Design of a Rule-based Stemmer for Natural Language Text in Bengali

Sandipan Sarkar
IBM India
sandipan.sarkar@in.ibm.com,
sandipansarkar@gmail.com

Sivaji Bandyopadhyay
Computer Science and Engineering Department
Jadavpur University, Kolkata
sbandyopadhyay@cse.jdvu.ac.in

Abstract

This paper presents a rule-based approach for finding out the stems from text in Bengali, a resource-poor language. It starts by introducing the concept of orthographic syllable, the basic orthographic unit of Bengali. Then it discusses the morphological structure of the tokens for different parts of speech, formalizes the inflection rule constructs and formulates a quantitative ranking measure for potential candidate stems of a token. These concepts are applied in the design and implementation of an extensible architecture of a stemmer system for Bengali text. The accuracy of the system is calculated to be ~89% and above.

1 Introduction

While stemming systems and algorithms are being studied for European, Middle Eastern and Far Eastern languages for sometime, such studies in Indic scripts are quite limited. Ramanathan and Rao (2003) reported a lightweight rule-based stemmer in Hindi. Garain et. al. (2005) proposed a clustering-based approach to identify stem from Bengali image documents. Majumdar et. al. (2006) accepted the absence of rule-based stemmer in Bengali and proposed a statistical clustering-based approach to discover equivalence classes of root words from electronic texts in different languages including Bengali. We could not find any publication on Bengali stemmer following rule-based approach.

Our approach in this work is to identify and formalize rules in Bengali to build a stemming system with acceptable accuracy. This paper deals with design of such a system to stem Bengali words tokens tagged with their respective parts of speech (POS).

2 Orthographic Syllable

Unlike English or other Western-European languages, where the basic orthographic unit is a character, Bengali uses syllable. A syllable is typically a vowel core, which is preceded by zero or more consonants and followed by an optional diacritic mark.

However, the syllable we discuss here is orthographic and not phonological, which can be different. As for example, the phonological syllables of word কর্তা [*kartaa*] are কর্ [*kar_*] and তা [*taa*]. Whereas, the orthographic syllables will be ক [*ka*] and র্তা [*rtaa*] respectively. Since the term '*syllable*' is more used in phonological context, we use '*o-syllable*' to refer orthographic syllables, which will be a useful tool in this discussion.

Formally, using regular expression syntax, an o-syllable can be represented as $C*V?D?$ where C is a consonant, V is a vowel and D is a diacritic mark or *halant*. If one or more consonants are present, the vowel becomes a dependent vowel sign [*maatraa*].

We represent the o-syllables as a triple (C, V, D) where C is a string of consonant characters, V is a vowel character and D is a diacritic mark. All of these elements are optional and their absence will be denoted by Ø. V will be always represented in independent form.

We define *o-syllabic length* |τ| of token (τ) as the number of o-syllables in τ.

Few examples are provided below:

Token (τ)	O-syllable Form	|τ|
মা [*maa*]	(ম,আ,Ø)	1
চাঁদ [*chaa`nd*]	(চ,আ,ঁ) (দ,অ,Ø)	2
অগস্ত্য [*agastya*]	(Ø,অ,Ø)(গ,অ,Ø)(সতয,অ,Ø)	3

Token (τ)	O-syllable Form	|τ|
আটকা [*aaT_kaa*]	(Ø,আ,Ø) (ট,Ø,্) (ক,আ,Ø)	3

Table 1: O-syllable Form Examples

3 Morphological Impact of Inflections

Like English, the inflections in Bengali work as a suffix to the stem. It typically takes the following form:
<token> ::= <stem><inflections>
<inflections> ::= <inflection> |
<inflection><inflections>

Typically Bengali word token are formed with zero or single inflection. Example: মায়ের [*maayer*] < মা [*maa*] (stem) + য়ের [*yer*] (inflection)

However, examples are not rare where the token is formed by appending multiple inflections to the stem: করলেও [*karaleo*] < কর্ [*kar_*] (stem) + লে [*le*] (inflection) + ও [*o*] (inflection), ভাইদেরকেই [*bhaaiderakei*] < ভাই [*bhaai*] (stem) + দের [*der*] (inflection) + কে [*ke*] (inflection) + ই [*i*] (inflection).

