উঁচা, পুঁড়া, ধুয়া, ধোয়া, শোয়া, খোয়া, খোঁচা, গোছা, পোঁছা, দৌড়া etc around 250 root-verbs
3	া is added at last	চটকা, সমঝা, কচলা, ধমকা, ছিটকা, হিঁচড়া, সিটকা, বিগড়া, দুমড়া, মুচড়া, উলটা, উপচা, ছোবলা, কোঁচকা, কোঁকড়া etc almost 150 root-verbs

Table 4: Different types of Noun Inflections

Objects	Singular	Plural
Human Beings	টা, টাকে, টি, টার, র, ের, কে etc	গুলো, গুলি, গুলোর, গুলির, রা, দের, দেরকে etc
Other Living or Non-Living Things	টা, টাকে, টি, টার, র, টাতে, টিতে, ের, ে etc	গুলো, গুলি, গুলোর, গুলির, এরা, জন, গুলিতে, গুলোতে, রাজি, রাশি, পুঞ্জ, সমূহ, বর্গ, বৃন্দ, বর্গ, মালা, দি etc

verbs are limited. We can generate root-verbs of different length from an inflected word by applying the mentioned rules. For example from the inflected word খেয়েছেন (*ate*) we can generate possible root-verbs of different length upto 3. If we consider length 3 we get খয়ছা, খুয়ছা, খিয়ছা etc about 6 words. Similarly for length 2 we can generate words like খয়, খহ,খায়, খিয়, খুয়্ etc about 8 words and finally for length 1 we have খ, খা, খি, খু . Out of these all possible root-verbs only one will be matched with valid one (খা) and corresponding verb(খাওয়া) (*eat*) will be retrieved.

At this point we would like to illustrate our proposed methodology for stemming. The algorithm is presented in Algorithm 1. This algorithm takes one word possibly an inflected word $\mathbb{I}_{\mathbb{W}}$ and access Bengali WordNet. Before doing any kind of suffix removal it checks the word in the WordNet to confirm that the word is inflected or not. In some cases suffixes are present in a word and creates a new word. Removing this suffix from the word changes the intended meaning of the word. For example আধার (*Dark*), if we remove 'র' from the word, it will be আধা (*Half or Half pieces*). On the other hand if we consider word গাধার (Donkey's) and if we remove `র' then it will be গাধা (*Donkey*) which is valid suffix removal. So before doing suffix removal we should search that particular word in the WordNet. In the next stage, algorithm perform the stemming mechanism according to its category (Noun or Verb). In this phase root form of the verb is generated. The function $\mathbb{G}()$ generates the root-verb according to the lengths ranging from 3 to 1. It generates all possible root-verbs based on the first 3 letters of the inflected word by applying rules mentioned in table 3. If it does not find any matching valid root-verb, it continues to generate all possible root-verbs of length 2 and so on. As the number of rules are constant and very few, it does not take too much amount of time. If a match found then the corresponding verb of the root-verb is returned. In the previous step we have already filtered the root words having some suffixes within those root words. In this stage we can emphasis that the words will have intentional suffixes merged with its

Algorithm 1 Stemming Algorithm

Input $\mathbb{I}_{\mathbb{W}}$ is the word that is the inflected form of a verb or noun along with its parts of speech tagged. We maintain a database for list of nouns and verbs. $\mathbb{N}_{\mathbb{D}}$ is the database containing nouns and $\mathbb{V}_{\mathbb{D}}$ is the database containing verbs. We maintain a list of root-verbs, root form of Nouns are in $\mathbb{R}_{\mathbb{ND}}$ database and $\mathbb{R}_{\mathbb{VD}}$ accordingly. $\mathbb{S}_{\mathbb{F}}$ is the set of suffixes used to inflect a Noun.

