Dzongkha Text Normalization Algorithm

Uden Sherpa[1]*, Pema Choejey*
Department of Information Technology

Abstract

This paper describes the process of text normalization for Dzongkha Language. It involves tokenization of the Dzongkha text into syllables. After tokenization, non standard words are identified, expanded and passed on to the letter to sound processor by employing a set of rules specific to Dzongkha text normalization.

1. Introduction

This paper describes the ongoing effort for normalization of an unrestricted Dzongkha text for Dzongkha text to speech system. The process of converting text to speech involves inputting text and converting text into speech. That text has to be first normalized before it can be inputted into the text to speech system. Text would contain a wide range of tokens including some non-standard tokens which would first have to be identified and then expanded into its correct form before its sound can be derived. These non-standard words include symbols, abbreviations, acronyms, dates, numbers, ordinal and homographs etc. For example, the token “1997” has to be correctly identified as a date but not as a number. Its correct pronunciation would be “Nineteen ninety seven” as appose to “One thousand nine hundred and ninety seven”. Similarly all non-standard words have to be correctly identified and expanded. This process of identifying unrestricted text and converting it to a consistent form, but without losing its contextual meaning, is called text normalization [1, 2, 3, 4].

2. Methodology

The process for Dzongkha text normalization includes tokenization, token identification, token expansion and grapheme to phoneme conversion. Table 1 shows the list of non-standard words identified in an unrestricted Dzongkha text. Figure 1 explains Dzongkha text normalization process.

Table 1: List of non-standards words

Token type	Example
Symbols	༺, ༻, -, :-,.
Dates	༡/༡༢/༠༩ , སྤྱི་ལོ་ ༢༠༠༩ སྤྱི་ཟླ་ ༡༢ པའི་སྤྱི་ཚེས་ ༡
Numbers	༡, ༢༣, ༧༦༨, ༥, ༩༠

[1] *Corresponding author, contact at uden.sherpa@gmail.com*

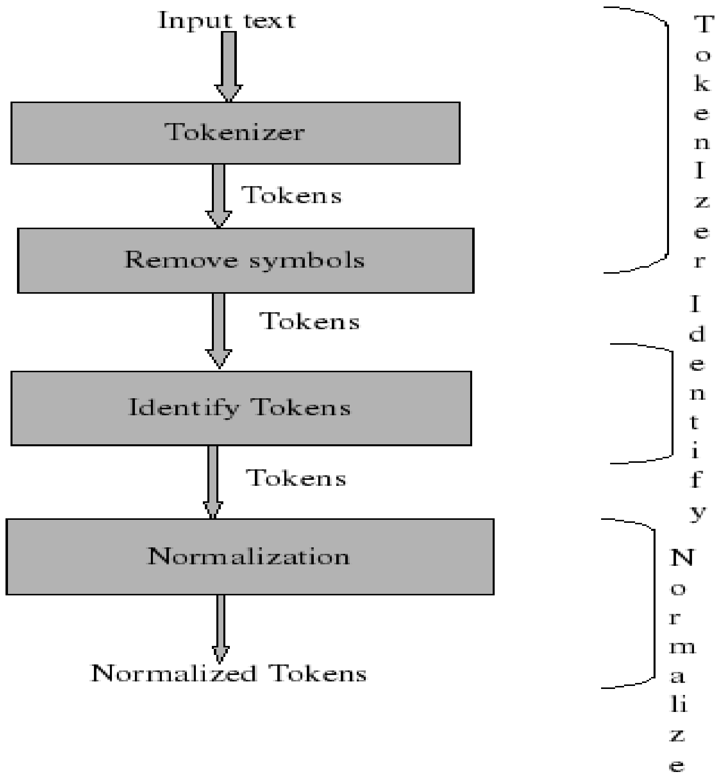

Figure 1: Text Normalization Processes

2.1 Tokenization

In Dzongkha the basic unit for writing is a syllable. There are no inter word space and each syllable is delimited by a syllable marker called a “tsheg”. Sometimes one syllable makes up a word and other times two or more syllables are taken together as a word. There is no rule as to how many syllables are required for the formation of a word. It depends from a particular word to another.

The fact that there is a syllable marker makes segmentation of syllables easier in a sentence. Hence, words can be syllabified using the syllable marker.

An example of a Dzongkha sentence with its pronunciation and meaning:

ཚོད་ག་སྟེ་ལཱ་འབདཔ་སྨོ?
ch-oe-x-0|g-a-x-0|t-e-x-0|l-a-x-0|b-eo-x-0|m-o-x-1|*

You where work do?

