ISSN 2319-2720

Volume 1, No.3, November – December 2012
International Journal of Computing, Communications and Networking
Available Online at http://warse.org/pdfs/ijccn03132012.pdf

An Improved Automatic Syllabification Rules for BODO Language

Nungleppam Gopil Singh[1], Purnendu Acharjee[2], Prof. P. H. Talukdar[3]
[1]Department of Instrumentation and USIC Gauhati University, India, gopilsn@live.in
[2] Department of Instrumentation and USIC Gauhati University, pbacharyaa@gmail.com
[3] Department of Instrumentation and USIC Gauhati University, phtassam@gmail.com

ABSTRACT

Syllabification acts as a backbone for the unit selection based text-to-speech systems. But with varying language structures the syllabification rules also varies with language. An attempt is made here for improving automatic syllabification rules for BODO language by the addition of extra rules to the current existing rules and implementing it into an algorithm which can be later incorporated into a text-to-speech system. An analysis on the algorithm using 5000 phonetically rich words reveals to yield a comparable result of 99% accuracy as compared to manual syllabification.

Key words : text-to-speech, phonemes, syllable, diphthongs etc.

1. INTRODUCTION

Syllable forms as a gap between a phonemes and words [1]. Various attempts have been made to define a syllable earlier. According to phonetics it is defined with respect to its articulation whereas in phonology it is simply termed as a sequence of phonemes [3].

Upon syllabification a word is broken down into its constituent syllables [5]. As we humans also syllabify a word, as far as possible before speaking if possible and phonemic segmentation if not, text-to-speech systems usually use syllable based approach as a basic unit. As syllable based text-to-speech systems performs better than a phoneme level approach in terms of naturalness and easy in boundary analysis.

Unit selection text-to-speech systems are considered as mature systems now [4]. It automatically performs synthesis of speech from written text with the use of automatic syllabification.

In this paper improved rules for syllabification of BODO language as well as an implemented algorithm for automatic syllabification are presented. The rules are based on the study and analysis of manual syllabification of 3000 phonetically rich words. The degree of accuracy is measured with the perfectness of syllabification, in percentage, with the manual syllabification.

The authors acknowledge Ministry of Communication and Information Technology (MIT), New Delhi, Govt. of India, for sponsorship and funding in the project development of BODO TTS.

2. THE PHONOLOGICAL STRUCTURE OF BODO LANGUAGE

BODO language belongs to the subgroup Tibeto-Burman of the Sino-Tibetan language group. BODO language belongs to the 22 scheduled languages of India as one of the official language of the Indian state Assam. There are six pure vowels, nine vowel glides and sixteen consonants including two semi vowels in native BODO language [11]. The BODO vowels and consonants are shown in the tables below.

Table 1: Pure Vowels in BODO.

	Front	Central	Back
Close	i		ɥ, ɯ
Mid	e		ɔ
Open		ə	

Table 2: Consonants in BODO.

Nature of articulation	Bi-labial	Alveolar	Palatal	Velar	Glottal
Plosive	b	d		g	
	p^h	t^h		k^h	
Nasal	m	n		ɳ	
Fricative		s z			h
Trill		r			
Lateral		l			
Semi Vowel	w		y		

The BODO language also consists of eight diphthongs viz. iɯ, eɯ, əəi, əi, əəɯ, ɔi, ɔɯ, ɯi. It also has a vowel glyph represented by M.

3. SYLLABIFICATION

Syllables are considered as an important unit of prosody [13] as phonological rules and constraints apply at syllable or syllable boundary. Syllables act as a backbone in speech

@ 2012, IJCCN All Rights Reserved

synthesis and recognition. Pronunciation of a phoneme depends on the position in which it is present inside a syllable. Text-to-speech systems basically consist of text to phoneme, prosody and synthesis modules. In all the modules the syllabification plays an important role.

3.1 Methodology

The methodologies followed in the present study are:

a) Examination of BODO syllables structures from linguistic literature.
b) Group discussions with linguists.
c) Dialect variations.
d) Study of previous works performed [2].

