# Customizing Grapheme-to-Phoneme System for Non-Trivial Transcription Problems in Bangla Language

**Sudipta Saha Shubha**[1], **Nafis Sadeq**[1], **Shafayat Ahmed**[1], **Md. Nahidul Islam**[1], **Muhammad Abdullah Adnan**[1], **Md. Yasin Ali Khan**[2], and **Mohammad Zuberul Islam**[2]

[1]Bangladesh University of Engineering and Technology (BUET)
[2]Samsung R&D Institute, Bangladesh
{sudipta,nafis,shafayat,nahid.rimon}@ra.cse.buet.ac.bd
adnan@cse.buet.ac.bd,{yasin.ali,m.zuberul}@samsung.com

## Abstract

Grapheme to phoneme (G2P) conversion is an integral part in various text and speech processing systems, such as: Text to Speech system, Speech Recognition system, etc. The existing methodologies for G2P conversion in Bangla language are mostly rule-based. However, data-driven approaches have proved their superiority over rule-based approaches for large-scale G2P conversion in other languages, such as: English, German, etc. As the performance of data-driven approaches for G2P conversion depend largely on pronunciation lexicon on which the system is trained, in this paper, we investigate on developing an improved training lexicon by identifying and categorizing the critical cases in Bangla language and include those critical cases in training lexicon for developing a robust G2P conversion system in Bangla language. Additionally, we have incorporated nasal vowels in our proposed phoneme list. Our methodology outperforms other state-of-the-art approaches for G2P conversion in Bangla language.

## 1 Introduction

Grapheme to phoneme (G2P) conversion provides a mapping between a word and its pronunciation. Such mapping provides opportunity for a non-native person to learn the correct pronunciation of words of a foreign language. Moreover, in modern Text to Speech (TTS) and Automatic Speech Recognition (ASR) systems, G2P conversion is an integral task.

The task of G2P conversion is generally language specific due to language specific conventions, rules, pronunciation constraints, etc. In this paper, we focus on Modern Standard Bangla. An example of G2P conversion in Bangla language: phonetic transcription of অনুশীলন (practice) is /o n u sh i l O n/. (Please refer to Table 1 for our phoneme symbols.)

The simplest means of G2P conversion is to build up a lexicon or dictionary containing the mapping from words to their corresponding pronunciations. However, it fails to provide pronunciations for unknown words and inclusion of newer words increases memory requirement. In another approach (Mosaddeque et al., 2006), there are predefined rules for the conversion of a word to its pronunciation. Though such rule-based approach can work for any word, the system becomes complex when it tries to formulate rules for incorporating all irregularities of pronunciation in a language.

Clearly, these approaches are not feasible for large-scale G2P conversion which is necessary in any modern TTS or ASR system. Data-driven machine learning approaches have great potential in such large-scale G2P conversion (Rao et al., 2015). In such an approach, a machine learning model predicts the phoneme conversion of a grapheme, being trained on a lexicon. A predominant work following such approach in Bangla language is by Google (Gutkin et al., 2016), where they train their system using $37K$ words and achieve word-level accuracy of $81.5\%$. However, a system trained on their lexicon will face several shortcomings, such as: কাদা(mud) and কাঁদা(to cry) are pronounced differently but will have same phoneme representation in their system as: /k a d a/. Similarly, পরী(fairy) and পড়ি(to read) are pronounced differently but will have same phoneme representation in their system as: /p o r i/. Moreover, G2P system trained on their lexicon performs poorly on our identified critical cases from the most frequent $100K$ words (Table 3).

Being motivated to increase the accuracy of grapheme to phoneme conversion in Bangla language, which will also perform well for critical inputs, we have developed a customized and robust G2P system for Bangla language.



*Proceedings of NAACL-HLT 2019*, pages 3191–3200
Minneapolis, Minnesota, June 2 - June 7, 2019. 

Our major contributions are as follows:

(i) We identify and categorize the critical cases for grapheme to phoneme (G2P) conversion in Bangla language by analyzing the most frequent $100K$ words.

(ii) We enrich the training lexicon for developing a robust G2P conversion system in Bangla language that performs much better for critical cases compared to other state-of-the-art G2P systems.

(iii) We perform phonetic transcriptions considering nasal vowels as separate phonemes.

(iv) We perform extensive simulations on large-scale dataset and show that our methodology outperforms other state-of-the-art approaches for G2P conversion in Bangla language by providing word-level accuracy of $90.2\%$.