Where do you work?

The tokenization of text by Tokenizer is based on syllables. The step wise tokenization algorithm is given below:

Steps:

1. Read from a text file
2. Store it in a variable
3. Break it down into sentence
4. Store all sentence in a list
5. Process the list sentence by sentence (list element)
6. Split the sentence by syllables (by the syllable marker)
7. Store the syllables of each sentence in a separate list
8. Repeat Step 5 to Step 7 till end of sentence list
9. Pass the syllables sentence wise to Normalization module to check for Date and number tokens.
10. Normalize Date and number tokens.
11. Pass Normalized tokens to Grapheme to Phoneme module.

Sample Input:

༡༽ སེམས་རྟོགས་ཁ་རིག་གཞུང་མཐོ་རིམ་སློབ་གྲྭ་ལས་ དུས་ཡུན་ཆུ་ཚོད་༢ ཀྱི་རིངས་ལུ་ ག་ནི་བ་བསིལ་ཉམས་དང་ལྡན་པའི་སྟོང་ཕུ་ཤིང་གི་ཚང་ནང་ལས་ ཁ་ཡར་འཛེགས་ཏེ་ འགྱོཝ་ད་ མཐོ་ཚད་མི་ཊར་ ༢༩༢༥ དང་ལྡན་པའི་ས་གནས་ ལྟག་ལ་ཁ་ལུ་ལྷོདཔ་ཨིནམ་དང་ ཁུར་ཆ་འདི་ཚུ་ཡང་ ཏ་གུ་བཀལ་ཏེ་འབབ་འགྱོ་ནི་ཡོདཔ་ཨིན།

ལྟག་ལ་ཁ་ལས་ ཁ་གྱེན་འཛེགས་ཏེ་ སྟོང་ཕུ་དང་བ་ཤིང་སོགས་ཀྱི་སྦུག་ལས་འགྱོ་སྟེ་ ཆུ་ཚོད་གཉིས་དང་ཕྱེད་ དེ་ཅིག་གི་ས་ཁར་ མཐོ་ཚད་མི་ཊར་ ༤༠༠༠ འབད་མི་བྱི་ལི་ཛི་ལ་ ལུ་ལྷོདཔ་ཨིན།

Output:

c-i-x-0|s-e-m-0|t-o-x-0|kh-a-x-0|r-i-g-0|zh-u-ng-0|th-o-x-0|r-i-m-0|l-o-b-1|dr-a-x-0|l-e-x-0|d-ue-x-0|y-ue-n-0|ch-u-x-0|tsh-oe-x-0|ny-i-x-1|k-i-x-0|r-i-ng-0|l-u-x-0|g-a-x-0|n-i-x-0|b-a-x-0|s-i-x-0|ny-a-m-0|d-a-ng-0|ng-e-n-1|p-ai-x-0|t-o-ng-0|ph-u-x-0|sh-i-ng-0|g-i-x-0|tsh-a-ng-0|n-a-ng-0|l-e-x-0|kh-a-x-0|y-a-r-0|dz-e-g-0|t-e-x-0|j-ou-x-0|d-a-x-0|th-o-x-0|tsh-e-x-0|m-i-x-0|tr-a-r-0|ny-i-x-1|t-o-ng-0|g-u-x-0|j-a-x-0|ny-e-r-0|ng-a-x-1|d-a-ng-0|ng-e-n-1|p-ai-x-0|s-a-x-0|n-e-x-1|t-a-x-0|l-a-x-0|kh-a-x-0|l-u-x-0|lhh-oe-p-0|@-i-m-0|d-a-ng-0|kh-u-x-0|ch-a-x-0|d-i-x-0|tsh-u-x-0|y-a-ng-0|t-a-x-0|g-u-x-0|k-e-l-0|t-e-x-0|b-a-x-0|j-o-x-0|n-i-x-0|y-oe-p-0|@-i-n-0|*

t-a-x-0|l-a-x-0|kh-a-x-0|l-e-x-0|kh-a-x-0|j-e-n-0|dz-e-g-0|t-e-x-0|t-o-ng-0|ph-u-x-0|d-a-ng-0|b-a-x-0|sh-i-ng-0|s-o-g-0|k-i-x-0|b-u-x-0|l-e-x-0|j-o-x-0|t-e-x-0|ch-u-x-0|tsh-oe-x-0|ny-i-x-1|d-a-ng-0|ch-e-x-0|d-e-x-0|c-i-x-0|g-i-x-0|s-a-x-0|kh-a-r-0|th-o-x-0|tsh-e-x-0|m-i-x-0|tr-a-r-0|zh-i-x-0|t-o-ng-0|b-e-x-0|m-i-x-0|j-i-x-0|l-i-x-0|dz-i-x-0|l-a-x-0|l-u-x-0|lhh-oe-p-0|@-i-n-0|*