OutputAfter applying the following procedure actual root word will be assigned to $\mathbb{R}_{\mathbb{W}}$.

```
procedure STEMMER($\mathbb{I}_{\mathbb{W}}$)
    $\mathbb{W}_{\mathbb{POS}} \leftarrow \mathbb{P}(\mathbb{I}_{\mathbb{W}})$
    if $\mathbb{W}_{\mathbb{POS}} = NOUN$ then
        if $\mathbb{S}(\mathbb{I}_{\mathbb{W}}, \mathbb{N}_{\mathbb{D}}) = true$ then
            $\mathbb{R}_{\mathbb{W}} \leftarrow \mathbb{I}_{\mathbb{W}}$
            $return(\mathbb{R}_{\mathbb{W}})$
        else
            $i \leftarrow 0$
            while $i > 4$ do
                $\mathbb{RW}_{\mathbb{P}} \leftarrow \mathbb{R}(\mathbb{I}_{\mathbb{W}}, \mathbb{S}_{\mathbb{F}}[i])$
                if $\mathbb{S}(\mathbb{RW}_{\mathbb{P}}, \mathbb{R}_{\mathbb{ND}}) = true$ then
                    $\mathbb{R}_{\mathbb{W}} \leftarrow \mathbb{RW}_{\mathbb{P}}$
                    $return(\mathbb{R}_{\mathbb{W}})$
                end if
                $i \leftarrow i + 1$
            end while
        end if
    end if
    if $\mathbb{W}_{\mathbb{POS}} = VERB$ then
        if $\mathbb{S}(\mathbb{I}_{\mathbb{W}}, \mathbb{V}_{\mathbb{D}}) = true$ then
            $\mathbb{R}_{\mathbb{W}} \leftarrow \mathbb{I}_{\mathbb{W}}$
            $return(\mathbb{R}_{\mathbb{W}})$
        else
            $length \leftarrow 3$
            while $length > 0$ do
                $\mathbb{RV}_{\mathfrak{g}} \leftarrow \mathbb{G}(length, \mathbb{I}_{\mathbb{W}})$
                for each $\mathcal{R}_{\mathcal{G}}$ in $\mathbb{RV}_{\mathfrak{g}}$ do
                    if $\mathbb{S}(\mathcal{R}_{\mathcal{G}}, \mathbb{R}_{\mathbb{VD}}) = true$ then
                        $\mathbb{R}_{\mathbb{W}} \leftarrow \mathbb{G}_{\mathbb{RV}}(\mathbb{RV}_{\mathfrak{g}}, \mathbb{R}_{\mathbb{VD}})$
                        $return(\mathbb{R}_{\mathbb{W}})$
                    end if
                end for
                $length \leftarrow length - 1$
            end while
        end if
    end if
end procedure
```

root. So we generate all possible root-verbs of length 3 then of length 2 and finally length of 1. It is worth mentioning that according to the Bengali grammar, the length of the root-verbs never exceed its length by 3. The next stage is set up for the Nouns.

We define an array of possible suffixes for Noun words mentioned in table 5 . Based on utilization of suffixes, we define a suffix stripping rules. Let us consider an inflected word ছেলেগুলোদেরকে (*to the boys*). In this word multiple suffixes are applied. So in general we first search for suffixes like কে, তে and remove those suffixes if present in the inflected word. Here in this example the inflected word ছেলেগুলোদেরকে becomes ছেলেগুলোদের. Now after that we search ◌ে, র ,এরা, য়, রা and remove those suffixes if available. Now the word becomes ছেলেগুলোদে. Again we search for দে, কা, টা, টি and remove the one is present in the word. So the word is now ছেলেগুলো (*Boys*). After that the algorithm searches for জন, গুলো, গুলি, খানা and removes the appropriate one. In this stage we get ছেলে (*Boy*) which is the root word. Furthermore, the algorithm will search for রাজি, রাশি, বর্ণ, পুঞ্জ, বৃন্দ, বর্গ, সমূহ but at any stage if the word can be found in the WordNet we return that word as the root form of the inflected word.

Another major notable limitation of Bengali WordNet is that, WordNet does not contain deviated words a lot. It is certainly much difficult to identify all linguistic deviation of a word and store them all in WordNet. A word can be spoken or written in different way and that is deviated from one region to another region of a country. As an example, the deviated form of the word বলা (*Telling*) can be বলতে or বলতা. Inhabitants of various regions of West Bengal (A state of India) use the word 'বলতে' and on the other hand some inhabitants of some regions of West Bengal use the word 'বলতা'.