3.2 Syllable Structure

BODO words are highly monosyllabic. But also contains poly-syllabic. Syllable structure of BODO phonemes of vowels (V) and consonants(C) are as follows:

1. V
2. VC
3. CV
4. CCV
5. CCVC
6. CVV

3.3 Syllabification rules

In BODO syllables every vowels has at least one vowel sound. Thus a monosyllabic word with single vowel is itself a syllable and need not be syllabified further. An assumption is taken in this rule that diphthongs are considered as a single vowel unit. To further simplify it the syllabification rules developed after study are given below:

i. The syllable boundary of a word having single vowel is at the end of the word.
ii. For vowels having different sounds syllable boundary is marked after the first vowel. i.e. VV then V/V.
iii. For a word having VCV structure then syllable boundary is marked after first vowel. i.e. V/CV.
iv. For a word having VCCC*V structure then syllable boundary is marked after the first consonant. i.e. VC/CC*V.
v. For a word having CCV or CCC structure then no syllable boundary is marked but further syllabification is processed after ignoring the first consonant.
vi. For a word having CVCV structure then syllable boundary is marked after the first vowel. i.e. CV/CV.
vii. For a word having CVCCV structure then check first vowel.
    a) If first vowel is “ i ” then syllable boundary is marked after the first vowel. i.e. CV/CCV.
    b) Else syllable boundary is marked after the second consonant. i.e. CVC/CV.
viii. For a word having CVVCX structure then check X.
    a) If X is a vowel then syllable boundary is marked after the second vowel. i.e. CVV/CV.
    b) Else syllable boundary is marked between the two vowels. i.e. CV/VCC

3.3 Implementation

All the above rules were implemented as a recursive function in a C++ environment with wxWidgets [12] as a graphical front end which checks the presence of syllables in the sub word till the word boundary is reached. The words are processed from left to right.

4. ALGORITHM

In this section BODO syllabification rules presented in section (3.3) are presented as a formal algorithm. The function *syllabify ()* accepts two variables one is the word and the other is the current position of the character and returns a processed syllable text.

There are various functions which are called by the *syllabify ()* function *viz.,*

a) total_vowels () , accepts a word with current position and returns the total number of vowels present in the word after the current position.
b) isvowel () , accepts a word with current position and returns 1 if vowel and 0 if consonant. As well as it returns the length of either vowel or consonant as a reference.
c) total_vowels_bet_cons (), accepts a word with current position and returns the total number of vowels till the next consonant from current position. It also returns the length of the total vowels, length of the first vowel and length of the first consonant, after current position, as reference.
d) total_cons_bet_vowels () , accepts a word with current position and returns the total number of consonants till the next vowel from current position. It also returns the length of the first consonants, after current position, as reference.
e) get_syllables () , accepts a word with current position, syllable boundary and returns the syllable.

The pseudo code of the algorithm is as shown:

```
wxString syllabify (wxString phonemes, int current_index)
{
   phoLen=phonemes.Len();
   temp_IV=is_vowel(phonemes,current_index,&length1);
```

```
  if (total_vowels(phonemes,current_index)==1)
  {
        syll = get_syllables (phonemes,current_index, phoLen) +
        wxT(" ");
        return syll;
  }
  else if(temp_IV==1)
  {
  temp_TC = total_cons_bet_vowels (phonemes, current_index +
  length1, &length2);
  if(temp_TC == 0)
  {
     syll = get_syllables (phonemes, current_index, current_index
     + length1)  +  wxT(" ");
     syll = syll + syllabify (phonemes, current_index + length1);
  }
  else if(temp_TC==1)
  {
     syll = get_syllables (phonemes, current_index,  current_index
     + length1) + wxT(" ");
      syll=syll + syllabify(phonemes,current_index+length1);
  }
  else if(temp_TC>=2)
  {
     syll= get_syllables (phonemes, current_index,  current_index
     + length1 + length2)+ wxT(" ") ;
     syll = syll + syllabify(phonemes, current_index + length1 +
     length2);
  }
}
else
{
  temp_TV = total_vowels_bet_cons (phonemes, current_index +
  length1, &length3, &length4, &length5);
  if(temp_TV==0)
  {
     if(length5!=0)
     {
        syll = syll + get_syllables (phonemes, current_index,
        current_index + length1) + syllabify (phonemes,
        current_index + length1);
     }
      else
      {
        syll = get_syllables (phonemes,current_index,
        current_index + length1 + length3 ) + wxT(" ");
      }
  }
  else if(temp_TV==1)
  {
     tempisi = phonemes.GetChar(current_index + length1);
     temp_IV1=is_vowel (phonemes, current_index + length1 +
     length3 + length5, &length6);
     if(temp_IV1==1)
     {
        syll = get_syllables (phonemes, current_index,
        current_index + length1 + length4)  + wxT(" ");
        syll = syll + syllabify( phonemes, current_index + length1
        + length4);
     }
  else
  {
     temp_IV3 = is_vowel (phonemes, current_index + length1 +
     length3 + length5 + length6, &length7);
     if(temp_IV3 == 1&& phoLen <= current_index + length1 +
     length3 + length5 + length6 + length7 + 1)
  {
  syll = get_syllables (phonemes, current_index, current_index +
  length1 + length4) + wxT(" ");
  syll = syll + syllabify (phonemes, current_index + length1 +
  length4);
}
else
{
  if(tempisi.IsSameAs(wxT("i")))
  {
     syll = get_syllables (phonemes, current_index, current_index
     + length1 + length4) + wxT(" ")
     syll = syll + syllabify(phonemes, current_index + length1 +
     length4);
  }
  else
  {
     syll = get_syllables (phonemes, current_index,  current_index
     + length1 + length4 + length5) +  wxT(" ");
     syll = syll + syllabify(phonemes, current_index + length1 +
     length4 + length5);
    }

  }
}
}
else if(temp_TV==2)
{
  temp_IV2 = is_vowel(phonemes, current_index + length1 +
  length3 + length5, &length6);
  if(temp_IV2==1)
  {
     syll= get_syllables (phonemes, urrent_index, current_index +
     length1 + length3) + wxT(" ");
    syll = syll + syllabify(phonemes, current_index + length1 +
    length3);
  }
  else
  {
     syll = get_syllables (phonemes, current_index, current_index
     + length1 + length4)+ wxT(" ");
     syll = syll + syllabify(phonemes, current_index + length1 +
     length4);
  }
}
}
}
  return syll;
}
```