The rest of the paper is organized as follows: we discuss the previous works in Section 2, our phoneme list in Section 3, identification of critical cases and categorization of errors in Section 4, development of our system in Section 5, experimental results in Section 6, and conclusion and future works in Section 7.

## 2 Previous Works

The research works for G2P in English are quite extensive. Chen (2003) investigate machine learning based systems for G2P in English. They experiment with joint maximum entropy n-gram model, conditional maximum entropy model, etc. Yao and Zweig (2015) utilize bi-directional LSTM (Long Short Term Memory) recurrent neural network for G2P and achieve $5.45\%$ PER on CMU dictionary (air, 2015). Thu et al. (2016) show comparisons among various machine learning algorithms for G2P in Burmese language. Joint sequence n-gram models aim to discover joint vocabulary consisting of graphemes and phonemes through the alignment of graphemes and phonemes. Bisani and Ney (2008) develop a joint-sequence model for G2P. Novak (2012), Novak et al. (2013) are other prominent works working on this model. Neural sequence to sequence models are popular for G2P conversion. Some prominent works on such models are: Caruana (1997), Jiampojamarn et al. (2007), Sutskever et al. (2014), Yao and Zweig (2015), Jiampojamarn et al. (2007), Rao et al. (2015), Yao and Zweig (2015), Schnober et al. (2016), Tsvetkov et al. (2016), He et al. (2016), Wu et al. (2016), Johnson et al. (2016), Toshniwal and Livescu (2016), and Vaswani et al. (2017).

Again, another line of research deals with G2P conversion for more than one language. Such works include: Mana et al. (2001), Kim and Snyder (2012), Deri and Knight (2016), and Milde et al. (2017).

Most of the works related to G2P conversion that are focused on Bangla language, follow rule-based approach. Rule-based approach of Mosaddeque et al. (2006) provides accuracy of $97.01\%$ on a previously seen corpus containing 736 words, but the system's accuracy is $81.95\%$ on an previously unobserved corpus containing 8399 words. This work was extended by Alam et al. (2011) describing 3880 rules with an accuracy of $89.48\%$ on another corpus. Basu et al. (2009) discuss a rule-based approach considering several information: parts-of-speech, subsequent context, etc. Their work describes only 21 rules and provides an accuracy of $91.48\%$ on a corpus of 9294 words. Ghosh et al. (2010) provide a heuristic for G2P that takes into account parts-of-speech, orthographic, and contextual information. Their work provides $70\%$ accuracy on a corpus containing of 755 words. A prominent work for data driven G2P in Bangla language is by Google (Gutkin et al. (2016)). They develop a lexicon and achieve word-level accuracy of $81.5\%$. Chowdhury et al. (2017) use conditional random field for G2P in Bangla. They report $14.88\%$ phoneme error rate on Google lexicon.

## 3 Phoneme List

Our Phoneme symbols are provided in Table 1. This table is a good reference for the 47 phoneme symbols that we have followed in this paper and their corresponding International Phonetic Alphabet (IPA) symbols. Throughout the paper, we use these 47 phoneme symbols, not the IPA symbols. There is disagreement between linguists whether nasal vowels should be considered as separate phonemes (Barman, 2009). We added nasal vowels in our phoneme list to differentiate between a word with its nasalized counterpart, such as the word কাঁদা(to cry) and কাদা(mud). Here, /a/, /e/, /u/, /i/, /o/, /O/, /E/, /an/, /en/, /un/, /in/, /on/, /On/, /En/ are normal vowels, /ew/, /ow/, /uw/, /iw/ are weak vowels, and the rest are consonants.

| Phoneme | IPA | Phoneme | IPA | Phoneme | IPA | Phoneme | IPA |
|---|---|---|---|---|---|---|---|
| i | i | On | ɔ̃ | D | ɖ | m | m |
| u | u | an | ã | Dh | ɖʱ | r | r |
| e | e | k | k | t | t | R | ɾ |
| o | o | kh | kʰ | th | tʰ | l | l |
| E | ɛ | g | g | d | d | h | ɦ |
| O | ɔ | gh | gʱ | dh | dʱ | s | s |
| a | a | c | ʧ | p | p | sh | ʃ |
| in | ĩ | ch | ʧʰ | ph | pʰ | iw | i̯ |
| un | ũ | j | ʤ | b | b | ew | e̯ |
| en | ẽ | jh | ʤʱ | bh | bʱ | ow | o̯ |
| on | õ | T | ʈ | N | ŋ | uw | u̯ |
| En | æ̃ | Th | ʈʰ | n | n | | |

Table 1: Our Phoneme Symbols with Their Corresponding IPA Symbols

## 4 Identification and Categorization of Non-Trivial Cases for Transcription

We envision of developing a robust G2P system that will perform reasonably well on any word in Bangla language. A G2P system that performs well on the most frequent words, should also do well on other words. With this motivation, we focus on increasing accuracy on the most frequent words. Especially, we are concerned about those words that are among the most frequent words but non-trivial or critical for phonetic transcription, i.e., current state-of-the-art methodologies perform poorly on these critical words. We investigate on identifying and categorizing such non-trivial or critical cases so that future research works can give special focus on developing methods for improving phonetic transcriptions of these critical words.