3.1 Verb

Verb is the most complex POS in terms of inflected word formation. It involves most number of inflections and complex formation rules.

Like most other languages, verbs can be finite and non-finite in Bengali. While inflections for non-finite verbs are not dependent on tense or person; finite verbs are inflected based on person (first, second and third), tense (past, present and future), aspect (simple, perfect, habitual and progressive), honour (intimate, familiar and formal), style (traditional [*saadhu*], standard colloquial [*chalit*] etc.) mood (imperative etc.) and emphasis. Bengali verb stems can yield more than 100 different inflected tokens.

Some examples are: করাতিস [*karaatis*] < করা [*karaa*] (stem) + তিস [*tis*] (inflection representing second person, past tense, habitual aspect, intimate honour and colloquial style), খাইব [*khaaiba*] < খা [*khaa*] (stem) +ইব [*iba*] (inflection representing first person, future tense, simple aspect and traditional style) etc.

A verb token does not contain more than two inflections at a time. Second inflection represents either emphasis or negation.

Example: আসবই [*aasabai*] < আস্ [*aas_*] (stem) + ব [*ba*] (inflection representing first person, future tense, simple aspect and colloquial style) + ই [*i*] (inflection representing emphasis).

While appended, the inflections may affect the verb stem in four different ways:

1. Inflections can act as simple suffix and do not make any change in the verb stem. Examples: করা (stem) + চ্ছি [*chchhi*] (inflection) > করাচ্ছি [*karaachchhi*], খা (stem) + ব (inflection) > খাব [*khaaba*] etc.

2. Inflections can change the vowel of the first o-syllable of the stem. Example (the affected vowels are in **bold and underlined** style): শুধ্রা [*shudh_raa*] (stem) + স [*sa*] (inflection) > (শ,**উ**,Ø) (ধ,Ø,্) (র,আ,Ø) + স > (শ,**ও**,Ø) (ধ,Ø,্) (র,আ,Ø) + স > শোধ্রা [*shodh_raa*] + স > শোধ্রাস [*shodh_raasa*].

3. Inflections can change the vowel of the last o-syllable of the stem. Example: আট্কা [*aaT_kaa*] (stem) + ছি [*chhi*] (inflection) > (Ø,আ,Ø) (ট,Ø,্) (ক,**আ**,Ø) + ছি > (Ø,আ,Ø) (ট,Ø,্) (ক,**এ**,Ø) + ছি > আট্কে [*aaT_ke*] + ছি > আট্কেছি [*aaT_kechhi*].

4. Inflections can change the vowel of both first and last o-syllable of the stem. Example: ঠোক্রা [*Thok_raa*] (stem) + ও [*o*] (inflection) > (ঠ,**ও**,Ø) (ক,Ø,্) (র,**আ**,Ø) + ও > (ঠ,**উ**,Ø) (ক,Ø,্) (র,**ই**,Ø) + ও > ঠুক্রি [*Thuk_ri*] + ও > ঠুক্রিও [*Thuk_rio*].

3.2 Noun

Noun is simpler in terms of inflected token formation. Zero or more inflections are applied to noun stem to form the token. Nouns are inflected based on number (singular, plural), article and case [*kāraka*] (nominative, accusative, instrumental, dative, ablative, genitive, locative and vocative). Unlike verbs, stems are not affected when inflections are applied. The inflections applicable to noun is a different set than verb and the number of such inflections also less in count than that of verb.

Example: বাড়িটারই [*baarhiTaarai*] < বাড়ি [*baarhi*] (stem) + টা [*Taa*] (inflection representing article) + র [*ra*] (inflection representing genitive case) + ই [*i*] (inflection representing emphasis), মানুষগুলোকে [*maanushhaguloke*] < মানুষ [*maanushha*] (stem) + গুলো [*gulo*] (inflection representing plural number) + কে [*ke*] (inflection representing accusative case) etc.

3.3 Pronoun

Pronoun is almost similar to noun. However, there are some pronoun specific inflections, which are not applicable to noun. These inflections represent location, time, amount, similarity etc.