The * indicate end of sentence. The | denotes the end of a syllable. The 0 and 1 are used to denote normal and high tones respectively. The **x** stands for the missing final consonant. Since the synthesis is based on the format “initial consonant-vowel-final consonant-tone”, where there is no final consonant in a syllable, x takes the place of the final consonant so that the format is preserved. It must be noted that 'x' itself does not have any pronunciation.

2.2 Symbols, Abbreviations and Numeral expansion

Dzongkha text normalization process involves normalization of symbols, dates and numbers.

There are many symbols in text corpus. They are mainly used for text decoration. Therefore, they do not have any pronunciation. These symbols (༺, ༻, -, :-,.) need to be removed.

Dzongkha does not have different ways of speaking dates like in English. For example, 24 -> TWENTY FOUR is a number whereas 24th -> TWENTY FOURTH is a date. But there are tokens (syllables) in the front and in between dates that indicates it is a date.

If the date is in short form like in English, 1/12/09, which is equivalent to ༡/༡༢/༠༩ in Dzongkha, this has to be expanded to its standard form, that is “སྤྱི་ལོ་ ༢༠༠༩ སྤྱི་ཟླ ༡༢ པའི་སྤྱི་ཚེས ༡ ”.

Similarly, digits have to be converted into its letter form before any processing can be done on the text.

Given below is a standard date form which is converted to its equivalent letter form in the normalization process before it is passed on for G2P conversion.

སྤྱི་ལོ་ ༢༠༠༩ སྤྱི་ཟླ ༡༢ པའི་སྤྱི་ཚེས ༡

སྤྱི་ལོ་ གཉིས་སྟོང་ལེབ་དགུ་ སྤྱི་ཟླ བཅུ་གཉིས་ པའི་ སྤྱི་ཚེས་ གཅིག་

c-i-x-0|l-o-x-0|ny-i-x-1|t-o-ng-0|l-e-b-0|g-u-x-0|c-i-x-0|d-a-x-0|c-u-x-0|ny-i-x-1|p-ai-x-0|c-i-x-0|tsh-e-x-0|c-i-x-0|*

2.2.1 Number to letter expansion

A number token is read first and then is broken down into different place values. For instance, a four digit number has four different places – unit, ten, hundred and thousand place. Processing of each place value is done using different functions. So for a unit place, a different processing is done from a tens place and so on. The processing actually looks up the digit in a already stored python dictionary [5] to get its equivalent letters. Below are some of the python dictionaries created for Dzongkha.

dzo_eng_digits = {u"༠":0,u"༡":1,u"༢":2, u"༣":3, u"༤":4, u"༥":5, u"༦":6, u"༧":7, u"༨":8, u"༩":9}

digits = {u"༠":u"",u"༡":u"གཅིག", u"༢":u"གཉིས", u"༣":u"གསུམ", u"༤":u"བཞི", u"༥":u"ལྔ", u"༦":u"དྲུག",

u"༧":u"བདུན", u"༨":u"བརྒྱད", u"༩":u"དགུ"}

tens_helper = {u"༡":u"བཅུ", u"༢":u"ཉེར", u"༣":u"སོ", u"༤":u"ཞེ", u"༥":u"ང", u"༦":u"རེ", u"༧":u"དོན", u"༨":u"གྱ", u"༩":u"གོ"}

place_word = [u"", u"", u"བརྒྱ", u"སྟོང", u"ཁྲི"]

tens_place_word = {u"༡":u"བཅུ", u"༢":u"", u"༣":u"ཅུ", u"༤":u"བཅུ", u"༥":u"བཅུ", u"༦":u"ཅུ", u"༧":u"ཅུ", u"༨":u"ཅུ", u"༩":u"བཅུ"}

The dzo_eng_digits is just for comparision of a Dzongkha digit to an English one. From digit dictionary we get each digits letter form. The tens helper is for those number in tens place. In Dzongkha if the number is ༡༢, we say བཅུ་གཉིས, so now the actual letter form is a combination of tens_helper and digits as shown above in the dictionary. Like wise for all other digits, equivalent letter form can be obtained by combination of above dictionary items. The combination depends on the place the digit occupy in the number.

