ระบบสังเคราะห์เสียงพูดภาษาดองกะ (ภูฏาน) ระยะที่ 2

Dzongkha Speech Synthesis System – Phase II

Dechen Chhoeden[1], Chungku[1], Chai Wutiwiwatchai[2], Ananlada Chotimongkol[2],
Ausdang Thangthai[2], Anocha Rugchatjaroen[2]

[1]Department of Information Technology,
Ministry of Information and Communication, Bhutan

[2]Human Language Technology Laboratory
National Electronics and Computer Technology Center (NECTEC), Thailand

---

Abstract

This report describes a construction of an advanced Dzongkha text-to-speech (TTS) system using an HMM-based method. We have developed additional modules like word segmentation and phrase prediction which are integrated with the earlier Dzongkha TTS prototype to improve the quality of the synthesized speech. The procedure of the integration is explained in the report.

Keyword – Dzongkha TTS system, HMM-based synthesizer, duration modeling, phrase boundary prediction

บทคัดย่อ -

คำสำคัญ -

Technical Challenge

The main challenge of this research is how to apply the knowledge and experience that the researchers at the Human Language Technology Laboratory have gained from developing a Thai text-to-speech (TTS) system to build a TTS system for a new language, in this case Dzongkha. Dzongkha is quite different from Thai both in terms of a written script and a sound system.
In the first phrase of the development of the Dzongkha TTS system, focused more on the linguistic aspect designing a phoneme set for Dzongkha and collecting a speech corpus. For the technology aspect, a simple text precessing module was constructed and a speech synthesizer was trained using a Hidden Markov Model-based Speech Synthesis System (HTS) framework. In the second phrase of the development, we have focused on applying advanced natural language processing techniques to improve the quality of the synthesized sound. This is done by adding a prosody generation module for proper phone duration and pausing which requires a development of many sophisticated natural language processing algorithms including a word segmentation algorithm, a POS tagger, a phone duration predictor and a phrase boundary predictor. These algorithms required a deeper understanding of a language to annotate both speech and text corpora with POS tags, word and phrase boundaries in order to train various statistical models. and a preparation of both annotated speech and text corpus to train statistical models. With this advanced algorithms, we could improve the quality of the synthesized sound as measured by a subjective listening test, Mean Opinion Score (MOS), from 2.41 to 2.98. This evaluation has been done by native speakers of Dzongkha who use this system back in Bhutan.

1. Introduction

A Bhutanese or Dzongkha text-to-speech synthesis (TTS) system has been developed under the collaboration between Department of Information and Technology (DIT), Bhutan, and National Electronics and Computer Technology Center (NECTEC), Thailand. This project started in 2008 with a matching funding from Bhutan through the PAN Localization project and from Thailand through the Asian Applied Natural Language Processing for Linguistic Diversity and Language Resource Development project (ADD) at NECTEC. In addition to the usual benefits of a TTS system, such as its applications in assistive technologies and mobile devices, the Dzongkha TTS system also helps with cultural preservation by integrating a local language into information and communication technologies, and thus into people's daily lives.

The project is composed of two development phases. In the first phase, which ended in 2008, a simple Dzongkha TTS system was constructed. The first prototype of a Dzongha TTS was developed using Hidden Markov Model-based Speech Synthesis System (HTS) and a simple text processing technique that incorporates a syllable segmentater and a syllable-level pronunciation dictionary. The synthesized speech produced by the first TTS prototype did not sound natural as reflected by the Mean Opinion (MOS) score that was taken during the evaluation process. Meaning that the resulting speech sounded is quite robotic. This can be due to the fact that at that time word boundaries, Part-of-Speech(POS) and phrase boundaries which are integral components of natural speech were not taken into account.

From the shortcoming of the first Dzongha TTS prototype, in the second phase, which started in 2009 and ended at the beginning of 2010, an enhanced Dzongkha TTS system in terms of vocabulary coverage and synthesized speech quality has been developed. The coverage issue is resolved by including more unique syllables in the pronunciation dictionary. To improve the synthesized speech quality, the size of the speech database has been increased and a prosody generation module has been included in the enhanced TTS system for proper phone duration and pausing. Word segmentation and Part of Speech (POS) tagging modules have also been implemented as they provide useful information for the prosodic generation module. We compared the enhanced Dzongkha TTS system to the previous system in terms of the naturalness of the synthesized speech using the Mean Opinion (MOS) score on a subjective listening test.

2. Objective

1. To develop a prototype of a Dzongkha TTS system version 2 which is improved over a Dzongkha TTS system version 1 on the following aspects:
    - A better coverage on word pronunciations
    - Adding a prosodic generation module in order to produce more natural sound which includes the following components:
        - A word boundary analyzer and a phrase boundary analyzer
        - A phone duration predictor
2. To evaluate the performance of the Dzongkha TTS system version 2

3. Current status

In the second development phase which aims at developing a Dzongkha TTS system version 2, both DIT and NECTEC agreed on another 2-month research visit at NECTEC to learn and develop a prosodic generation module. Prior to the training period, a text corpus, which contains approximately 40,000 words, was annotated with information such as word boundaries, phrase boundaries and parts of speech (POS) of each word. These features are necessary for training machine learning algorithms to predict an appropriate duration of each phone and phrase

boundaries in a stream of text for proper pausing. A speech corpus was also enlarged to include 400 more utterances to improve the quality of the trained HMM synthesizer. So, the current speech corpus contains 900 utterances. The syllable pronunciation dictionary was expanded from 4,000 to 5,000 entries to increase pronunciation coverage.

During the training period, 2 Bhutanese researchers learned all the necessary techniques for developing a phrase breaking module and a duration prediction module and implement these modules using the annotated text corpus that have been prepared. The schedule of the 2-month training is shown below.

Training schedule:

Activity	Week							
	1	2	3	4	5	6	7	8
Clean and format a text corpus	■	■						
Enlarge and label speech corpus	■	■						
Learn necessary theories and algorithms		■			■			
Develop a word segmentater and a POS tagger			■	■				
Develop a phrase boundary analyzer					■	■		
Develop a phone duration predictor			■	■				
Integrate components, evaluate and write a report							■	■

The first and the second activities, corpus cleaning and formatting, took longer than we expected. These activities are necessary preparation steps before all the required information can be extracted and used as features for training machine learning algorithms used by components such as a phrase boundary analyzer and a phone duration predictor. Since the text corpus was initially created to be used in another natural language processing research at DIT, some changes had to be made to make it suitable for our research and some had to be done manually. Here are examples of the problems that we found in the corpus.

- Missing syllable markers especially at the end of a word
- Inconsistency at whether there should be a syllable marker or not in the case of abbreviation, name and special symbol

All the components were evaluated and integrated into the Dzongkha TTS system version 2. The naturalness of the synthesized speech was also evaluated. The result shows that the Dzongkha TTS system version 2 is better than the previous version. Nevertheless, the system could be further improved by adding more training data both the speech corpus for training the HMM synthesizer and the text corpus for training the word segmentater, the POS tagger, the phrase boundary analyzer and the phone duration predictor.