Example: সেথা [*sethaa*] < সে [*se*] (stem) + থা [*thaa*] (inflection representing location). This inflection is not applicable to nouns.

Moreover, unlike noun, a pronoun stem may have one or more post-inflection forms.

Example: stem আমি [*aami*] becomes আমা [*aamaa*] (আমাকে < আমা + কে) or মো [*mo*] (মোদের < মো + দের) once inflected.

3.4 Other Parts of Speeches

Other POSs in Bengali behave like noun in their inflected forms albeit the number of applicable inflections is much less comparing to that of noun.

Example: শ্রেষ্ঠতম [*shreshhThatama*] < শ্রেষ্ঠ [*shreshhTha*] (adjective stem) + তম [*tama*] (inflection representing superlative degree), মধ্যে [*madhye*] < মধ্য [*madhya*] (post-position stem) + ে [*e*] (inflection) etc.

4 Design

4.1 Context

As we identified in the previous section, the impact of inflections on stem are different for different POSs. Also the applicable list of inflections varies a lot among the POSs. Hence, if the system is POS aware, it will be able to generate more accurate result. This can be achieved by sending POS tagged text to the stemmer system, which will apply POS specific rules to discover stems. This proposition is quite viable as statistical POS taggers like TnT (Brants, 2000) are available.

The context of the proposed system is provided below:

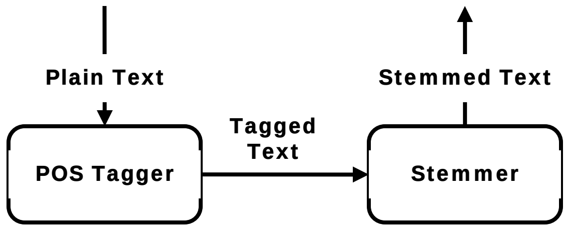

Figure 1: Context of Proposed Stemmer

4.2 Inflection Rule Observations

To discover the rules, we took the help of the seminal work by Chatterji (1939). For this work we limited our study within traditional and standard colloquial styles (dialects) of Bengali. For each of the POSs, we prepared the list of applicable inflections considering these dialects only. We studied these inflections and inflected tokens and framed the rules inspired by the work of Porter (1981). We had following observations:

1. To find out the stem, we need to replace the inflection with empty string in the word token. Hence all rules will take the following form:

<inflection> → ""

2. For rules related to verbs, the conditionals are present but they are dependent on the o-syllables instead of '*m*' measure, as defined and described in Porter (1981).

3. For pronouns the inflection may change the form of the stems. The change does not follow any rule. However, the number of such changes is small enough to handle on individual basis instead of formalizing it through rules.

4. A set of verb stems, which are called *incomplete verbs*, take a completely different form than the stem. Such verbs are very limited in number. Examples: যা [*Jaa*] (গেলাম [*gelaam*] etc. are valid tokens for this verb), আস্ (এলাম [*elaam*] etc. are valid tokens), আছ্ [*aachh_*] (থাকলাম [*thaakalaam*], ছিল [*chhila*] etc. are valid tokens)

5. For non-verb POSs, there is no conditional.

6. Multiple inflections can be applied to a token.

7. The inflections may suggest mutually contradictory results. As for example token খেলি [*kheli*] can be derived by applying two legitimate inflections লি [*li*] and ি [*i*] on two different stems খা [*khaa*] and খেল্ [*khel_*] respectively. Finding out the correct stem can be tricky.

8. Because of contradictory rules and morphological similarities in different stems there will be ambiguities.

4.3 Analysis and Design Decisions

Based on the observations above we further analyzed and crafted a few design decisions, which are documented below:

POS Group Specific Inflection Sets: It is observed that multiple POSs behave similarly while forming inflected word tokens. We decided to group them together and keep a set of inflections for each such group. By separating out inflection sets, we are minimizing the ambiguity.

We identified following inflection sets based on the tagset developed by IIIT Hyderabad for Indic languages. The tags not mentioned in the table below do not have any inflected forms. Size indicates the number of inflections found for that set.