At this point we briefly illustrate the steps involved in the proposed algorithm. The algorithm takes inflected word ($\mathbb{I}_{\mathbb{W}}$) as input. In the next step function $\mathbb{P}()$ extracts the parts of speech ($\mathbb{W}_{\mathbb{POS}}$) of $\mathbb{I}_{\mathbb{W}}$. If it is a "NOUN" then a function $\mathbb{S}()$ searches the Noun database ($\mathbb{N}_{\mathbb{D}}$) for a match. If a match found, it means the inflected word has a meaning itself hence there is no need of stemming. Otherwise we can strip off different suffixes depending upon the presence of suffixes within the inflected word. We iterate through the suffix set $\mathbb{S}_{\mathbb{F}}$ and a function $\mathbb{R}()$ searches different possible suffixes within the inflected word and removes them. $\mathbb{R}()$ returns a word $\mathbb{RW}_{\mathbb{P}}$ in every iteration that does not contain i^{th} suffix set of $\mathbb{S}_{\mathbb{F}}$. $\mathbb{RW}_{\mathbb{P}}$ then searched in the "Root Noun" database $\mathbb{R}_{\mathbb{ND}}$ using function $\mathbb{S}()$. Whenever a match found the word is returned as the root form of the inflected word.

If $\mathbb{W}_{\mathbb{POS}}$ is "VERB" then again the search function $\mathbb{S}()$ searches the word in verb database ($\mathbb{V}_{\mathbb{D}}$) for a match. A match indicates

Table 5: Suffix Array

Index	1	2	3	4	5
Suffixes	কে, ত	◌ে, র ,এরা, য়, রা	দে, কা, টা, টি	জন, গুলো, গুলি, খানা	রাজি, রাশি, বর্গ, পুঞ্জ, বৃন্দ, সমূহ

that the word has its own meaning. So we return the word as it was. Otherwise a function $\mathbb{G}()$ will generate all possible root-verb from the inflected word and store them in set $\mathbb{RV}_{\mathfrak{z}}$. In first iteration it takes first 3 characters of the inflected word and make all possible root-verbs using the predefined rules. Then it takes 2 characters and so on. As in Bengali grammar root-verbs can have of length maximum of three, we start finding root-verbs in descending order. Every element within the set $\mathbb{RV}_{\mathfrak{z}}$ is checked in the root-verb database $\mathbb{R}_{\mathbb{VD}}$. At any point if a match found then the corresponding stem verb ($\mathbb{R}_{\mathbb{W}}$) is returned.

5 Resources Used

Bengali WordNet: Bengali WordNet is a part of IndoWordNet[1]. Bengali WordNet is a lexical database for Bengali words and it contains around 61 thousand Bengali words along with the Synsets. We imported the Bengali WordNet in MySQL database. We created a separate table for all the root-verbs corresponding to the Bengali verbs.

TDIL Corpus: For the corpus, we used the Technology Development for Indian Languages (TDIL) corpus (Jha, 2010) in this work.

6 Experiment

6.1 Dataset

We tested our algorithm on a testset of randomly chosen 500 sentences from the TDIL Bengali corpus of health and tourism domain articles. In this test dataset there are 2,756 unique content words containing 1304 Nouns, 1230 Verbs, 54 Adverbs and 168 Adjectives. Since in Indian languages, nouns and verbs get highly inflected, we concentrated on the stemming of nouns and verbs in Bengali. We used the *Stanford Bengali POS Tagger* to assign POS tag to each word of the testset. Finally we manually verified the POS tags and corrected the wrongly assigned tags.

[1]http://tdil-dc.in/indowordnet/

Table 6: System performance

Category	#Words	#Correctly Stemmed	Accuracy
Noun	1274	1234	96.86
Verb	1230	1227	99.75

6.2 Implementation

We implemented our algorithm in Python 3.6 and MySQL. In the very first step each sentence of the corpus is scanned by our python script. The Natural Language Tool Kit (nltk) package has been used to accomplish the necessary preprocessing tasks. Within the nltk package we have used Stanford Bengali POS Tagger to tag parts of speech for each word of the sentence. Then the sentence is tokenized into set of words. Another python script has been used to remove stop words listed by TDIL. At this point we use our actual python program to implement our proposed algorithm. It takes the tokenized words along with tagged parts of speech. We have used MySQL to store the Bengali WordNet. Whenever a searching in the WordNet is required, a particular module is responsible to search a particular word in the WordNet.