4. Related theories

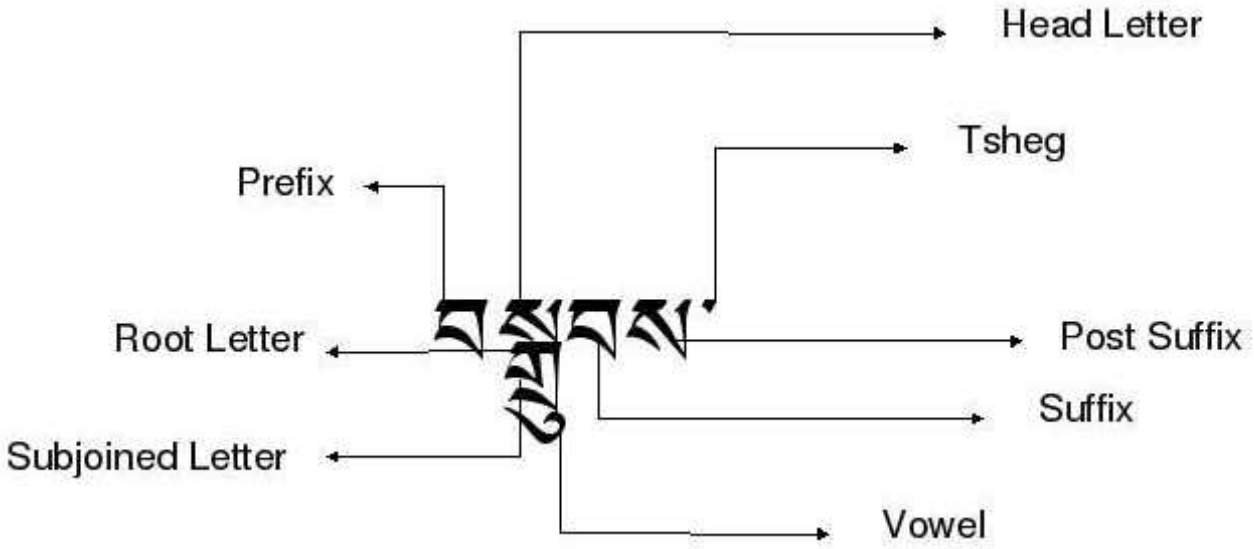
4.1 Bhutanese or Dzongkha language

Dzongkha is the official language of Bhutan. Dzongkha and its dialects are the native tongue of eight western districts of Bhutan. It belongs to the Sino-Tibetan family of languages.

Infact written Tibetan is very similar to Dzongkha. The script for writing is the same. Spoken form, though, is different from Tibetan, in that the vocabulary set used are different from each other leaving very few exceptions. It was found that spoken Dzongkha could be represented by 30 initial consonants, 5 initial consonant clusters, 10 vowels, 10 dipthongs, 8 final consonants and 2 tons. More detail on Dzongkha's phoneme inventory can be found in (Sherpa et al., 2008).

A syllable is the basic unit of meaning or morpheme in Dzongkha. Syllables are normally delimited by a delimiter named as the 'tsheg' bar or another punctuation. Each syllable contains a root letter (ming-zhi) and may additionally have any/or all of the following parts in the given order: prefix; head letter; sub-fixed letter; vowel sign; suffix and postsuffix, as shown in Figure 1.

Figure 1: The writing system of a Dzongkha syllable.



For ending a sentence or phrase, a 'shed' marker (|) or a question mark is placed. Figure 2 shows an example of Dzongkha script with syllable and sentence markers. Dzongkha writing has no word boundaries. Dzongkha words can comprise of one or more syllables. Even though Dzongkha has sentences clearly delimited by sentence boundaries, there are no explicit rules to mark phrase boundaries. These issues are taken into consideration and necessary steps are taken to built word segmentation and phrase prediction models.

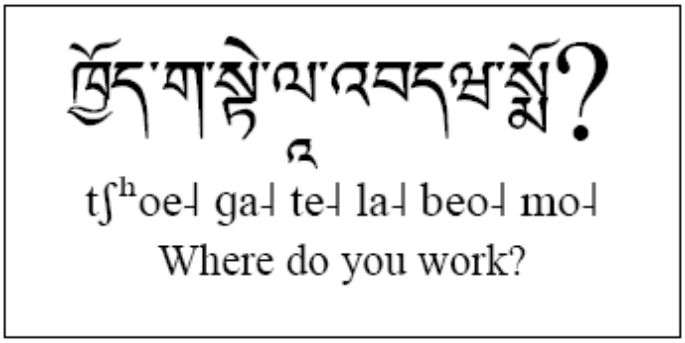


Figure 2: An example of Dzongkha script showing syllable delimiters 'tsheg' and a question marker at the end.

4.2 TTS generic development framework

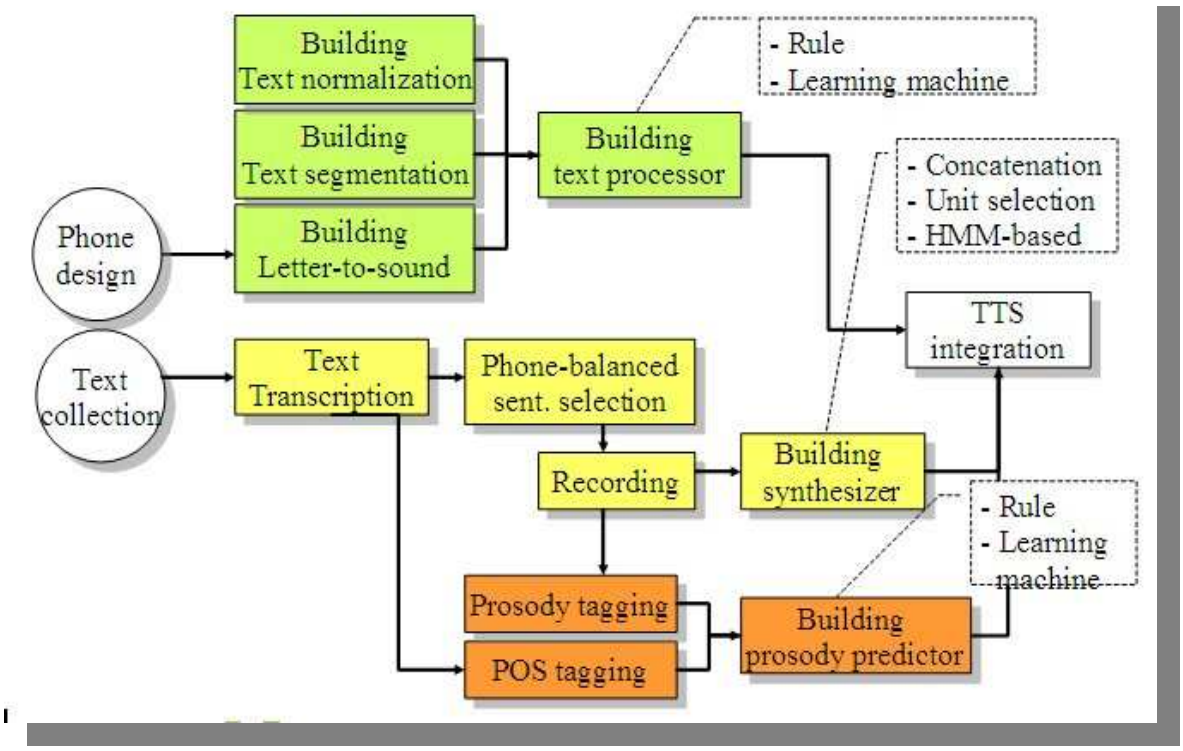


Figure 3: TTS generic development framework

5. Prototype specification

Software specification

1) Only Dzongkha text is considered.
2) the system has been tested only on windows XP and Linux.
3) Only 8000 syllables in synthesizer's dictionary could be synthesized.

