PRONUNCIATION RULES FOR INDIAN ENGLISH TEXT-TO-SPEECH SYSTEM

Aniruddha Sen
School of Technology and Computer Science
Tata Institute of Fundamental Research
Homi Bhabha Road
Mumbai 400 005 INDIA
asen@tifr.res.in

ABSTRACT

Text-to-speech synthesis in Indian English is useful for delivering messages stored in computers and web to the Indian users unfamiliar with standard English accent. Such work is going on at TIFR and the paper reports the salient features of the front-end language processor that generates pronunciation plus stress information. The important components of the language processor are the parser to categorize words, an Indian English phonetic dictionary, morphological analyzer, letter-to-sound rules, phonological rules, prosody rules and Indian name detector. The relevant rules are formulated with the aid of a large CMU pronunciation dictionary and a language tool GENEX, developed in-house, that can generate a sub-dictionary following a set of specified constraints. The paper outlines the rule formulation procedure and provides examples of various types of rules. A few important morphological rules and letter-to-sound rules are described in detail.

1. INTRODUCTION

Text-to-speech synthesis is a useful tool to communicate, by voice, the information and text stored in computer and web. 'Indian English' is for the benefit of the Indian audience who can understand English speech in various degrees, but are not comfortable with standard English pronunciation and accents. Also intended is to generate better pronunciation for Indian names included in English text.

Tata Institute of Fundamental Research (TIFR) is conducting R & D on Indian English Text-to-Speech (TTS) systems. It is tempting to think that the task can be achieved by reverse engineering of the commercially available standard English Text-to-speech systems. But in reality, that is not viable due to several technological and commercial constraints. For example, any desired sub-module cannot be just obtained and plugged in. We, therefore, have undertaken development of the complete system, while putting the existing knowledge on technology to good use.

The Indian English TTS [1] is basically a combination of: (a) a front-end text processor that accepts text as input and generates phoneme string and accent markers as output and (b) a phoneme-to-speech synthesizer that is based on formant synthesis, implemented in two stages. The first stage creates a time-varying acoustic-phonetic parameter stream, given the input phoneme string. This is done by rules that covers all typical Indian speech sounds in various phonetic contexts. The rules were formulated through considerable research and analysis of Indian speech [2]. Next, a Klatt-type source filter model [3] is utilized to generate speech, given the parameter stream [Fig. 1].

Whereas the phoneme-to-speech module can be adapted to most Indian languages or to Indian English with minimal changes, the front-end text processor is strictly language specific. In our Indian English TTP, we therefore use the TIFR phoneme-to-text synthesizer with only a few alterations and develop a separate Indian English text processor.

Yes

Text String

English?

No
In this paper, we outline the text processor implementation in section 2. In section 3, we describe our method of rule formulation with the help of a standard pronunciation dictionary and a language tool developed by us. In Section 4, we present morphological analysis and letter-to-sound conversion: the issues on which we worked extensively. We outline our morphological analysis scheme with prioritized alternate root hypothesis. Then, we present a few important rule sets for morphological analysis and letter-to-sound conversion in detail. We also discuss specific problems for Indian names. In Section 5, we outline phonological rules and prosody rules, on which we are currently working. The implementation status and future course are discussed in Section 6.

2. TEXT PROCESSOR OVERVIEW

The input text is first parsed and broken down into various categories. Date, time, acronyms, number strings, alphanumerics, abbreviations etc. are handled routinely. The core problem is to generate pronunciation for the normal text words.

This is done basically in the following manner. The text words are looked up in a phonetic dictionary. As a dictionary is unlikely to contain all the root and derivative words, morphological analysis (separating prefixes, roots and suffixes) is done before that. If the dictionary look-up succeeds, the pronunciation is obtained from dictionary. If it doesn't, we fall back to letter-to-sound rules for generating pronunciation of the unknown words.