Set	Comment	Size
I_N	The inflection set for noun group. It covers NN, NNP, NVB, NNC and NNPC tags.	40
I_P	The inflection set for pronoun group. It covers PRP and QW tags. This is a superset of I_N.	54
I_V	The inflection set for verb group. It covers VFM, VAUX, VJJ, VRB and VNN tags.	184
I_J	The inflection set for adjective group. It covers JJ, JVB, QF and QFNUM tags.	14
I_R	The inflection set for adverb, post-position, conjunction and noun-location POSs. It covers RB, RBVB, PREP, NLOC and CC tags.	6

Table 2: POS Groups

Pronoun – Post-inflection vs. Actual Stem Map: For pronoun we decided to keep a map of post-inflection stems and actual stems. After inflection stripping, this map will be consulted to discover the stem. Since number of pronouns in Bengali is limited in number, this approach will provide the most effective and performance friendly mechanism.

Verb – Morphological Rules: Based on observation 2, we further studied the verb POS and identified four classes of stems that exhibits own characteristics of morphological changes when inflections are applied. These classes can be identified for a stem σ based on the following two measures:

$$n = |\sigma| \text{ and } \lambda = \sum_{j=2}^{n} c_j$$

where c_j is the number of consonants in j-th o-syllable of the stem.

Class	Identification Characteristics
I	If $n = 1$. Example: খা [*khaa*], দে [*de*] etc.
II	If $n > 1$ and the n-th o-syllable has *halant* as diacritic mark. Only this class of verb stems can have *halant* at the last o-syllable. Example: কর্, শিখ্ [*shikh_*] etc.
III	If $n > 1$, $\lambda = 1$ and vowel of the n-th o-syllable is 'আ'. Example: করা, শিখা [*shikhaa*], দৌড়া [*dourhaa*] etc.
IV	If $n > 1$, $\lambda > 1$ and vowel of the n-th o-syllable is 'আ'. Example: আট্কা, ধম্কা [*dham_kaa*] etc.

Table 3: Verb Stem Classes

Since the verb inflections may affect the stems by changing the vowels of first and last o-syllable, a rule related to verb inflection is presented as a 5-tuple:

$$(L_1, R_1, L_n, R_n, i)$$

where

- L_1 is the vowel of the first o-syllable of post-inflection stem
- R_1 is the vowel of the first o-syllable of actual stem
- L_n is the vowel of the last (n-th) o-syllable of post-inflection stem
- R_n is the vowel of the last (n-th) o-syllable of actual stem
- i is the inflection

The vowels are always presented in their independent form instead of *maatraa*. This is because, we are going to apply these rules in the context of o-syllables, which can deterministically identify, which form a vowel should take. However, for inflection, we decided to differentiate between dependent and independent forms of vowel to minimize the ambiguity.

As for example, for the token ঠুক্রিও, inflection is ও, post-inflection stem is ঠুক্রি, and the actual stem is ঠোক্রা. Hence the rule for this class IV verb will be (উ, ও, ই, আ, ও).

Absence of an element of the 5-tuple rule is represented by 'Ø'. Example: for token খেয়ে [*kheye*], which is derived from stem খা, a class I verb stem; the rule will be (এ, আ, Ø, Ø, য়ে).

After completion of analysis, we captured 731 such rules. The distribution was 261, 103, 345 and 22 for class I, II, III & IV combined and IV respectively.

Map for Incomplete Verbs: For incomplete verbs, we decided to maintain a map. This data structure will relate the tokens to an imaginary token, which can be generated from the stem using a 5-tuple rule. Taking the example of token গেলাম, which is an inflected form of stem যা, will be mapped to যেলাম [*Jelaam*], which can be generated by applying rule (এ, আ, Ø, Ø, লাম). The system will consult this map for each input verb token. If it is found, it will imply that the token is an incomplete verb. The corresponding imaginary token will be retrieved to be processed by rules.

Recursive Stem Discovery Process: Since multiple inflections can be applied to a token, we decided to use a stack and a recursive process to discover the inflections and the possible stems for a token. However, we do special processing for verb tokens, which cannot have more than two inflections attached at a time and require extra morphological rule processing.