Hardware specification

CPU:Pentium III or higher, Pentium 4 is recommended
RAM: 256 MB or higher
Harddisk: Full 170 MB, Compressed 70 MB
Operating system: Window 2000, NT, XP

6. Design and methodology

In this research, we have developed a Dzongkha TTS system version 2 which is improved over a Dzongkha TTS system version 1 along two main aspects vocabulary coverage and synthesized speech quality. The development process for each aspect is discussed in detail below.

6.1 Increase vocabulary coverage

Since Dzongkha is a syllable-separated scripting, a simple text-processing module which takes an input string and converts each separated syllable in the string into its pronunciation is utilized in the Dzongkha TTS system version 1. With quite limited number of unique syllables

appeares in Dzongkha, a syllable-pronunciation dictionary is used for grapheme-to-phoneme (G2P) conversion. To make the Dzongkha TTS system version 2 able to pronounce more words, more unique syllables were included in the pronunciation dictionary. About 900 new syllables were transcribed with their respective pronunciation (a sequence of phones). Transcription of syllables involves marking each phone in the syllable with special symbols borrowed from the IPA system.

For e.g, the syllable 'ཁཅ' is transcribed as 'kh-a-b-0', which consists of three phonemes and '0' indicates normal tone ('1' indicates high tone).

6.2 Improve synthesized speech quality

To improve the synthesized speech quality, we have increased the size of the speech database used for training the speech synthesizer. We have also incorporated a prosody generation module to the Dzongkha TTS system version 2 to make the synthesized speech sounds more natural with more appropriate phone duration and pausing. The following diagram illustrates the process that was used to improve the quality of the synthesized speech in the Dzongkha TTS system version 2.

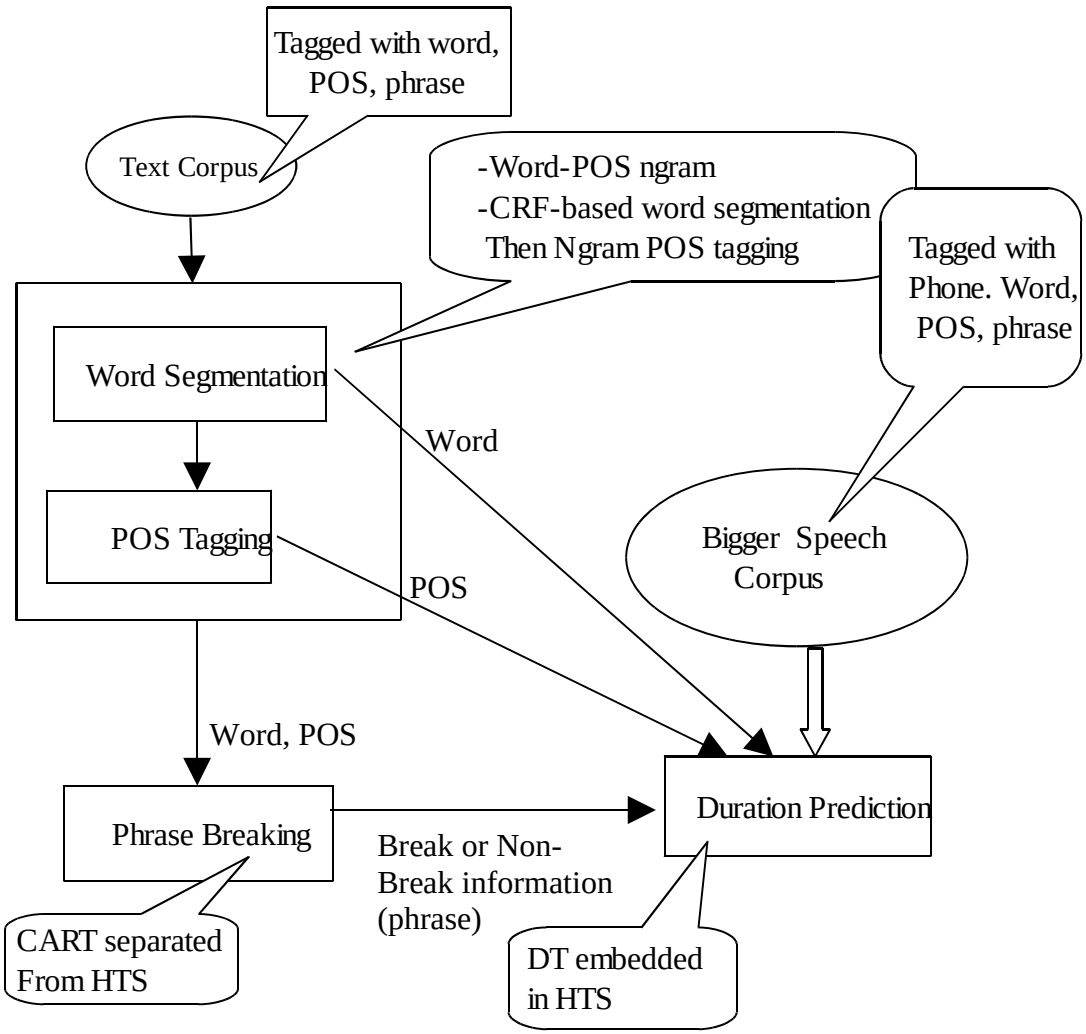


Figure 4: The development process of the Dzongkha TTS system versions 2 (training phase)

From the diagram, we can see that word boundary and Part of Speech (POS) are crucial

information for predicting a phrase boundary. Phrase boundaries together with word boundaries and POS tags are, in turn, crucial information for predicting phone duration. Hence, in order to predict phrase boundaries and phone duration in the prosodic generation module, a word segmentater and a POS taggers are necessary components for extracting features during the executing phase of the TTS system.

The training phase of the Dzongkha TTS system versions 2 composes of 5 steps, data preparation and the development of 4 necessary components: word segmentation, POS tagging, phrase boundary prediction and phone duration prediction. The detail of each step is discussed below.

6.2.1 Data preparation

Both a speech corpus and a text corpus are necessary for training a better HTS (HMM-based speech synthesis system) synthesizer and training all the components (e.g. a phrase break predictor, a phone duration predictor) in the prosodic generation module.

Speech corpus

More recording data were collected and labeled with necessary information. Previously 509 best sentences covering all the Dzongkha phones in the phoneme dictionary were recorded. We selected 434 additional sentences (along with their corresponding transcription) covering all the phones in the language for recording. The sentence selection process is similar to the process described in (Sherpa et al., 2008) and (C. Wutiwiwatchai et al., 2007). The idea is to gather iteratively a sentence having the most distinctive tonal-diphones (two connected tonal-phonemes) not presented in the selected set. The selection process is terminated when all tonal-diphones in the text corpus are all presented in the selected set.