After thus obtaining the pronunciation of each morph, phonological rules are applied to suitably modify the pronunciation of each unit in the context of the adjacent ones. (An example of phonological rule is 'Sandhi' in Indian languages.) In addition, for natural accent, prosody rules are applied to modulate pitch, intensity and (intrinsic) phoneme durations. In the language processor, prosody markers are generated. The actual changes of acoustic-phonetic parameters are done accordingly in the phoneme-to-speech module.
3. LANGUAGE TOOL AND DICTIONARY

For relevant research, we used the Carnegie Mellon Pronouncing Dictionary [cmudict.0.6] as a tool. The dictionary is of approximately 100,000 words. It contains root words, commonly used derivative words, prefixes, suffixes and also some names. The dictionary is available for free download (http://www.speech.cs.cmu.edu/cgi-bin/cmudict, ftp://ftp.cs.cmu.edu/project/speech/dict/). Its use for research or commercial purpose is completely unrestricted.

The dictionary consists of words and their pronunciation. For our usage, we have made a few modifications. The pronunciation was changed to the corresponding and nearest 'Indian' pronunciation, represented by the phoneme set of the TIFR synthesizer that basically is the Hindi phoneme set plus a few additional English phonemes. The conversion was done by simple, mostly one-to-one rules. For simplicity, we removed the multiple pronunciation of the same words and retained only the most common ones.

This modified dictionary was however not suitable for being used as our phonetic dictionary. The one-to-one rules were good enough in most occasions, but were often inadequate, particularly in the context of neutral vowels. We therefore have prepared our own (Indian English) phonetic dictionary that is much smaller as of now. But the large amount of phonetic information stored in the CMU dictionary was utilized to formulate rules for morphological analysis and letter-to-sound conversions.

To search this huge dictionary manually and collate information is not quite feasible. We therefore developed a language tool GENEX (Generate Examples).

In its present version, GENEX generates instances present in a dictionary file that satisfy the specified constraints. The name of the input file, output file and the constraints are all to be specified in a definition file. The input file can be any file that is in the same format as that of the modified CMU dictionary, i.e. each line having a spelled word, followed by the pronunciation specified by a string of TIFR synthesizer phoneme symbols. The output file is a sub-dictionary that contains the entries, satisfying the constraints. It is possible to run GENEX next on that: to generate a sub-sub dictionary.

As constraints, it is possible to specify a character string in the spelling field, to be searched at either word initial, word final or in intermediate positions. A field may optionally be specified as 'vowel' or 'consonant'. It is also possible to specify a character string in the pronunciation field. But it should be noted that the search cannot be intelligent enough to correlate the character strings in the word and pronunciation fields. For example, if we specify 'g' in spelling and 'j' in pronunciation field together, we cannot hope to list only the words that have this particular 'g' pronounced as 'j'. If any 'j' is present in the pronunciation, the entry will be listed. Manual screening is therefore needed. This, however, reduces the search space: often to 1/2 or even 1/3.

A few other special facilities were also incorporated in GENEX that is discussed in subsequent sections along with their applications. The tool is under constant improvement as per the demands of research.

4. FORMULATION OF PRONUNCIATION RULES

4.1. General Strategy

Here, we mainly describe the formulation of morphological and letter-to-sound rules that have been researched to some depth. We will briefly touch upon the phonological and prosody rules, on which work has been initiated, in next section.
At the background, we acknowledge that work on language processing in Indian languages is not as advanced as that in English. It is not very easy to implement things such as phrase-level parser in Indian context. We therefore take what is called an ‘engineering approach’, i.e. doing things in more practical ways by cutting a few corners.

We assume that as the CMU dictionary has a good number of words, it is quite representative and we can take it as near-exhaustive. With GENEX, we list instances of words in particular contexts automatically and then analyze or sub-categorize them manually. If needed, we apply more constraints to the list file and generate a sub-file by running GENEX again. We then try to categorize rules and exceptions. As our Indian English phonetic dictionary is small, we try to ensure that we at least put in that the words with exceptional pronunciation.

