Rule based Automated Pronunciation Generator

Ayesha Binte Mosaddeque, Naushad UzZaman, and Mumit Khan

Center for Research on Bangla Language Processing, BRAC University, Dhaka, Bangladesh
lunaticbd@yahoo.com, naushad@bracu.ac.bd, mumit@bracu.ac.bd

Abstract

This paper presents a rule based pronunciation generator for Bangla words. It takes a word and finds the pronunciations for the graphemes of the word. A grapheme is a unit in writing that cannot be analyzed into smaller components. Resolving the pronunciation of a polyphone grapheme (i.e. a grapheme that generates more than one phoneme) is the major hurdle that the Automated Pronunciation Generator (APG) encounters. Bangla is partially phonetic in nature, thus we can define rules to handle most of the cases. Besides, up till now we lack a balanced corpus which could be used for a statistical pronunciation generator. As a result, for the time being a rule-based approach towards implementing the APG for Bangla turns out to be efficient.

1. Introduction

Based on the number of native speakers, Bangla (also known as Bengali) is the fourth most widely spoken language in the world [1]. It is the official language in Bangladesh and one of the official languages in the Indian states of West Bengal and Tripura. In recent years Bangla websites and portals are becoming more and more common. As a result it has turn out to be essentially important for us to develop Bangla from a computational perspective. Furthermore, Bangla has as its sister languages Hindi, Assamese and Oriya among others, as they have all descended from Indo-Aryan with Sanskrit as one of the temporal dialects. Therefore, a suitable implementation of APG in Bangla would also help advancement of the knowledge in these other languages.

The Bangla script is not completely phonetic since not every word is pronounced according to its spelling (e.g. কলম্বর /kolmi/). Therefore, we need to use some pre-defined rules to handle the general cases and some case specific rules to handle exceptions. For example, the words ‘নেক’ and ‘নটি’ both start with ‘ন’ but their pronunciations are [nık] and [nɔ] respectively. These changes with the pronunciation of ‘ন’ are supported by the phonetic rules:

\[
\text{অ + ক + ত (এ কার) > অ}
\]

\[
\text{অ + ই / C + র (ই কার) > ৷, where C= Consonant}
\]

2. Previous work

A paper about the Grapheme to Phoneme mapping for Hindi language [2] provided the concept that, an APG for Bangla that maps graphemes to phonemes can be rule-based. No such work has yet been made available in case of Bangla.

Although Bangla does have pronunciation dictionaries, these are not equipped with automated generators and more importantly they are not even digitized. However, the pronunciation dictionary by Bangla Academy provided us with a significant number of the phonetic rules [3]. And the phonetic encoding part of the open source transliteration software ‘pata’ [4] provided a basis.

3. Methodology

In the web version of the APG, queries are taken in Bangla text and it generates the phonetic form of the given word using IPA\(^1\) transcription. Furthermore, there is another version of the system which takes a corpus (a text file) as input and outputs another file containing the input words tagged with the corresponding pronunciations. This version can be used in a TTS\(^2\) system for Bangla.

In terms of generating the pronunciation of Bangla graphemes a number of problems were encountered. Consonants (except for ‘f’ and ‘s’) that have vocalic allographs (with the exception of ‘s’) are considerably easy to map. However there are a number of issues: Firstly, the real challenge for a Bangla

---

\(^1\) International Phonetic Alphabet

\(^2\) Text To Speech
pronunciation generator is to distinguish the different vowel pronunciations. Not all vowels, however, are polyphonic. ‘ɔ’ /ɔ/ and ‘œ’ /œ/ have polyphones (‘ɔ’ /ɔ/’ can be pronounced as [o] or [ə], ‘œ’ /œ/ can be pronounced as [e] or [ə], depending on the context) and dealing with their polyphonic behavior is the key problem. Secondly, the consonants that do not have any vocalic allograph have the same trouble as the pronunciation of the inherent vowel may vary. Thirdly, the two consonants ‘ŋ’ /ŋ/ and ‘ɔ’ /ɔ/ also show polyphonic behavior. And finally, the ‘consonantal allographs’ (‘ŋ’ /ŋ/, ‘θ’ /θ/, ‘f’ /f/, ‘v’ /v/) and the grapheme ‘ɔ’ /ɔ/ complicate the pronunciation system further.