These sentences were recorded by a female native speaker of Dzongkha and the resulting wave files were edited to a 44 Khz and 16 bits format. Now in total we have 943 wav files named 001.wav, 002.wav and so on till 943.wav in our speech corpus database. These wav files are needed to feed into the HTS for training. So the speech database has a total of 943 wav files.

Besides the text transcription and pronunciation script (the corresponding phone sequence) , each wave file is labeled with word and phrase boundaries. Each word in the transcript was also tagged with its respective POS tag. This is a crucial step in our TTS building phase as this data will be used in the labeling process to train a HTS model.

Text corpus

Dzongkha Corpus is a collection of text with 600,000 syllables (400,000 words approximately) from different genres like newspaper articles, samples from traditional books, novels, dictionary, magazines. So as to make it a representative of every linguistic phenomena of Dzongkha. At present corpus is separated into three domains mainly world affairs, Political Science and Arts.

A small part of this corpus which contains approximately 40,000 words, was annotated with word boundaries, phrase boundaries and POS tag of each word. These features are necessary for training a phrase boundary predictor and a phone duration predictor as mentioned earlier.

6.2.2 Word segmentation

In Dzongkha, the smallest unit of text that is meaningful is the syllable. Dzongkha texts are marked with syllabic and sentence boundaries; however, there is no word boundary or word delimiting character as described in Section 4.1. A word can comprise of one or more than one syllable. So a word segmentation algorithm must be devised to break the text into words.

Word segmentation involves the process of tokenizing a string of text into words which is one of the fundamental tasks in any NLP applications. Many different methods are available for

word segmentation nowadays. The accuracy of each of the methods will differ from language to language. Hence a language needs to adopt the best algorithm that gives the best accuracy. For Dzongkha TTS we have adopted the longest string matching method for segmenting words. We have yet to explore and evaluate other algorithms to find the best match for our language. A machine learning model called CRF was tested with our data, but our text corpus was too small to train the model. This particular model has shown to post better accuracy of word segmentation in almost all languages. So in the future we plan to increase the size of the text corpus and then train the CRF model to achieve better word segmentation accuracy as being demonstrated for Thai word segmentation (Haruechaiyasak, et al., 2008).

From the limited amount of training data, we used the longest matching technique which is a dictionary-based method of word segmentation. This method depends on a match between word entries in the dictionary and a string of characters in an input text, and segments the text accordingly. Given the input text the algorithm scans the characters from left to right and finds the longest matching entry in the dictionary.

Evaluation

F-measure was used to evaluate the overall performance of a word segmentation algorithm. The F-measure is taken by calculating by using the following equations.

F-Measure = 2 (Precision x Recall) / (Precision + Recall)
Precision = Corr / OutputWord
Recall = Corr / RefWord

When comparing between an output of a word segmentation algorithm and a word-segmented reference,

Corr = The number of words in the output that is correctly segmented;
RefWord = The total number of words in the reference;
OutputWord = The total number of words in the output.

To evaluate the performance of the longest matching algorithms on Dzongkha text, we created a dictionary from all unique words in the text corpus described in Section 6.2.1. The dictionary contains 30,856 words. The test set is the transcript of 943 wave files in the speech corpus also described in Section 6.2.1. The test set contains 8,701 words. The result from the evaluation is presented in the following table.

Precision	86.73
Recall	84.67
F-measure	**85.69**

Table 1: The performance of the longest matching algorithm

The problem with this method of word segmentation is that there may be cases where the words in the input text may not match any word in the dictionary. This problem is known as an unknown word or out-of-vocabulary (OOV) word problem. Secondly, there may be more than one ways to segment the input sequence of characters. For these two problems, a more sophisticate word segmentation algorithm such as a machine learning one is required to solve the problems.

Another problem is inconsistencies in the corpus. For example, a syllable marker is missing especially at the end of a word. There is also inconsistency at whether there should be a syllable

marker or not in the case of abbreviation, name and special symbol. Also when there are numbers and special symbols in the input text, the algorithm fails to separate those numbers and symbols from the words. For these two problem, if the inconsistencies can be resolved, we can write a set of rules to find word boundaries for those special cases.

6.2.3 POS tagging

POS Tagset

The POS tag set for Dzongkha language is developed based on Penn Guidelines and it is available in Penn-Tree-bank tags. The tag set consist of total 66 different tags with information about number, case, case, punctuation etc (Chungku et al., 2010). Major open word class includes seventeen tags which includes all kinds of noun, verb, adjective, adverb, interjection.

POS tagging technique

For automatic POS tagging, we used a tagger called *TreeTagger* (Schmid, 1994). TreeTagger is a tool for annotating POS and lemma information of a given text. TreeTagger is a probabilistic POS tagger whose algorithm is based on a decision tree. The tagger consists of two programs: train-tree-tagger and tree-tagger. Train-tree-tagger is used to create a language model, or a so-called parameter file, from a lexicon describing tokens and their respective tags and a manually tagged corpus. Tree-tagger is then used the parameter file to perform automatic tagging on the given input texts. The input of the tree-tagger must be tokenized first, namely, the input must be in a one-token-per-line format as shown in the following example.

Example of Dzongkha TreeTagger input text

སྤྱི་རྒྱ་
དཔལ་སྦྱིན་རྫོང་ཁ་
།
སྤྱི་རྒྱ་རྫོང་ཁ་
འདི་
༡༩༩༩
ལྷ་

Example of Dzongkha TreeTagger output text

སྤྱི་རྒྱ་	NNP
དཔལ་སྦྱིན་རྫོང་ཁ་	NNP
।	PUN
སྤྱི་རྒྱ་རྫོང་ཁ་	NNP
འདི་	DT
༡༩༩༩	ND
ལྷ་	PP

The work on automatic part of speech tagging started with annotation of first 20,000 words manually as a training data. During the manual tagging, some words are hard for a linguist to disambiguate reliably. In order to keep training data consistent such word are tagged according to its possible tag. In case of religious text lots of rules were considered to get the word tagged accurately. The following example shows a raw Dzongkha text and its corresponding POS annotation that was done manually.

Raw text

འབྲུག་དང་བཅའ་དྲྭ་ཤ་གཉིས་ཀྱི་བར་ན་ཟུང་འབྲེལ་ལ་ཚོང་འབྲེལ་གྱི་གསུང་གྲོས།

Which was annotated as:-

འབྲུག་ /NNP དང་/CC བཅའ་དྲྭ་ཤ་ /NNP གཉིས་ /CD ཀྱི་/CG བར་ན་/PRL ཟུང་འབྲེལ་/NN ཚོང་འབྲེལ་ /NN གྱི་/CG གསུང་གྲོས /NN ।/PUN

note: for the word separation, words were identified according to its meaning in the sentence. So most of the words identified here cannot be found in lexicon.

With this 20,000 words of training corpus, an initial POS tagger for Dzongkha has been created. This tagger was used to tag 5,000 more words automatically. The output from the tagger was manually corrected and found that the accuracy of the tagger is around 80%. This initial tagger was used during the development of the Dzongkha TTS system version 2. However, a better POS tagger trained with approximately 40,000 tokens was reported to achieve 93.1% accuracy (Chungku et al., 2010).