Forming rules, particularly the letter-to-phoneme ones automatically is a real possibility that has been worked upon [4]. However, (i) the issue is still at research stage, with the accuracy around 70-80% and (ii) the approach needs an authentic pronunciation dictionary. Whereas that is available for standard English, we don't have a reliable one for Indian English as yet. So, we make use of the base pronunciation of the CMU dictionary and do rule formulation for Indian English manually. This will be the order of things for some time to come.

4.2. Morphological Analysis

For morphological analysis, a standard methodology, used for MITalk, has been described in [5]. The basic problem of morphological analysis is resolving ambiguities. (e.g. cared = car + ed, or care + ed?). MITalk applies an elaborate scoring system and selects the hypothesis with the highest score.

In comparison, we lack a few necessities such as a good, categorized morph dictionary or a good language parser. In our ‘engineering approach’, we generate all alternate hypotheses that are possible in a given context. We prioritize them by observed statistics and then try to search for the root word(s), corresponding to the most probable hypothesis, in the dictionary. If the search fails, we try the next one and so on.

Here, we assume that a more probable hypothesis fails only when the corresponding root word is not a dictionary word. For example, the most probable analysis (elaborated latter) of abandoned = abandone + ed. This analysis is of course wrong, the correct one being: abandoned = abandon + ed. Here, however, the word 'abandone' doesn't exist. So, if our first rule fails, we try the next best, and that now gives the correct result.

Our method, in practice, works reasonably well, except for creating some minor errors in some contexts that may lead to improper accent.

Usually, prefixes don't modify the root and are easily separable. Root modification by suffixes, particularly the ones starting with vowel, is the main reason for ambiguities and is discussed in some length hereafter.

We have formulated separate rules for each common suffix. Great care was taken in rule formation for the suffixes that start with vowels. The method adapted is to try for a look-up of the whole word first. If that fails, we mechanically search for suffix at the end and tentatively split the word into suffix and root. Then we assume a root modification that is most likely in the given letter context and try for a look-up again. If that fails, we try the next most probable modification and so on. If all such attempts fail, we pick up the first attempted (most probable) root modification. Letter-to-sound conversion is done for the root thus assumed. This was found to have given better results than conversion of the whole word.

However, for the method to work, if there is an exception to the most probable analysis, the corresponding word must be present in dictionary. In the previously mentioned example, ‘abandon' must be there in the dictionary, or else after the abortive first round, the first possibility (abandone) will be passed for letter-to-sound analysis. With the help of GENEX, we can generate the list of all such contexts and can manually examine the exceptions so that all such words are put into
dictionary. Another problem is that, say, there are two words 'care' and 'car'. If we encounter, say 'cared', we first try cared = care + ed. Now, if the word 'care' is not in the dictionary, we fall back to cared = car + ed, that is again wrong. To avoid such situations, there is a special command that can list all minimal pairs in some given context. For example, we can list all occurrences where a word, as well as the word + 'e' are present (e.g. 'car' and 'care'). We can check our rules and exceptions for such pairs and can append necessary entries into the phonetic dictionary.

As example, we here list a set of important rules: for three very common suffixes ing, ed and es.

Mainly the important rules are listed here. The exceptions in each case are found by GENEX and put appropriately in our dictionary. We specify the sequence of modifications attempted to the root in a given context by the respective type nos., as follows:

Type 1: Add 'e', e.g. cared = car + ed
Type 2: 1Æ y, e.g. carried = carry + ed
Type 3: Delete one of the repeated consonants, e.g. puting = put +ing
Type 4: Retain root without modification.
Type 5: False suffix estimation, treat the whole word as root.

(a) If the 'root' doesn't have any vowel or 'y': 5

(b) If the root ends with a vowel, the types nos. of the sequences to be tried are as per Table 1. Superscripts indicate examples, as listed below.