Ranking: Since there will be ambiguity, we decided to capture all candidate stems discovered and rank them. The client of the system will be expected to pick up the highest ranked stem.

Our observation was – stems discovered by stripping a lengthier inflection are more likely to be correct. We decided to include the o-syllabic length of the inflection as a contributing factor in rank calculation.

Additionally, for verb stems, the nature of the 5-tuple rule will play a role. There is a degree of strictness associated with these rules. The strictness is defined by the number of non-Ø elements in the 5-tuple. The stricter the rule, chances are more that the derived stem is accurate.

Taking an example – token খেয়ে [*kheye*] can be derived from two rules: খা [*khaa*] + য়ে [*ye*] is derived from (এ, আ, Ø, Ø, য়ে) and খায় [*khaay_*]+ ে [*e*] is derived from (Ø, Ø, Ø, Ø, ে). Since rule (এ, আ, Ø, Ø, য়ে) is stricter, খা should be the correct stem, and that matches with our knowledge also.

Let τ be a token and σ is one of the candidate stem derived from inflection ω.

For non-verb cases the rank of σ will be:

$$R_\sigma = |\omega|$$

For verb, the strictness of the rule that generated σ has to be considered. Let that rule be

$$\rho = (L_1, R_1, L_n, R_n, i)$$

The strictness can be measured as the number of non-Ø elements in the 5-tuple. Element *i* always demands an exact match. Moreover, (*L₁, R₁*) and (*Lₙ, Rₙ*) always come in pair. Hence the strictness S_ρ of rule ρ can be calculated as

$$S_\rho = \begin{cases} 1, \text{ if } L_1 = L_n = \phi \\ 2, \text{ if } L_1 \neq \phi \text{ and } L_n = \phi \\ 2, \text{ if } L_1 = \phi \text{ and } L_n \neq \phi \\ 3, \text{ if } L_1 \neq \phi \text{ and } L_n \neq \phi \end{cases}$$

Hence for verb stems the rank of σ will be:

$$R_\sigma = |\omega| + S_\rho$$

Overchanged Verb Stems and Compensation: Because of the rule strictness ranking some verb stems might be *overchanged*. As for example, token ভেজালাম [*bhejaalaam*] is an inflected form of stem ভেজা [*bhejaa*]. This is a class III stem. There are two relevant rules ρ_1 = (Ø, Ø, Ø, Ø, লাম) and ρ_2 = (এ, ই, Ø, Ø, লাম) which identifies the candidate stems ভেজা and ভিজা [*bhijaa*] respectively. Since the ρ_2 has higher strictness, ভিজা will rank better, which is wrong.

This type of situation only happens if the applied rule satisfies following condition:

(*L₁, R₁*) χ ((ই, এ), (এ, ই), (উ, ও), (ও, উ)).

This effect comes because the verbs with first vowel of these pairs at first o-syllable exhibits morphologically similar behaviour with such verbs for the last vowel of the pair once inflected.

শিখা and ভেজা are example of such behaviour. With inflection লাম, both of them produce similar morphological structure (শেখালাম [*shekhaalaam*] and ভেজালাম) even though their morphology is different at their actual stem.

To compensate that, we decided to include a stem to the result set without changing the first o-syllable, with same calculated rank, once such rule is encountered. Going back to example of ভেজালাম, even though we identified ভিজা as the stem with highest rank, since ρ_2 satisfies the above condition, ভেজা will be included with same rank as compensation.

Dictionary: To reduce ambiguity further, we decided to introduce a stem dictionary, which will be compared with potential stems. If a match found, the rank of that stem will be increased with a higher degree, so that they can take precedence.

Bengali word can have more than one correct spelling. As for example, জন্‌ম [*jan_ma*] and জন্ম [*janma*] are both correct. Similarly, গর্জা [*garjaa*] and গর্‌জা [*gar_jaa*], বর্‌ষা [*bar_shhaa*] and বর্ষা [*barshhaa*] etc.

To take care of the above problem, instead of exact match in the dictionary, we decided to introduce a quantitative match measure, so that some tolerance threshold can be adopted during the search in the dictionary.