Hypothetically, all the pronunciations are supposed to be resolved by the existing phonetic rules. But as a matter of fact they do not; some of them require heuristic assumptions. The phonetic rules that have been described in Table 1. Some of the rules are described below:

- If a word starts with ‘ɔ’ /ɔ/ followed by ‘f’ or ‘θ’ /θ/ then the ‘ɔ’ /ɔ/ is pronounced as [o]. For example, ‘क्रम’ /kram/’.
- If the first grapheme of a word is devoid of any vocalic allographs and a ‘‘ or ‘‘ is attached to it then the implicit ‘ɔ’ /ɔ/’ (associated to the first grapheme) is pronounced as [o]. For example, ‘क्रोम’ /krom/’.
- If the first grapheme of a word is devoid of any vocalic allographs and the consonant following it is ‘f’ /f/ or ‘v’ /v/ then the ‘ɔ’ /ɔ/ is pronounced as [o]. For example, ‘क्रोम’ /krom/’.
- If a word starts with ‘ŋ’ /ŋ/ followed by ‘f’ /f/ or ‘v’ /v/ or their corresponding vocalic allographs (‘f’ /f/’ then the ‘ɔ’ /ɔ/ is pronounced as [o]. For example, ‘क्रोम’ /krom/’.
- If a word starts with ‘ɔ’ /ɔ/ associated with a consonant followed by a ‘‘ /ŋ/’ or ‘‘ /ŋ/’ then if there is no ‘‘ /ŋ/’ ‘‘ /ŋ/’ ‘‘ /ŋ/’ or ‘‘ /ŋ/’ then the ‘ɔ’ /ɔ/’ is pronounced as [ə]. But if there is a ‘‘ /ŋ/’ ‘‘ /ŋ/’ ‘‘ /ŋ/’ then the ‘ɔ’ /ɔ/’ is pronounced as [e]. For example, ‘बेंζ’ /bɛnζ/’, ‘बेंζ’ /bɛnζ/’.

- If the consonantal allograph ‘ŋ’ /ŋ/’ is associated with the first grapheme of a word then the ‘ŋ’ /ŋ/’ is silent. For example, ‘स्वन’ /sɔn/’.
- If the consonantal allograph ‘ŋ’ /ŋ/’ is associated with a grapheme in the middle or at the end of a word then that grapheme’s pronunciation is doubled. For example, ‘बिश्व’ /bɪʃv/’.
- If the consonantal allograph ‘ŋ’ /ŋ/’ is associated with a grapheme in the middle or at the end of a word then that grapheme’s pronunciation is doubled. For example, ‘बिश्व’ /bɪʃv/’.

The consonantal allograph ‘ŋ’ /ŋ/’, when associated with the middle or end grapheme of a word, is not pronounced. For example, ‘शंध’ /ʃʌnd/’.
- If there is ‘ŋ’ /ŋ/’ at the end of the word and no vowel is associated with and the previous letter contains a ‘‘ /ŋ/’ or ‘‘ /ŋ/’, the ‘ŋ’ /ŋ/’ is pronounced as ‘‘ /ŋ/’. For example, ‘जातीय’ /jatijɔ/’.

Table 1 contains the phonetic rules. Apart from these some heuristic rules are used in APG. Few such rules are shown in Table 2. They were formulated while implementing the system. Most of them serve the purpose of generating pronunciation for some specific word pattern.

**Note:** C = Consonant. C C = Conjunct Consonant

<table>
<thead>
<tr>
<th>Letter/IPA/</th>
<th>Pronunciation</th>
<th>Rules</th>
<th>Example</th>
</tr>
</thead>
</table>
### Pronunciation

<table>
<thead>
<tr>
<th>Letter/IPA</th>
<th>Pronunciation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>a / o /</strong></th>
<th><strong>a / o /</strong></th>
<th><strong>a / o /</strong></th>
<th><strong>a / o /</strong></th>
</tr>
</thead>
</table>

#### Rules

- **Implicit** अ + ए = (ैैै) (concluded with a consonant)
- **Implicit** अ + ए (ैैैै) + ए (ैैैै)
- **Implicit** अ + ए (ैैैै) + ए (ैैैै)

### at the beginning

- **अ + ए / C+ै / (ै/ै/ै/ै)**
- **अ + ए / C+ै / (ै/ै/ै/ै)**
- **अ + ए (ै/ै/ै/ै)**
- **अ + ए (ै/ै/ै/ै)**
- **अ + C (ै/ै/ै/ै)**
- **अ + C+ (ै/ै/ै/ै)**

### at middle (implicit with consonants)

- **अ + ए / C+ै / (ै/ै/ै/ै)**
- **अ + ए / C+ै / (ै/ै/ै/ै)**
- **अ + ए (ै/ै/ै/ै)**
- **अ + ए (ै/ै/ै/ै)**
- **अ + ए (ै/ै/ै/ै)**
- **अ + ए (ै/ै/ै/ै)**

### at middle (implicit with consonants)

- **अ + ए / C+ै / (ै/ै/ै/ै)**
- **अ + ए / C+ै / (ै/ै/ै/ै)**
- **अ + ए (ै/ै/ै/ै)**
- **अ + ए (ै/ै/ै/ै)**
- **अ + ए (ै/ै/ै/ै)**
- **अ + ए (ै/ै/ै/ै)**

### at end

- **Words that end with ‘-ा, -ु, -ू’**
- **Words that end with ‘-वा, -व’**
- **Words that end with ‘-वा, -व’**
- **Words that end with ‘-वा, -व’**

### at the beginning

- **अ / a /**
- **अ / a /**
- **अ / a /**
- **अ / a /**
- **अ / a /**
| /e/ | একা > একা, দেখা > দেখার, খায়া, এহু, খায়া, মাথা-র, হলে, বাচ, কছে | । |
| /i/ | এই > এই, গুরু > গুরু, হায়া, হায়া | । |
| /u/ | ইং > ইং, নাই > নাই, যাতে | । |