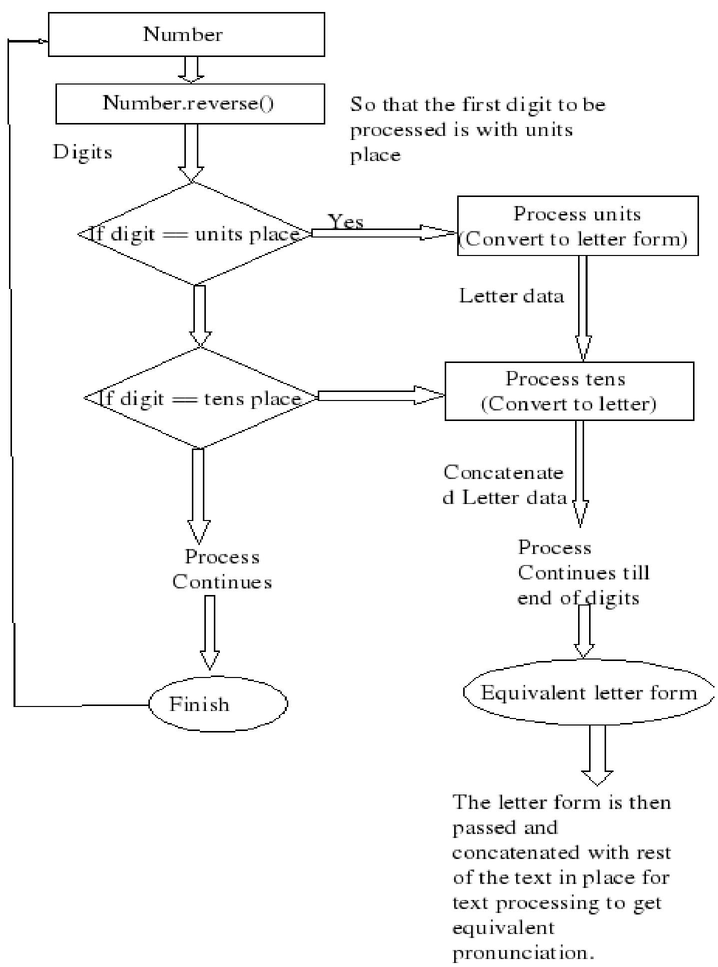

Figure 3: Number to letter conversion

2.2.2 Date Processor

A pattern matching is done to find date tokens by the date processor. If the token matches and returns true, the year token is converted to its standard form of four digits instead of two digits. Then, the month and the day tokens are appended in the front of the year so that it is in its expanded form. Once it is in its standard form the digits are converted to its equivalent letter for processing.

Sample code for matching ༡/༡༢/༠༩ date

```
X =  re.compile ("[༠༡༢༣༤༥༦༧༨༩]*/[༠༡༢༣༤༤༥༦༧༨༩]*/[༠༡༢༣༤༥༦༧༨༩]*", re.L)
re.search (x, token)
```

Input to Date processor: ༡/༡༢/༠༩
Output: སྤྱི་ལོ་ ༢༠༠༩ སྤྱི་ཟླ ༡༢ པའི་སྤྱི་ཚེས ༡

Input to the Number to letter Processor: སྤྱི་ལོ་ ༢༠༠༩ སྤྱི་ཟླ ༡༢ པའི་སྤྱི་ཚེས ༡
Output: སྤྱི་ལོ་ གཉིས་སྟོང་ལེབ་དགུ་ སྤྱི་ཟླ བཅུ་གཉིས་ པའི་ སྤྱི་ཚེས་ གཅིག་

Input to Grapheme To Phoneme Module: སྤྱི་ལོ་ གཉིས་སྟོང་ལེབ་དགུ་ སྤྱི་ཟླ བཅུ་གཉིས་ པའི་ སྤྱི་ཚེས་ གཅིག་
Output: c-i-x-0|l-o-x-0|ny-i-x-1|t-o-ng-0|l-e-b-0|g-u-x-0|c-i-x-0|d-a-x-0|c-u-x-0|ny-i-x-1|p-ai-x-0|c-i-x-0|tsh-e-x-0|c-i-x-0|*