<table>
<thead>
<tr>
<th>Vowel</th>
<th>a</th>
<th>e</th>
<th>i</th>
<th>o</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>ing</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>ed</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>es</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1 Sequences for ing, ed, es

Examples: a: going = go + ing
b: gluing = glue + ing
c: agreed = agree + ed

d) If the root ends with two consonants:
For 'ff', 'll' and 'ss': 4 (calling = call + ing), 3.
For any other repeated consonants CC: 3 (putting = put +ing) and 4, only if CC is a valid cluster at end of word.

(e) If the root ends with consonant, preceded by vowel:

Situation is complicated here. We by and large go by a table that states basically if 1, 4 or 4, 1 sequence is to be tried.

4.3. Letter-to-Sound Rules

These rules can be broadly classified for consonants and vowels. The basic scheme here is that first appropriate rules are applied and the character string is suitably modified. Then, the letter to phoneme conversion is done through a table look-up. In the table, pronunciation of letter clusters (currently, up to size 4) are stored. The larger matches are tried before the smaller ones.

As for consonants, the pronunciation of many (e.g. b, d, m, p) are unambiguous. Ambiguity is associated mostly with c, g, s, x and y. The corresponding rules used are briefly discussed here. The rules, however, generate just the most probable pronunciation, determined with the aid of GENEX or otherwise.

(a) 'x': This is 'ks'. But word initial 'x' is 'z'.
(b) 's': This is pronounced as 's' or 'z', depending on context. 'ss' and word-initial 's' is always 'z'. Else, in standard English, it becomes 'z', unless any adjacent phoneme is unvoiced. (e.g.: easy → ezay, bags → bazi, but looks → looks). In Indian English, however, conversion of 's' to 'z' is much less frequent. We are trying to gather statistics from people with different background and native tongue to estimate the best solution.
(c) 'c.': This can be 'ch' (as in 'Chalk'), 'k' or even 's', depending on context. Main rules are:

- 'cc' is 'ks' (e.g. access → akssess).
- 'ch' is kept unaltered by default. Many instances of 'ch' → 'k' are there and are included in dictionary as far as possible.
- 'c' in other consonant cluster is 'k' (e.g. acting → akting).
- 'c' followed by 'd', 'o' and 'u' is 'k'. Exceptions are rare.
- 'c' followed by 'e', 'o' or 'y' is 'ch'. Exceptions are more and are mostly included in dictionary.

(d) 'g': This can be 'g' or 'j'. Main rules are:

- 'gh' is voiced-aspirated velar 'gh' of Hindi.
- 'g' in any other consonant cluster is 'g'.
- 'g' followed by 'd', 'o' and 'u' is 'g'. Exceptions are rare.
- 'g' followed by 'e', 'o' or 'y' is 'j'. Exceptions are more and are mostly included in dictionary.

(e) 'y': 'y' rules are elaborate. We are listing the main ones.

Word-initial 'y' is the glide 'y' (as in 'your'). So is it in the following context: (preceded by single vowel and followed by single vowel) or (preceded by double vowel) or (followed by double vowel).

Else, if preceded by a vowel:

- 'ay' → 'ey', as in 'ate'
- 'ey' → 'i', as in 'eat'
- 'oy' → 'oi', as in 'toy'
- 'uy' → 'ai', as in 'hide'
- 'iy' → 'i' + 'y' glide, occurs rarely.

Else, if preceded by a consonant: This rule is basically for word-final 'y', preceded by a consonant. It is discussed elaborately now:

Default pronunciation is 'i' (as in 'sympathy').

Usual alternative is 'ai' (as in 'try').

(i) If there is no vowel in the word: 'y' → 'ai' (e.g. 'spry', 'shy').
(ii) Preceding consonant other than 'f', 't' and 'b': 'y' → 'i' (e.g. 'amity'). Exception: 'Occupied'.