6.2.4 Phrase boundary prediction

It is important to predict phrase breaks (or phrase boundaries) in any TTS system to make the synthesized speech sounds more natural, that is, for a long utterance it should be broken down into meaningful sub-utterances. Dzongkha text has no explicit word and phrase boundaries. We have an advantage where Dzongkha text at least have boundaries at syllabic and sentence levels. At every sentence end, it is certain that a speaker must pause and then start reading the next sentence, however it is unsure of the pattern of phrases uttered within a sentence. In Dzongkha speech phrase breaks can occur between words, spaces and symbols. Almost all of the spaces and punctuation marks indicate phrase boundaries.

To predict phrase breaks in a Dzongkha sentence, we utilize a machine learning algorithm called *Classification And Regression Tree (CART)*. CART is a learning algorithm that is based on a binary decision tree and has been applied widely to the task of phrase boundary prediction (Hansakunbuntheung et al., 2005). The advantage of CART is its ability to manipulate both symbolic and real-value features. Furthermore, the model can efficiently handle sparse training data which is the case of Dzongkha where a training corpus is relatively small.

We use POS information, features at word and syllable levels to train a CART tree to predict whether to insert a pause at the current location or juncture. These features are described in more detail in the following table. The last two rows in the table are the possible output of the CART tree.

Feature	**Description**
POSL2,POSL1,POSR1,POSR2	POS with respect to the position of the current juncture. For e.g, POSL1 is the POS of the word one place on the left hand side of the current juncture. POSR1 is the POS of the word one place to the right with respect to the position of the current juncture. [each]
CurrSylInPhr	No. of Syllables between the current juncture and the beginning of the current sentence.
CurrWrdInPhr	No. of Words between the current juncture and the beginning of the current sentence.
WrdInPhr	Total no. of words in the current sentence.
SylInPhr	Total no. of syllables in the current sentence.
NB	Non-break for junctures that is not a phrase break.
B	Break for junctures that is a phrase break.

Table 2: Required Features for each word

To extract necessary features, two corpora were utilized. Both of them contain the same text data but are annotated with different information. The text in the first corpus was segmented into words and also tagged with POS information. The second corpus, on the other hand, was manually annotated with phrase boundaries. The annotations from both corpora had to be merged to create a training set. This was done using a simple merge program as shown in Figure 5. Furthermore, the number of syllables in each word, which is another feature, was also added to the training set. At the end of this process, we get a file with the aforementioned details which is the main input file for extracting vital features (described in Table 2) using the feature extraction program. These important features are necessary to help predict phrase boundaries given a Dzongkha text input. The entire process of building a phrase boundary predictor is illustrated in the following diagram (Figure 5).

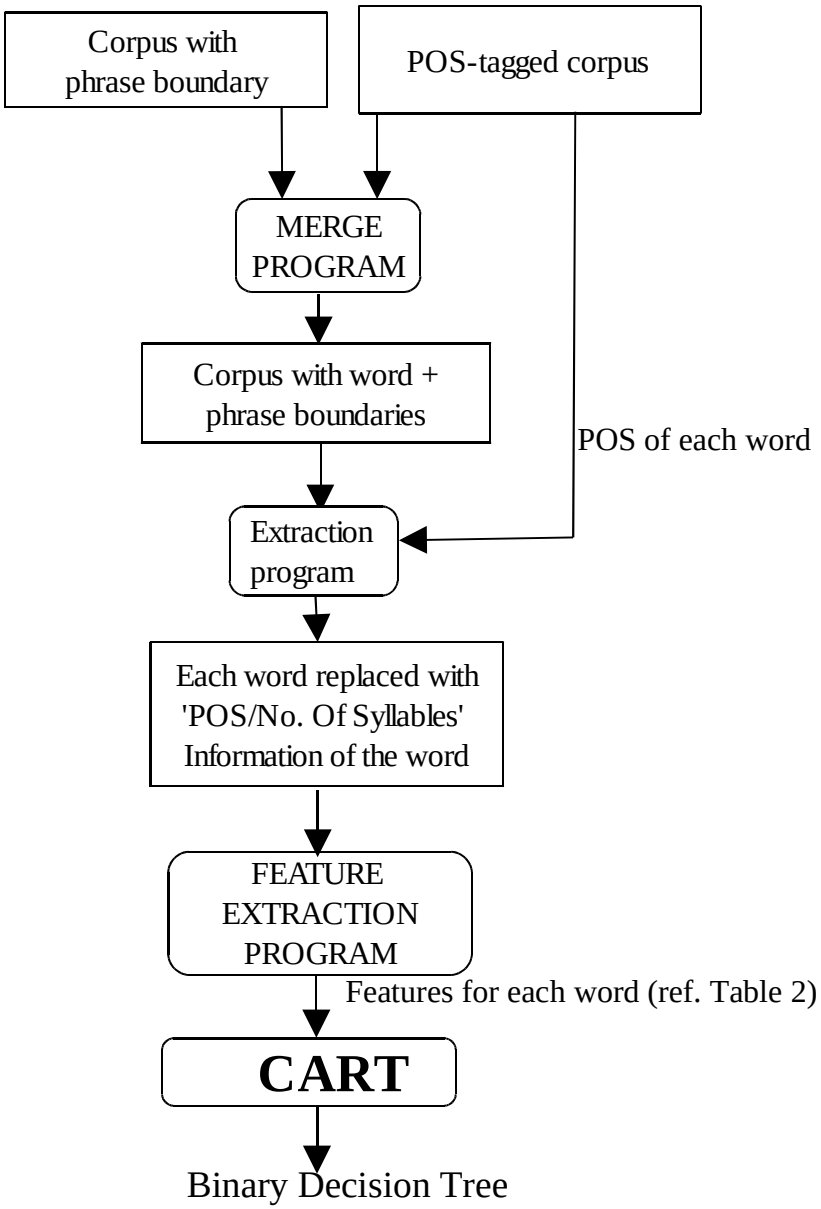


Figure 5. A phrase boundary predictor training process using CART

Using CART, the extracted features are used to create a decision tree which can then be used to automatically predict phrase boundaries for the given text input.
CART takes the whole input file (with features) as one input and it also takes 10 percent of the same file input as the second input. By using these two inputs a binary decision tree is built by CART. It basically looks at the Break or Non-Break status of each word and puts this into a decision tree along with the other features. In the main executing phase, this decision tree is used to predict B or NB features of each word of the input text. Therefore given an input text, the output from the 'phrase prediction' process will output a file containing phrase non-break (NB) or phrase break (B) information for each word in the text input. So if a word is the last word in a phrase, it will obviously be predicted as the area where the phrase break occurs and will be marked as 'B' whereas the other words will be marked as 'NB'. This output file will then be used in the 'labelling' process for training TTS, i.e, the B and NB features will then be used to train HTS to

help produce synthesized speech. The output speech will contain silences wherever the phrase breaks were predicted in the process.

Evaluation

The performance of a phrase boundary predictor is measures in terms of percentage of correct-predicted breaks and correct-predicted non-breaks.