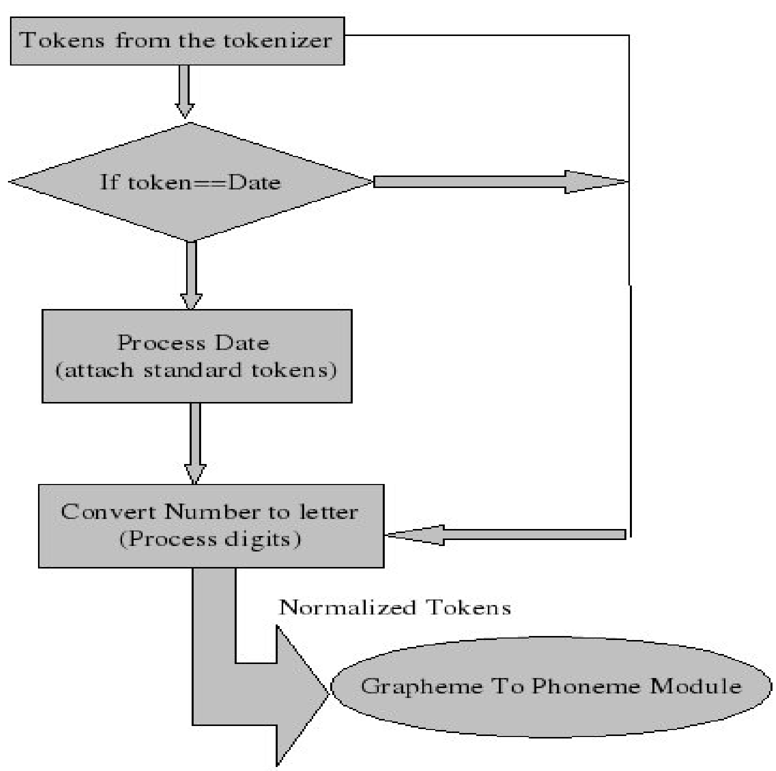

Figure 4: Date processing

2.3 Grapheme to phoneme conversion

In Dzongkha Text to Speech (TTS) system, Grapheme to Phoneme (G2P) conversion is implemented using a look up dictionary. The dictionary just consists of Dzongkha syllables along with its pronunciation.

3. Result

The above normalization algorithm is used to implement the Dzongkha TTS prototype [6]. The accuracy of the text analysis module (syllabification, normalization and grapheme to phoneme conversion) is shown in Table 2.

Table 2: Results of Text Analysis Module

No. of sentence	No. of syllables	No. of non-standard Words	No. of tokens outputted	No. of non-standard words correctly normalized
5	216	8	226	8
50	352	1	347	1
100	739	1	731	1
500	3469	3	3444	3

The difference in number of tokens input and the number of tokens output is because the non-standard words are first expanded before tokenization and this causes in the number of tokens outputted to be slightly more in the first case. On subjective evaluation, the accuracy of text normalization is more than 98%.

4. Conclusion and Future work

The text normalization algorithm was developed and implemented in Dzongkha TTS prototype. In future, an automatic token labeler may be investigated so that the normalization will become more accurate and better. Word and phrase detection algorithms may be explored to further improve the accuracy and performance.

5. References

[1] “Text Normalization” http://en.wikipedia.org/wiki/Text_normalization
[2] Document Centered Approach to Text Normalization www.scils.rutgers.edu/~muresan/551_IR/2004_Spring/Presentations/indexingTamburin.ppt

[3] Firoj Alam, S.M. Murtoza Habib, Mumit Khan, “Bangla Text Normalization”, www.crulp.org/clt09/download/**Text-Normalization**-Sys_Firoj_**Bangla**.pdf

[4] K. Panchapagesan , Partha Pratim Talukdar, N. Sridhar Krishna, Kalika Bali, A. G. Ramakrishna, “Hindi Text Normalization”, www.cis.upenn.edu/~partha/papers/KBCS04_HPL-1.pdf

[5] Python, www.python.org

[6] Sherpa, U., Pemo, D., Chhoeden, D., Rugchatjaroen, A., Thangthai, A., and Wutiwiwatchai, C. “Pioneering Dzongkha Text-to-Speech Synthesis”, International Conference on Speech Database and Assessments, November, 2008.

Acknowledgement

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

We would like to thank the Human Language Technology (HLT) of National Electronics and Computer Technology Centre (NECTEC), Thailand, particularly Dr. Chai Wutiwiwatchai and his team members for their technical guidance and for supporting us to design and develop the Dzongkha TTS.