(iii) For 'yj':

1. Preceded by a consonant:
   a. Other than 's': 'y' → 'i' (e.g. 'comfy').
   b. 's': 'y' → 'ai' (e.g. 'satisfy').

2. Preceded by single vowel: 'y' → 'ai' (e.g. 'beauty', 'deify').

3. Preceded by double vowel, of that:
   a. Last one is 'i': 'y' → 'ai' (e.g. 'deify').
   b. First one is 'i', preceded by 'q': 'ai' (e.g. 'liquify').
   c. Else: 'y' → 'i' (e.g. 'leafy').
   (iv) For 'hy':

1. Preceded by 'f' (i.e. 'fly'):
   If the word is a derivative of 'fly' (e.g. butterfly, overfly, stored as exceptions), 'y' → 'ai', else 'y' → 'i'.

2. Preceded by 'pp': 'y' → 'ai' (e.g. 'supply').

3. Else, 'y' → 'i'. (e.g. commonly duly). Important exceptions are: July, bely, rely, comply, imply, multiply, reply.

(v) For 'by':

'y' → 'ai' for derivatives of 'by' (e.g. flyby, goodby, hereby, nearby, passerby, standby, thereby, whereby), else, 'y' → 'i'.

We considered a fact regarding Indian English while formulating vowel rules. A native Indian speaker/ listener usually tries to map the English vowels into the nearest ones in the speaker's native tongue. We therefore mapped English vowel phonemes into Hindi vowel phonemes, plus a few more English phonemes that were unavoidable. This gives rise to a problem. In English, there are at least 15 vowels, whereas in Hindi the number is 10. So, there are approximations.

An English vowel is usually scripted by either a single vowel letter (a/e/i/o/u), or that followed by another vowel letter/w/y. We have kept table entries for pronunciation, corresponding to all such single/ double letter combinations.

A well-known and important vowel rule set is for word-final 'e', preceded by a consonant. The rule set is basically:

'e' is (usually) dropped. (Exceptions are put into dictionary.)

If there is a single vowel before the consonant (e.g. care: cons.-vowel-cons.-e), pronunciation of 'e' is modified in a standard way. Basically, the vowel (before the consonant) is stressed/diphthongized. The conversions are as follows:

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>As in</th>
<th>Example</th>
<th>Contrast</th>
</tr>
</thead>
<tbody>
<tr>
<td>'a'</td>
<td>'ey'</td>
<td>'ate'</td>
<td>'rate'→'reyt'</td>
<td>'rat'</td>
</tr>
<tr>
<td>'e'</td>
<td>'i'</td>
<td>'eat'</td>
<td>'meet'→'mit'</td>
<td>'met'</td>
</tr>
<tr>
<td>'i'</td>
<td>'ai'</td>
<td>'mine'</td>
<td>'dine'→'dain'</td>
<td>'din'</td>
</tr>
<tr>
<td>'o'</td>
<td>'oh'</td>
<td>'goal'</td>
<td>'note'→'noht'</td>
<td>'not'</td>
</tr>
<tr>
<td>'u'</td>
<td>'eu'</td>
<td>'due'</td>
<td>'cut'→'cut'</td>
<td>'cut'</td>
</tr>
</tbody>
</table>

Table 2 Conversion for Word-final 'e' preceded by a single vowel
4.4. Pronunciation of Indian Names

The issues here are (a) to identify an Indian name and (b) to apply a different set of letter-to-sound rules for them.

Identifying an Indian name in English text is not easy and cannot be done with cent percent success rate. By various means, we cut the corners and increase success rate. We consider if the word is not the first word in a sentence and still starts with capital letter. But even that doesn't ensure that it is a name, as it may be just an emphasized word. Then, we maintain a name dictionary, that may preferably be domain-specific. If we get it there, it is an Indian name. Else, if we get it in the main word dictionary, it is an English word. But all these don't resolve ambiguity on many occasions, particularly with the first word of a sentence that always starts with capital. So, we are trying to understand the letter structure of Indian names and English words that may help better. For example, 'bh', 'gh', word-initial 'sr' etc. are clusters that occur very rarely in standard English and suffixes such as 'tion', 'ous', 'ing' are typically English. Here, it may be noted that although 'Sing' is a common Indian surname, 'ing' here will not be identified in our system as suffix, as the remaining 'root' doesn't have vowel or 'y', so is invalid.