As mentioned earlier, two types of Dzongkha text corpus were used. Now the corpus, apart from the normal words, contains punctuation marks, symbols, numbers, etc. We experimented three types of corpus data to compare the accuracy of the phrase prediction model incorporated.

1)Data-set without removing any punctuation, symbols and numbers

For this experiment, a test data which is about 10 % of the training data was used along with the actual training data. The training data is the final data set produced by the feature extraction program. The training data (train.txt) is a dataset of 37537 words (9022 phrases and 2743 sentences) and the test data (test.txt) is a training data subset containing 3719 words. An accuracy of 88.420% was achieved using precision and recall.

2) Removed punctuation & symbols, ignored numbers

All the punctuation marks like brackets and question marks were removed from the two corpus data sets (corpus1 and corpus2) before merging the two to get a merge file. Which was then used by the feature extraction program to extract the ten features as depicted in the table. The training data (train.txt) is a dataset of 33683 words and the test data (test.txt) is a training data subset containing 3186 words. An accuracy of 88.871% was achieved using precision and recall.

3) Ignored the word that had more than two POS features missing in the final data set for the same corpus data as the second experiment

The training data (train.txt) is a dataset of 33628 words and the test data (test.txt) is a training data subset containing 3182 words. An accuracy of 88.8520% was achieved using precision and recall.

Result:

From the experimental data, regardless of the accuracy, the first type of data in experiment no. 1 is used. That's because in real life all kinds of text-data will be present. We cannot exempt any symbols, numbers or punctuation.

6.2.5 Phone duration prediction

In phase-II, in-order to improve the quality of synthesized speech produced given a text input, the synthesizer needs to identify important features like word boundaries, phrase boundaries and POS of each word. These extra features would make the speech more natural. To do that we needed to manually tag the word boundaries and the phrase boundaries of the 943 sentence utterances from the speech corpus. In the HTS training process these information will be utilized to generate synthesized speech.

Example:

ཁ་ུབཆལཆལཆལཆསལ་ཟེ་ར་དོ་་བཟུམ།

kh-a-x-0|*ch-e-l-0|ch-au-x-0|*$-$-$-$|*@-u-b-0|*ch-e-l-0|ch-au-x-0|*$-$-$-$|*z-e-r-0|d-o-x-0|*z-u-m-0|*$-$-$-$|*

Here, all the words are marked with the symbol '*' along with the phrase boundaries indicated by the symbol "$-$-$-$|*"

POS tagging of all words in each sentence utterance
Furthermore, words of each of these sentences were tagged with their respective POS tags.

Example:

མཀྣཚྭཔཔཏ་མདའ་གཉི་སཔ་ཆ་རང་བཀག་རེ་ག་ཕོ་ག་ནི།

d-a-x-0|*/NN/c-a-b-0|*/VBAt/d-a-x-0|*/TM/d-a-x-0|*/NN/ny-i-p-1|ch-a-x-0|*/NN/r-a-ng-0|*/CC/k-a-x-0|r-e-x-0|*/NN/ph-o-x-0|n-i-x-0|*/VB/$-$-$-$|*/ /

Next we needed to create label files for the new (434 sentences) speech files. For that purpose mono.dic, word.mlf and mono.mlf files are required to create label files which can then be used for training HTS. The file mono.dic is generated from the syllable dictionary. As for word.mlf file is derived from the sentence data. We generated a new word.mlf file which includes the new set of sentences (434 sentences). And mono.mlf file is generated with the help of mono.dic and word.mlf files. These files are used to generate label files.

HTS Training Process for Dzongkha TTS – Phase II

According to the previous phase of Dzongkha TTS implementation in Thailand last two years ago, this year the new version came out with much more naturalness in synthesized speech. The HTS speech engine still works as a synthesizer of the system. Therefore, HTS training process is also included in the implementation. In the new version, the naturalness is improved by adding more contexts to each phone label for creating duration trees and letting log-F0 works correctly at more appropriate context-based HMMs.

HTS training tool is divided into two main processes which are data preparation and HMM training. In data preparation, HTS training tool works on a file named *Makefile* which is located in *ProjectDir/data/Makefile*. *Makefile* contains a set of commands which prepares input files for HMM training process. For running *Makefile*, there are some required files which are:

- Wav files
- Monophone and Context-based label files
- Question file

In this phase, there are more 434 files recorded, marked and added into the training set. All of them are named as files in a HTS example project which is *CRP_SPK_XXXX*. Then names of data files in the training set are *DIT_DAW_0000.XXX*.

With more contexts, there are several features are added in to label files following a representation format which is set in the question file. Those features are POSs, word boundaries,

phrase boundaries, positions in word and positions in phrase. Then the question file from the previous phase is added the following lines.

```
QS "PositioninWord_1"                   {*_w:1_ph:*}
QS "PositioninWord_2"                   {*_w:2_ph:*}
QS "PositioninWord_3"                   {*_w:3_ph:*}
QS "PositioninPhrase_1"                 {*_ph:1}
QS "PositioninPhrase_2"                 {*_ph:2}
QS "PositioninPhrase_3"                 {*_ph:3}
QS "POS-AM"     {*_p:AM_w:*}
QS "POS-CA"     {*_p:CA_w:*}
QS "POS-CC"     {*_p:CC_w:*}
QS "POS-CDt"    {*_p:CDt_w:*}
QS "POS-CG"     {*_p:CG_w:*}
QS "POS-CV"     {*_p:CV_w:*}
QS "POS-DT"     {*_p:DT_w:*}
QS "POS-DTCG"{*_p:DTCG_w:*}
QS "POS-DTDm"{*_p:DTDm_w:*}
QS "POS-DTI"    {*_p:DTI_w:*}
QS "POS-IrM"    {*_p:IrM_w:*}
QS "POS-Irm"    {*_p:Irm_w:*}
QS "POS-JJ"     {*_p:JJ_w:*}
QS "POS-JJCG"  {*_p:JJCG_w:*}
QS "POS-JJCt"   {*_p:JJCt_w:*}
QS "POS-JJCtCG"         {*_p:JJCtCG_w:*}
QS "POS-JJP"    {*_p:JJP_w:*}
QS "POS-JJR"    {*_p:JJR_w:*}
QS "POS-JJS"    {*_p:JJS_w:*}
QS "POS-MD"     {*_p:MD_w:*}
QS "POS-N"      {*_p:N_w:*}
QS "POS-NEG"   {*_p:NEG_w:*}
QS "POS-NN"     {*_p:NN_w:*}
QS "POS-NNCG"{*_p:NNCG_w:*}
QS "POS-NNH"   {*_p:NNH_w:*}
QS "POS-NNHCG"          {*_p:NNHCG_w:*}
QS "POS-NNP"   {*_p:NNP_w:*}
QS "POS-NNPCG"          {*_p:NNPCG_w:*}
QS "POS-NNQ"   {*_p:NNQ_w:*}
QS "POS-NNQCG"          {*_p:NNQCG_w:*}
QS "POS-NNS"   {*_p:NNS_w:*}
QS "POS-NNSCG"          {*_p:NNSCG_w:*}
QS "POS-PP"     {*_p:PP_w:*}
QS "POS-PRD"   {*_p:PRD_w:*}
QS "POS-PRL"   {*_p:PRL_w:*}
QS "POS-PRLCG"          {*_p:PRLCG_w:*}
QS "POS-PRP"   {*_p:PRP_w:*}
QS "POS-PRPCG"          {*_p:PRPCG_w:*}
QS "POS-PRRF" {*_p:PRRF_w:*}
QS "POS-RB"     {*_p:RB_w:*}
QS "POS-RBB"   {*_p:RBB_w:*}
QS "POS-SC"     {*_p:SC_w:*}
QS "POS-T"      {*_p:T_w:*}
QS "POS-TI"     {*_p:TI_w:*}
QS "POS-TM"     {*_p:TM_w:*}
QS "POS-UH"     {*_p:UH_w:*}
QS "POS-VB"     {*_p:VB_w:*}
QS "POS-VBAUX"          {*_p:VBAUX_w:*}
QS "POS-VBAt"  {*_p:VBAt_w:*}
QS "POS-VBCG"{*_p:VBCG_w:*}
QS "POS-VBH"   {*_p:VBH_w:*}
QS "POS-VBI"    {*_p:VBI_w:*}
QS "POS-VBMD"{*_p:VBMD_w:*}
QS "POS-VBMDSC"         {*_p:VBMDSC_w:*}
QS "POS-VBN"   {*_p:VBN_w:*}
QS "POS-VBNa" {*_p:VBNa_w:*}
```

```
QS "POS-VBSC" {*_p:VBSC_w:*}
QS "POS-null"    {*_p:-_w:*}
```