Monosyllabic pronouns

| /æ/ | আ / অ | । |

Letter/IPA/ Rules

| /ŋ/ | ন / ও / Ong/ | । |
| /ŋ/ | ন / ও | । |
| /ɑ/ | আ / Ong | । |

| /n/ | ন / য | । |
| /n/ | ন / য | । |

Unpronounced

| /m as | ম / ম | । |

Example

| /j as | জ / জ | । |

Unpronounced

| /e/ | এ / এ | । |

Doubled

| /e/ | এ / এ | । |
At the beginning, if C does not have any vowel associated with it then it is pronounced as ‘/bn_o/’, and C is not doubled.

At middle or end, C is doubled.

At middle or end with conjuncts, C + /bn_ra/ is not doubled.

At the beginning, C is not doubled.

At middle or end, C is doubled.

At the beginning, C is not doubled.

At middle or end, C is doubled.

At the beginning, C is not doubled.

At middle or end, C is doubled.

At the beginning, C is not doubled.

At middle or end, C is doubled.

At the beginning, C is not doubled.

At middle or end, C is doubled.

At the beginning, C is not doubled.

At middle or end, C is doubled.

For foreign words,

Table 2: Heuristic Rules

<table>
<thead>
<tr>
<th>Letter/IPA/</th>
<th>Rules</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>झ / ʃ/</td>
<td>Always pronounced as झ / ʃ</td>
<td>पुर्व-शमन &gt; पुर्व-शमन-श</td>
</tr>
<tr>
<td>झ / j/</td>
<td>If there is ‘/bn_ja/’ before ‘/ʃ/’ and no vowel is associated with ‘/ʃ/’, then ‘/ʃ/’ is pronounced as ‘/ʃ/’.</td>
<td>सुनान &gt; सुनान-स</td>
</tr>
<tr>
<td>झ / j/</td>
<td>If ‘/ʃ/’ is followed by ‘/bn_yya/’ or ‘स/’, and no vowel is associated with ‘/ʃ/’, and then if there is any ‘ः’ or ‘ा’, then ‘/ʃ/’ is pronounced as ‘/ʃ/’, else it is pronounced as ‘/ʃ/’.</td>
<td>सवेत्स &gt; शवेत्स-श</td>
</tr>
<tr>
<td>झ / j/</td>
<td>If there is ‘/ʃ/’ at the end of the word and no vowel is associated with ‘/ʃ/’, and the previous letter contains a ‘/्या/’ or ‘/्या/’, the ‘/ʃ/’ is pronounced as ‘/ʃ/’.</td>
<td>सवेत्स &gt; शवेत्स-श</td>
</tr>
</tbody>
</table>
4. Implementation

APG has been implemented in Java (jdk1.5.0_03). The web version of APG contains a Java applet that can be used with any web client that supports applets. The other version of APG is also implemented in Java. Both versions generate the pronunciation on the fly; to be precise no look up file has been associated. Figure 1 illustrates the user interface of the web version and Figure 2 illustrates the output format of the other version.

Figure 1 : The web interface of APG. The input word is ‘জন’ and the corresponding output is ’ʧef’.

Figure 2 : The output file generated by the plug-in version of APG.

5. Result

The performance of the rule-based APG proposed by this paper is challenged by the partial phonetic nature of Bangla script. The accuracy rate of the proposed APG for Bangla was evaluated on two different corpora that were collected from a Bangla newspaper. The accuracy rates observed are shown in Table 3.

Table 3: Evaluation of accuracy

<table>
<thead>
<tr>
<th>Number of words</th>
<th>Accuracy Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>736</td>
<td>97.01</td>
</tr>
<tr>
<td>8399</td>
<td>81.95</td>
</tr>
</tbody>
</table>

The reason of the high accuracy rate of the 736-word corpus is that, the patterns of the words of this corpus were used for generating the heuristic rules. The words in the other corpus were chosen randomly. The error analysis was done manually by matching the output with the Bangla Academy pronunciation dictionary.

6. Conclusion

The proposed APG for Bangla has been designed to generate the pronunciation of a given Bangla word in a rule based approach. The actual challenge in implementing the APG was to deal with the polyphone graphemes. Due to the lack of a balanced corpus, we had to select the rule-based approach for developing the APG. However, a possible future upgrade is implementing a hybrid approach comprising both a rule based and a statistical grapheme-to-phoneme converter. Also, including a look up file will increase the efficiency of the current version of APG immensely. This will allow the system to access a database for look up. That way, any given word will first be looked for in the database (where the correct pronunciation will be stored), if the word is there then the corresponding pronunciation goes to the output, or else, the pronunciation is deduced using the rules.

7. Acknowledgement

This work has been supported in part by the PAN Localization Project (www.panl10n.net) grant from the International Development Research Center, Ottawa, Canada, administrated through Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan.

8. References