Edit-distance measure (Levenshtein, 1966) was a natural choice for this. However direct usage of this algorithm may not be useful because of the following. For any edit operation the cost is always calculated 1 in edit-distance algorithm. This may mislead while calculating the edit-distance of a pair of Bengali tokens. As for example: The edit-distance for (বর্ষা, বর্‌ষা) and (বর্ষা, বর্শা [*barshaa*]) pairs are same, which is 1. However, intuitively we know that বর্‌ষা should be closer to বর্ষা than বর্শা.

To address the above problem we propose that the edit cost for diacritic marks, *halant* and dependent vowel marks should be less than that of consonants or independent vowels. Similarly, edit cost for diacritic marks and *halant* should be less than that of dependent vowel marks.

Formally, let VO, CO, VS and DC be the set of vowels, consonants, dependent vowel signs and diacritic marks (including *halant*) in Bengali alphabet.

We define the insertion cost C_i and deletion cost C_d of character γ as:

$$C_i(\gamma) = C_d(\gamma) = \begin{cases} 1, \text{ if } (\gamma \in CO) \text{ or } (\gamma \in VO) \\ 0.5, \text{ if } (\gamma \in VS) \\ 0.25, \text{ if } (\gamma \in DC) \\ 0, \text{ otherwise} \end{cases}$$

We also define the substitution cost C_s of character γ_1 by character γ_2 as:

$$C_s(\gamma_1, \gamma_2) = \begin{cases} 0, \text{ if } (\gamma_1 = \gamma_2) \\ Min(C_i(\gamma_1), C_i(\gamma_2)), \text{ otherwise} \end{cases}$$

We refer this modified distance measure as *weighted edit-distance* (WED) hereafter.

Going back to the previous example, the WED between বর্ষা and বর্শা is 1 and between বর্ষা and বর্‌ষা is 0.25. This result matches our expectation.

We proposed that the discovered stems will be compared against the dictionary items. If the WED is below the threshold value θ, we enhance the previous rank value of that stem.

Let D = (w_1, w_2, ... w_M) be the dictionary of size M. Let us define η_σ for stem σ as below:

$$\eta_\sigma = Min(\theta, \underset{k=1}{\overset{M}{Min}}(WED(\sigma, w_k)))$$

The modified rank of σ is:

$$R_\sigma = \begin{cases} |\omega| + S_p + \dfrac{100(\theta - \eta_\sigma)}{\theta}, \text{ if } \sigma \text{ is verb} \\ |\omega| + \dfrac{100(\theta - \eta_\sigma)}{\theta}, \text{ otherwise} \end{cases}$$

The match score is raised by a factor of 100 to emphasise the dictionary match and dampen the previous contributing ranking factors, which are typically in the range between 0 - 20.

5 System Architecture

The proposed system structure is provided below using Architecture Description Standard notation (Youngs et. al., 1999):

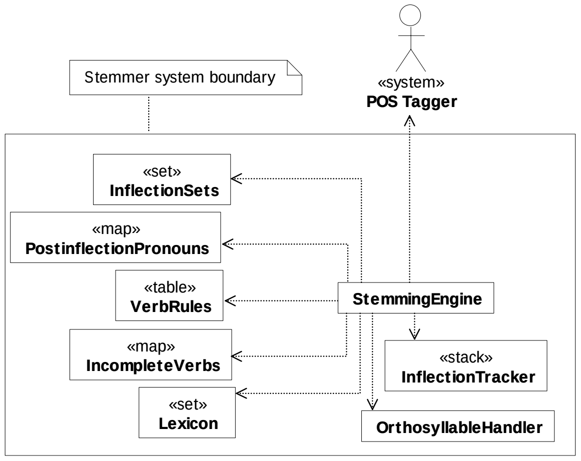

Figure 2: Stemmer Architecture

The components of the system are briefly described below:

StemmingEngine: It receives a tagged token and produces a set of candidate stems with their assigned ranks and associated inflection.

OrthosyllableHandler: This component is responsible for converting a token into o-syllables and vice-versa. It also allows calculating the WED between two Bengali tokens.