3. RESULT AND ANALYSIS

On testing the above algorithm on 5000 distinct BODO words, collected from BODO corpus of news and books, and compared it with manual syllabification we get the result of about 19657 syllables with accuracy of 99%. Error analysis reveals that non native BODO words create a problem along with some silent phonemes in words.
The earlier algorithm [2], were not able to syllabify words containing diphthongs and CVCCVC structure. But the newly defined rules and algorithm here solves the diphthong problem as well as words having CVCCVC.

Thus accuracy is improved which will help in better and natural sounding text-to-speech system for BODO language.

ACKNOWLEDGMENT

The authors Nungleppam Gopil Singh, Purnendu Acharjee and Prof. P. H. Talukdar thanks Ministry of Communication and Information Technology (MIT), New Delhi, Govt. of India, for sponsoring the project on development of BODO TTS and giving financial support to the project.

REFERENCES

1. ChandanSarma, U.Sharma, C.K.Nath, S.Kalita, P.H.Talukdar. **Selection of Units and Development of Speech Database for Natural Sounding Bodo TTS System**, *CISP Guwahati*, March 2012.
2. Jyotismita Talukdar, Chandan Sarma, Prof.P.H Talukdar. **Automatic Syllabification Rules for Bodo Language**, *International Journal Of Computational Engineering Research*, vol. 2 issue. 6, pp 110-114, October 2012.
3. Parminder Singh, Gurpreet Singh Lehal. **Syllables Selection for the Development of Speech Database for Punjabi TTS System**, *IJCSI International Journal of Computer Science Issues*, vol. 7, Issue 6, November 2010.
4. R.A. Krakow. **Physiological organization of syllables: a review,** *Journal of Phonetics*, vol. 27, pp. 23-54, 1999.
5. Susan Bartlett, Grzegorz Kondrak, Colin Cherry**. On the Syllabification of Phonemes, Human Language Technologies**, *The 2009 Annual Conference of the North American Chapter of the ACL*, pages 308– 316,Boulder, Colorado, June 2009.
6. Y. A. El-Imam. **Phonetization of arabic: rules and algorithms,** *Computer Speech & Language*, vol. 18, pp. 339–373, October 2004.
7. A.W. Black and K.A. Lenzo. **Building synthetic voice**, http://festvox.org/bsv/, 2003
8. R. Dale et al. (Eds.). **A Rule Based Syllabification Algorithm for Sinhala**, *IJCNLP 2005, LNAI 3651*, pp. 438 – 449, Springer-Verlag Berlin Heidelberg 2005.
9. Couto I., Neto N., Tadaiesky, V. Klautau, A. Maia, R.2010. **An open source HMM-based text-to-speech system for Brazilian Portuguese**, *in Proc. 7th International Telecommunications Symposium Manaus*.
10. Madhu Ram Baro, **Structure of Boro language**, 2008.
11. Phukan Basumatary. **An Introduction to BORO Language**, 1st Ed, 2010.
12. Kevin Hock, Julian Smart, Stefan Csomor. **Cross-Platform GUI Programming with Wxwidgets**, Pearson edition, 2006.
13. Juliette Blevins,**The syllable in phonological theory**,1995
14. George Kiraz and Bernd M¨obius. **Multilingual syllabification using weighted finite-state transducers, i**n Proc. *of the 3rd Workshop on Speech Synthesis*, 1998.
15. Robert Damper. **Learning about speech from data: Beyond NETtalk**, *in Data-Driven Techniques in Speech Synthesis*, pp 1–25, 2001.

@ 2012, IJCCN All Rights Reserved