To work corporately with the new question file, those features are represented as context-based features in label files (full label). This is an example of a full label file in the training set.

```
0 14600000 q_q-sil+d=e/A:0_p:-_w:1_ph:1
14600000 15300000 q_sil-d+e=n/A:0_p:NN_w:1_ph:1
15300000 16400000 sil_d-e+n=dz/A:0_p:NN_w:1_ph:1
16400000 17300000 d_e-n+dz=ue/A:0_p:NN_w:2_ph:1
17300000 18000000 e_n-dz+ue=n/A:0_p:NN_w:2_ph:1
18000000 18700000 n_dz-ue+n=dz/A:0_p:NN_w:3_ph:1
18700000 19800000 dz_ue-n+dz=o/A:0_p:NN_w:3_ph:1
19800000 20400000 ue_n-dz+o=m/A:0_p:VB_w:1_ph:1
20400000 21500000 n_dz-o+m=t/A:0_p:VB_w:2_ph:1
21500000 22900000 dz_o-m+t=e/A:0_p:VB_w:3_ph:1
22900000 23300000 o_m-t+e=x/A:0_p:CC_w:1_ph:1
23300000 25300000 m_t-e+x=g/A:0_p:CC_w:2_ph:2
25300000 25900000 t_e-x+g=a/A:0_p:CC_w:3_ph:2
25900000 27000000 e_x-g+a=x/A:0_p:NN_w:1_ph:2
27000000 27800000 x_g-a+x=d/A:0_p:NN_w:1_ph:2
27800000 28200000 g_a-x+d=u/A:0_p:NN_w:2_ph:2
28200000 28900000 a_x-d+u=r/A:0_p:NN_w:2_ph:2
28900000 29300000 x_d-u+r=b/A:0_p:NN_w:3_ph:2
29300000 30100000 d_u-r+b=e/A:0_p:NN_w:3_ph:2
30100000 31100000 u_r-b+e=x/A:0_p:VB_w:1_ph:2
31100000 31800000 r_b-e+x=n/A:0_p:VB_w:1_ph:2
31800000 32100000 b_e-x+n=i/A:0_p:VB_w:2_ph:3
32100000 32800000 e_x-n+i=x/A:0_p:VB_w:2_ph:3
32800000 33400000 x_n-i+x=tsh/A:0_p:VB_w:3_ph:3
33400000 33700000 n_i-x+tsh=u/A:0_p:VB_w:3_ph:3
33700000 35100000 i_x-tsh+u=x/A:0_p:NNS_w:1_ph:3
35100000 36200000 x_tsh-u+x=sil/A:0_p:NNS_w:2_ph:3
36200000 37500000 tsh_u-x+sil=q/A:0_p:NNS_w:3_ph:3
37500000 94699998 u_x-sil+q=q/A:0_p:-_w:1_ph:1
```

Before running *Makefile*, all files have to be rechecked their formats as the following check list.

- All wav files have to be recorded and converted into 16 kHz, 16 bits in mono channel.
- File names have to be in the format.
- Question file have to be prepared and placed in *ProjectDir/data/questions*
- All label files have to be created and placed in correct directories (*ProjectDir/data/labels/mono/* and *ProjectDir/data/labels/full/*)

Then *Makefile* is run by

```
$       make
```

After that all data files are prepared for training HMMs process. The next step is training HMMs by running a HMM training script file which is *ProjectDir/scripts/Trainin.pl*. This script file works corporately with a configuration file (*ProjectDir/scripts/Configs.pm*) to run HTS training commands. To control the training, the configuration file is used as a switch board and parameter setting file.

In this second phase the configuration file does not need much editing. There is only one parameter needed to be changed which is a question number parameter (in case of changed question file name for the new version).

```
# Settings ==============================
$spkr = 'daw';
$data = 'dit';
$qnum = '003';
$ver  = '1';
```

To run the script file as a background process, the following command is needed.

```
$       /usr/bin/perl scripts/Training.pl scripts/Config.pm > log 2>&1 &
```

By the way, the command is written in a global Makefile (*ProjectDir/Makefile*) which is called while running the whole process of HTS training in the previous phase.

After the training process is successfully done, HMMs and tree files which are needed for the HTS synthesizer are created in a folder (ProjectDir/Voices/). To synthesize a speech file, the synthesizer also needs a context-based phone label file of a target speech. For more information on using HTS synthesizer please follow the instruction in the previous phase document.

7. Experiments and results

The developed prototaty, a Dzongkha TTS system version 2, has been used and evaluated by native speakers of Dzongkha back in Bhutan. The detail of the evaluation and the result are evaluated below.