InflectionTracker: While discovering the inflections recursively, this stack will help the Stemming Engine to keep track of the inflections discovered till now.

InflectionSets: Contains the POS group specific inflection sets (I_N, I_P, I_V, I_J and I_R).

PostinflectionPronouns: A map of post-inflection pronoun stems against their corresponding actual stem form.

VerbRules: A table of 5-tuple verb rules along with their verb stem class association.

IncompleteVerbs: A map of incomplete verb tokens against their formal imaginary forms.

Lexicon: The dictionary where a discovered stem will be searched for rank enhancement.

As presented, the above design is heavily dependent on persisted rules, rather than hard-coded logic. This will bring in configurability and adaptability to the system for easily accommodating other dialects to be considered in future.

The high level algorithm to be used by the *StemmingEngine* is provided below:

```
global stems;

Stem(token, pos) {
    Search(token, pos);
    return stems;
}

Search(token, pos) {
    if (pos is verb and token χ IncompleteVerbs)
      token ← IncompleteVerbs[token];

    for (i = 1; i < token.length; i++) {
      candidate ← first i characters of token;
      inflection ← remaining characters of token;

      if (inflection ϖ InflectionSets)
        continue;

      if (pos is verb) {
        if (inflection is representing emphasis or negation) {
          InflectionTracker.push(inflection);
          Search(candidate, pos);
          InflectionTracker.pop(inflection);
        }

        class ← verb stem class of candidate;

        for each matching rule R in VerbRules for
        candidate and class {
            modify candidate by applying R;
            a ← inflection + inflections in InflectionTracker;
            r ← rank of the candidate based on |inflection|,
            strictness of R and match in Lexicon;
            Add candidate, a and r to stems;

            if (R is an overchanging rule)
              Modify candidate by compensation logic;
              Add candidate, a and r to stems;
        } // for each
      } // if pos is verb
      else {
        a ← inflection + inflections in InflectionTracker;

        if (pos is pronoun and
        candidate χ Postinflection Pronouns) {
          candidate ← PostinflectionPronouns[candidate];
        }

        r ← rank of the candidate based on |inflection|
        and match in Lexicon;
        Add candidate, a and r to stems;

        if (inflection != "") {
          InflectionTracker.push(inflection);
          Search(candidate, pos);
          InflectionTracker.pop(inflection);
        }
      } // else
    } // for
}
```

6 Evaluation

Based on the above mentioned approach and design, we developed a system using C#, XML and .NET Framework 2.0. We conducted the following experiment on it.

The goal of our experiment was to calculate the level of accuracy the proposed stemmer system can achieve. Since the system can suggest more than one stems, we sorted the suggested stems based on ranking in descending order and picked up the first (s'_i) and the next (s''_i) stems. We compared these stems against truthed data and calculated the accuracy measures A' and A'' as below:

Let $T = (t_1, t_2, \ldots t_N)$ be the set of tokens in a corpus of size N, $S = (\sigma_1, \sigma_2, \ldots \sigma_N)$ be the set of truthed stems for those tokens. Let s'_i and s''_i be the best and second-best stems suggested by the proposed stemmer system for token t_i. Then we define

$$A' = \frac{\sum_{i=1}^{N} f'(i)}{N}, \text{ where } f'(i) = \begin{cases} 1, \text{ if } \sigma_i = s'_i \\ 0, \text{ otherwise} \end{cases}$$

and

$$A'' = \frac{\sum_{i=1}^{N} f''(i)}{N}, \text{ where } f''(i) = \begin{cases} 1, \text{ if } \sigma_i \in (s'_i, s''_i) \\ 0, \text{ otherwise} \end{cases}$$

A' and A'' will be closer to 1 as the system accuracy increases.