7 Result and Analysis

Our algorithm works for Verbs and Nouns. Out of 1230 verbs it successfully stems 1217 Verbs. In the other hand it successfully stems 1274 Nouns. Our algorithm fails for 43 words due to lack of words in Bengali WordNet. The accuracy of our algorithm shows 96.86% for the Verbs and 99.75% for Nouns. The accuracy of our algorithm is shown in table 6. The comparison of accuracy of different stemming techniques are presented in Figure-1. In figure-1 it is shown that our technique gives better accuracy than state-of-art stemming techniques in Bengali language.

In our approach we validate inflected words before applying our proposed rule based suffix removal technique. This validation technique

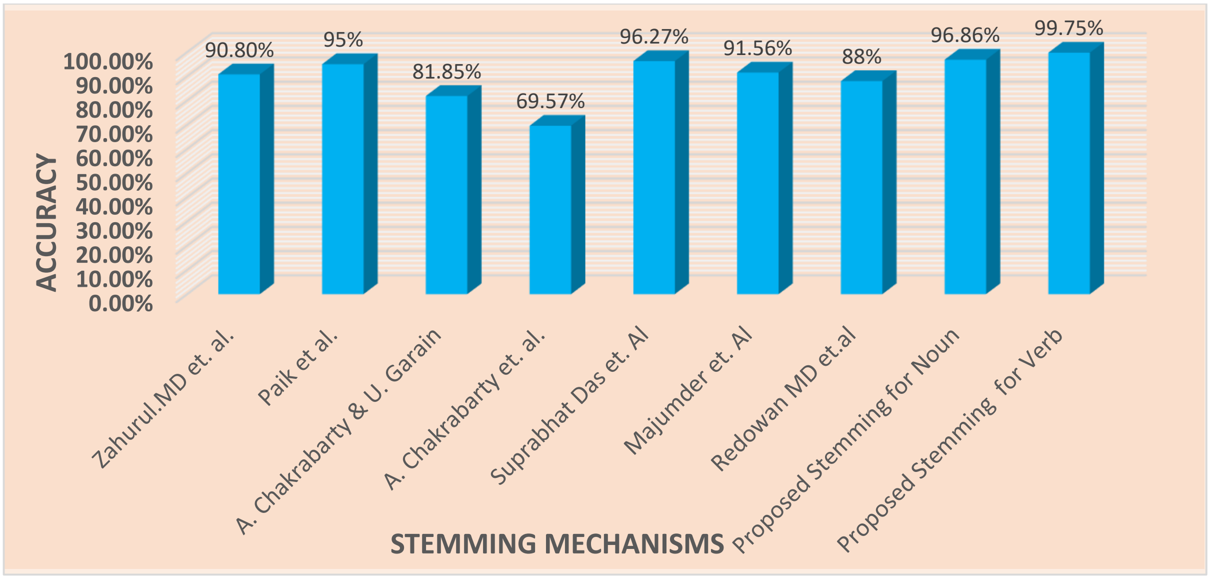

Figure1
Comparison of Accuracy of Different Stemming Techniques

is done by searching the word in the WordNet to check whether the word is inflected or not. We incorporate the rules of Bengali grammar to understand how verbs are formed from the root-verbs. Subsequently we observed that there are very limited rules for inflections of nouns. We extract the root-verbs from the inflected verbs by finding the combinations of suffixes and root-verbs of different lengths. Root words are also extracted from inflected nouns by applying step by step suffix removal.

The mechanism extracts actual root word from the inflected word and verifies it with the WordNet entry. There are some advantages and disadvantages in this approach. One of the major advantages is that the extracted root word will always be correct. This validation enhances the correctness of the extraction of root words.

One vital limitation of this entire mechanism is that, our algorithm rely on WordNet. There may be a situation where our algorithm extracts correct root word but it is not present in the WordNet and the extracted root word will be discarded.