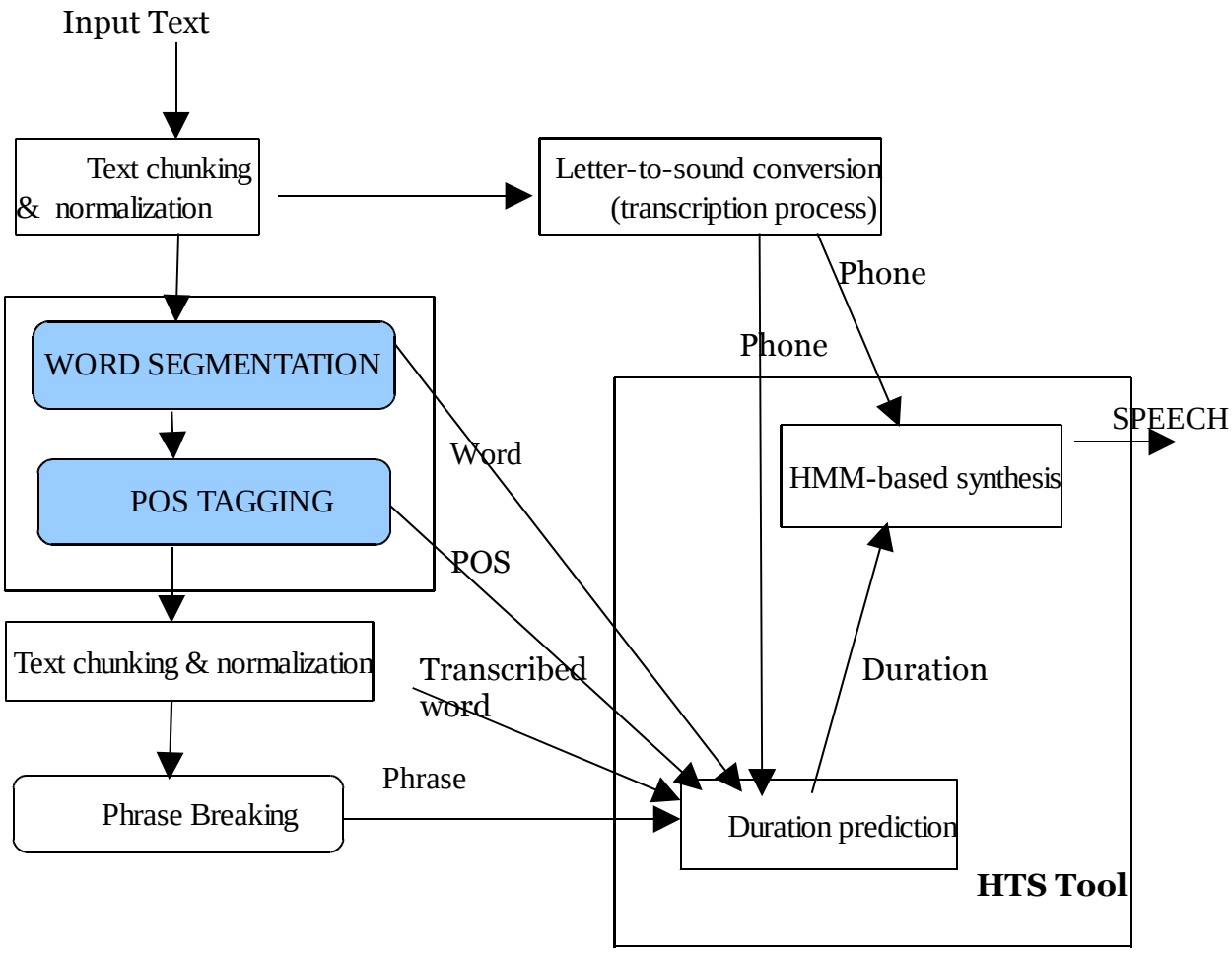


Figure 5. Dzongkha TTS Advanced Development - Execution Phase

In the executing phase all the modules that were developed in the training phase are integrated together with the HTS module in a command prompt application. So given a text input, the string of text is first segmented into words. These words are then pos-tagged with their corresponding POS tags. Numbers and dates are then normalized into letter-form for easy look-up using syllable dictionary. Phrase breaking is also done to predict phrase breaks and non-breaks.

After these steps we have an output in the format as below. This output file has word entries (from the input text). Each word has it's POS information, i.e., whether the word is a noun, a verb, an adjective etc. Also the information whether the word is a break or non-break ('B' means the word is a phrase break) is given. And finally, the transcription (letter to sound) of each word along with the tone of each word is printed out in the output file.

Word	POS	B/NB	Transcription
ཉི་ཧཱ་	NN	NB	ny-i-m-0|
ཟ་པ	NN	NB	ny-i-x-1|p-a-x-0|
། ཉི་ཧཱ་	NN	NB	ny-i-m-0|
ཟ་པ	NN	NB	ny-i-x-1|p-a-x-0|
། ལྟ་ག་ལ་ཁ་	NN	NB	t-a-x-0|l-a-x-0|kh-a-x-0|
བཀ་ཕུ་ན་	NN	B	j-a-x-0|ph-u-x-0|n-a-x-0|

This output file is then used in the labelling process along with the files that were prepared for the same process to get a label file for that particular input string. Then it is used to train the HTS engine together with question files to generate synthesized speech.

Fifteen individuals were requested to rate the new synthesized speech (syn II) along with the natural and synthesized speech (syn I) produced by the former system using Mean Opinion Score (MOS) system of evaluation. An evaluation form was prepared to allow people to rate the fifteen speech samples using a 1 to 5 rating, 1 being the worst and 5 being the best. The rating is based on the quality and naturalness of the synthesized speech After evaluation, recorded speech had an average rating of 2.41 for syn I and 2.98 for syn II. While the natural speech, understandably, had the highest average score of 4.63. The resulting MOS scores clearly shows that the new system has improved considerably comparing to the old system.

8. Conclusion

The inclusion of all modules of a TTS system has brought about a marked improvement in Dzongkha TTS system. Which is to say that the synthesized speech produced by the system sounds more natural with words having more than one syllable being pronounced together (one after another) while long sentences are uttered with some pauses in between phrases in the sentences. The MOS score for the new system confirms the improvement yet it is far from being as good as the natural speech. Future work such as increasing both the text and speech corpus, improvement of the word segmentation module, increasing the entries in the syllable dictionary will help improve the system furthermore.

9. Acknowledgement

Again the phase-II of Dzongkha TTS would not have progressed as it has without the kind helping hand extended by our friends from Human Language Technology (HLT) lab at NECTEC. Hence we are very much grateful to NECTEC, Dr. Chai Wutiwiwatchai, Dr. Ananlada Chotimongkol, Ms. Anocha Rugchatjaroen, Mr. Ausdang Thangthai and the whole team from HLT for their kind support and contribution in the making of the advanced TTS system. Without their help and support the project wouldn't have happened. We are forever grateful and thankful to them.

10. References

Chungku, Faaß, G., Rabgay, J., 2010. NLP resources for Dzongkha: Report on the design of a tagset, building a 570,247 tokens corpus, and annotating it with parts of speech. Technical Report.

Hansakunbuntheung, C., Thangthai, A., Wutiwiwatchai, C., Siricharoenchai, R., 2005. Learning methods and features for corpus-based phrase break prediction on Thai. In: *European Conference on Speech Communication and Technology (EUROSPEECH)*, pp. 1969-1972.

Haruechaiyasak, C., Kongyoung, S., and Dailey, M.N., 2008. A Comparative Study on Thai Word Segmentation Approaches. In *Proceedings of ECTI-CON.* Krabi, Thailand.

Schmid, H., 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*. Manchester, UK, pages 44 – 49.

Sherpa, U., Pemo, D., Chhoeden, D., Rugchatjaroen, A., Thangthai, A., Wutiwiwatchai, C., 2008. Pioneering Dzongkha text-to-speech synthesis. In: *Proc. Oriental COCOSDA*. pp. 150–154.

Wutiwiwatchai, C., Rugchatjaroen, A. and Saychum, S., 2007. An Intensive Design of a Thai Speech Synthesis Corpus. *The Seventh International Symposium on Natural Language Processing*. Thailand.