Initially we ran it for three classic short stories by Rabindranath Tagore[1]. Since the proposed system accuracy will also depend upon the accuracy of the POS tagger and the dictionary coverage, to rule these factors out we manually identified the POS of the test corpus to emulate a 100% accurate POS tagger and used an empty dictionary. Apart from calculating the individual accuracies, we also calculated overall accuracy by considering the three stories as a single corpus:

[1] ইঁদুরের ভোজ [*i`ndurer bhoj*], দেনাপাওনা [*denaapaaonaa*], and রামকানাইয়ের নির্বুদ্ধিতা [*raamakaanaaiyer nirbuddhitaa*] respectively

Corpus	N	A'	A''
RT1	519	0.888	0.988
RT2	1865	0.904	0.987
RT3	1416	0.903	0.999
Overall	3800	0.902	0.992

Table 4: Accuracies for Short Stories by Tagore

As shown above, while A'' is very good, A' is also quite satisfactory. We could not compare this result with other similar Bengali stemmer systems due to unavailability. The closest stemmer system we found is the Hindi stemmer by Ramanathan et. al. (2003). It did not use a POS tagger and was run on a different corpus. The recorded accuracy of that stemmer was 0.815.

To check whether we can further improve on A', we introduced lexicon of 352 verb stems, ran it on the above three pieces with $\theta = 0.6$ to tolerate only the changes in *maatraa* and diacritic mark. We calculated A' for verbs tokens only with and without lexicon scenarios. We received the following result:

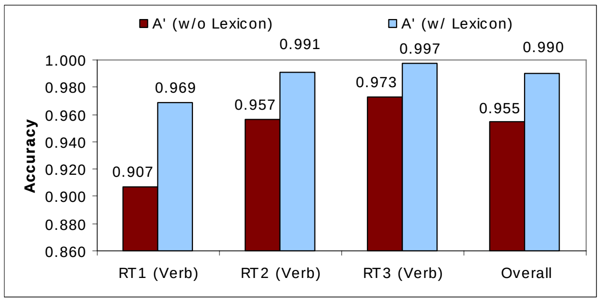

Figure 3: Comparison of Accuracies with and without Verb Lexicon

Above graph suggests that a lexicon can improve the accuracy significantly.

7 Conclusion

This paper proposed a system and algorithm for stripping inflection suffixes from Bengali word tokens based on a rule-based approach. The conducted experiments produced encouraging results.

Currently, our work is limited to the traditional and standard colloquial dialects of Bengali. Future works can be carried out to include other dialects by including more inflections in the respective data structure of this system.

The system suggests a set of ranked stems for a word token. The client of this system is expected to choose the highest ranked stem. This can be misleading for some of the cases where tokens derived from different stems share low or zero edit-distance among each other. As for example, when the verb token খেলি can be derived from both খা and খেল্, the system will suggest খা over খেল্.

This problem can be addressed by taking hints from word sense disambiguation (WSD) component as an input. Further studies can be devoted towards this idea. Moreover, a blend of rule-based and statistical approaches may be explored in future to improve the resultant accuracy of the stemmer.

While input from POS tagger helped to achieve a good performance of this system, it is yet to be studied how the system will perform without a POS tagger.

References

S. Chatterji. 1939. *Bhasha-prakash Bangla Vyakaran*. Rupa & Co. New Delhi, India

M. F. Porter. 1980. *An algorithm for suffix stripping*. Program 14(3):130-137.

U. Garain and A. K. Datta. 2005. *An Approach for Stemming in Symbolically Compressed Indian Language Imaged Documents*. Proceedings of the 2005 Eight International Conference on Document Analysis and Recognition (ICDAR'05). IEEE Computer Society

P. Majumder, M. Mitra, S. Parui, G. Kole, P. Mitra, and K. Datta. 2006. *YASS: Yet Another Suffix Stripper*. ACM Transactions on Information Systems.

T. Brants . 2000. *TnT: a statistical part-of-speech tagger*. Proceedings of the sixth conference on Applied natural language processing: 224-231. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA

V. I. Levenshtein. 1966. *Binary codes capable of correcting deletion, insertions and reversals*. Cybernetics and Control Theory, 10:707-710.

R. Youngs, D. Redmond-Pyle, P. Spaas, and E. Kahan. 1999. *A standard for architecture description*. IBM System Journal 38(1).

A. Ramanathan and D. D. Rao. 2003. *A lightweight stemmer for hindi*. In Proc. Workshop of Computational Linguistics for South Asian Languages - Expanding Synergies with Europe, EACL-2003: 42–48. Budapest, Hungary.