8 Conclusions and Future Work

We have proposed a rule based algorithm for stemming verbs and nouns in Bengali. Incorporation of WordNet adds an extra degree in validation and extracting root words from inflected words. Bengali grammar rules have been used to find root-verbs of verbs efficiently. We have covered almost all kinds of root-verbs and possible suffixes to create a root word from an inflected verb.

References

Qurat-ul-Ain Akram, Asma Naseer, and Sarmad Hussain. 2009. Assas-band, an affix-exception-list based urdu stemmer. In *Proceedings of the 7th workshop on Asian language resources*, pages 40–46. Association for Computational Linguistics.

Samit Bhattacharya, Monojit Choudhury, Sudeshna Sarkar, and Anupam Basu. 2005. Inflectional morphology synthesis for bengali noun, pronoun and verb systems. In *Proc. of the National Conference on Computer Processing of Bangla (NCCPB 05)*, pages 34–43.

Abhisek Chakrabarty, Akshay Chaturvedi, and Utpal Garain. 2016. A neural lemmatizer for bengali. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2558–2561.

Abhisek Chakrabarty and Utpal Garain. 2016. Benlem (a bengali lemmatizer) and its role in wsd. volume 15, pages 1–18. ACM New York, NY, USA.

Suprabhat Das and Pabitra Mitra. 2011. A rule-based approach of stemming for inflectional and derivational words in bengali. In *Students' Technology Symposium (TechSym), 2011 IEEE*, pages 134–136. IEEE.

Niladri Sekhar Dash, Pushpak Bhattacharyya, and Jyoti D Pawar. 2017. The wordnet in indian languages. Springer.

Nadir Durrani and Sarmad Hussain. 2010. Urdu word segmentation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 528–536. Association for Computational Linguistics.

Heshaam Faili and Hadi Ravanbakhsh. 2010. Affix-augmented stem-based language model for persian. In *Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering (NLPKE-2010)*, pages 1–4. IEEE.

William B Frakes and Christopher J Fox. 2003. Strength and similarity of affix removal stemming algorithms. In *ACM SIGIR Forum*, volume 37, pages 26–30. ACM.

Md Islam, Md Uddin, Mumit Khan, et al. 2007. A light weight stemmer for bengali and its use in spelling checker. BRAC University.

Girish Nath Jha. 2010. The tdil program and the indian langauge corpora intitiative (ilci). In *LREC*.

Anjali Ganesh Jivani et al. 2011. A comparative study of stemming algorithms. volume 2, pages 1930–1938.

Gerald J Kowalski. 2007. *Information retrieval systems: theory and implementation*, volume 1. Springer.

Dinesh Kumar and Prince Rana. 2011. Stemming of punjabi words by using brute force technique. volume 3, pages 1351–1357.

Md Redowan Mahmud, Mahbuba Afrin, Md Abdur Razzaque, Ellis Miller, and Joel Iwashige. 2014. A rule based bengali stemmer. In *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, pages 2750–2756. IEEE.

Prasenjit Majumder, Mandar Mitra, Swapan K Parui, Gobinda Kole, Pabitra Mitra, and Kalyankumar Datta. 2007. Yass: Yet another suffix stripper. volume 25, page 18. ACM.

Sangita D Makhija. 2016. A study of different stemmer for sindhi language based on devanagari script. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2326–2329. IEEE.

Jiaul H Paik, Dipasree Pal, and Swapan K Parui. 2011. A novel corpus-based stemming algorithm using co-occurrence statistics. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 863–872. ACM.

Miral Patel and Apurva Shah. 2016. An unsupervised stemming: A review. volume 14, page 476. LJS Publishing.

Martin F Porter. 1980. An algorithm for suffix stripping. volume 14, pages 130–137. MCB UP Ltd.

Ananthakrishnan Ramanathan and Durgesh D Rao. 2003. A lightweight stemmer for hindi. In *the Proceedings of EACL*.

M Thangarasu and R Manavalan. 2013. A literature review: stemming algorithms for indian languages.

Tapashee Tabassum Urmi, Jasmine Jahan Jammy, and Sabir Ismail. 2016. A corpus based unsupervised bangla word stemming using n-gram language model. In *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 824–828. IEEE.