Letter-to-sound conversion for such names also have problems. We consider Indian phoneme set for Indian name. But no uniform spelling convention is observed to code typical Indian phonemes in Roman script. For example, retroflex and dental differentiation or nasalized vowel and nasal consonant differentiation are not done unambiguously. So, there will be errors, till a convention is evolved and observed in practice. However, our experience shows that despite such errors the pronunciation is acceptable if we do a good work on the vowels.

5. PHONOLOGICAL AND PROSODY RULES

5.1. Phonological Rules

Such rules are to be applied, once the pronunciation has been established. These are in the context of phonemes, and not letters.

We hereby describe as example some simple and common ones, regarding suffixes 'ed' and 'es'.

'ed':
- If preceded by 't' or 'd': 'ed' remains unaltered. (e.g. batted, padded).
- Else, if preceded by an unvoiced phoneme (e.g. k, p, s, sh), 'ed' → 't' (e.g. baked → beykt).
- Else, 'ed' → 'd' (e.g. bagged → bagd, zeroed → zeroed).

'es':
- (These rules are for s → z conversion, as per standard English rules. In Indian English s → z conversion is much less frequent.)
- If preceded by 's', 'z', 'ch' or 'j': Usually, 'es' → 'ez'. (e.g. razes → razz).
- Else, if preceded by an unvoiced phoneme (e.g. k, p, t), 'es' → 's' (e.g. bakes → beyks).
- Else, 'es' → 'z' (e.g. bares → beyrz, zeroes → zeroz).

5.2. Prosody Rules

Prosody may be applied at different levels, e.g. at sentence, phrase, word, syllable or phoneme levels. Accordingly, the rules are formulated. In our scheme, some of the rules (e.g. phoneme level ones) are decided at the phoneme-to-speech conversion module. But most of the prosody rules can be determined only at the language processor level. The outputs from such rules are prosody markers, indicating to the phoneme-to-speech module the nature and extent of time-modulation of related acoustic-phonetic parameters.

The three things that we currently modulate are: (a) pitch (b) intensity and (c) phoneme duration. There are separate prosody markers for each of these. A prosody marker, when inserted at a particular place in the phoneme string, indicates qualitative level of change needed for a given parameter from its base value.

The rules derived so far are elementary and very few in number. Research is on for figuring out more. But the rule structure has been fully developed and additional findings can be easily absorbed by simple add-ons.

The rules are primarily based on the type of sentence: declarative, interrogative, or...
exclamatory. We figure out key words in the sentence accordingly. For example, in an interrogative sentence, the last word and also the 'wh' word(s) (e.g. which, who, whom) are keywords. In an exclamatory sentence, an adjective is the keyword. To figure out these, we have included the word status also in the dictionary. The stress contour is decided as per the presence of such keywords. If there is no such detected keyword, some default (e.g. second word of the sentence) is taken. In the absence of a good language parser, currently we are doing with simple clause analysis, mainly on the basis of conjunctions.

6. CONCLUSIONS

We have discussed the rule formulation methodology for Indian English language processor, aided by the CMU pronunciation dictionary and our language tool GENEX. The rules presented as examples are the important ones, and not the complete set. We have applied many of the rules to the system and have checked out their authenticity. But the rule generation with the aid of the tools described is in general faster than their absorption and many are yet to be put into the system. In general, we are in advance stage of rule formulation in morphological analysis and letter-to-sound conversion, whereas in phonological and prosody rules, considerably more are to be done. The need for developing a quality Indian English TTS is acute. We are slowly, but definitely progressing towards that goal.

7. REFERENCES