### 4.1 Identifying the Most Frequent Words

To get a hold of contemporary usage of Bangla language, we do extensive crawling. We crawled 42 websites of various Bangla newspapers, blogs, e-book libraries, wikipedia, etc. covering various domains such as: politics, economics, sports, drama, novel, story, education, entertainment, general knowledge, history, etc. After data cleaning and data normalization, we had about $10M$ sentences. We counted how many times each of the unique words appeared in those sentences. We then consider the most frequent $100K$ words and aim to identify the critical cases for phonetic transcription among these most frequent words.

### 4.2 Identifying the Critical Cases for Transcription

After changing the Google lexicon (of size $60K$ (around)) according to our phoneme symbols (Table 1), we prepare 4 versions of Google's lexicon of size $12K$, $24K$, $40K$, and $60K$ respectively for identifying the critical cases for phonetic transcription. Algorithm 1 shows prefix comparing algorithm that we use for compressing a phonetic lexicon or dictionary of grapheme sequence to phoneme sequence. The algorithm matches the prefix of consecutive words (grapheme sequence) of a sorted dictionary (sorted according to ascending order of grapheme sequence of a word) and keeps a word (with its corresponding phoneme sequence) only if it does not share its prefix with any other words. We run the algorithm successively 3 times, i.e., we use the destination dictionary of one iteration as the source dictionary of next iteration. Each iteration produces a minimized version of the basic lexicon (Google lexicon). After 3 iterations, the dictionary does not get any more compressed. We find the phonetic transcriptions of each of the $100K$ most frequent words using models trained on each of the 4 versions of Google's lexicon (basic + 3 minimized). So, from 4 models (each model trained on a version of the basic Google lexicon), we get 4 sets of transcriptions for the most frequent $100K$ words. For most of the words (around $70K$ words), we observe that the phonetic transcriptions are exactly same in each of the 4 set. However, for the remaining $30K$ words (29105 words to be exact), we observe that at least one set provides different transcription. We take these $30K$ words to be the critical cases. Our intuition is that if two G2P systems: one trained on a smaller version of the basic lexicon, and another trained on a larger version of the basic lexicon provide the same transcription for a word, then the word is a trivial case for phonetic transcription. We then manually verify the phonetic transcriptions of these $30K$ words taking help from 3 linguists and following Chowdhury (2016), and consider these $30K$ words as critical cases for phonetic transcription.

### 4.3 Categorizing the Critical Cases

We categorize the critical cases into 7 categories and observe the distribution of the critical transcriptions into these 7 categories. These 7 cate-

**Algorithm 1** Algorithm for Compressing a Dictionary or Lexicon

```
sd ← sorted sourceDictionary
dd ← sorted destinationDictionary
a.grs ← grapheme sequence of lexicon
                entry a
add sd[0] to dd
i = 1
while i ≠ length(sd) do
    pw = sd[i − 1]
    cw = sd[i]
    if length(pw.grs) ≥ 3 & pw.grs is
  prefix of cw.grs then
        continue
    else
        add cw to dd
    i ← i + 1
```

gories capture most of the errors. The categories are:

- **Open Close Vowel Confusion:** G2P system provides pronunciation as close vowel that should be pronounced as open vowel ideally, and vice-versa. For example, correct phoneme of ব্যাঙ (frog) is /b E n g/, but if G2P system provides output /b e n g/, then it is an error under this category as in the place of open vowel (here, /E/), G2P system is giving close vowel (here, /e/).

- **Inherent Vowel Confusion:** G2P system does not provide inherent vowel as output where there should be an inherent vowel ideally. For example, correct phoneme of সকাল (morning) is /sh O k a l/, but if G2P system provides output /sh k a l/, then it is an error under this category as the output of G2P does not give inherent vowel (here, /O/).

- **Diphthong Confusion:** G2P system does not provide falling diphthong in output where there should be a falling diphthong ideally. Or, system does not provide rising diphthong in output where there should be a rising diphthong ideally. For example, correct phoneme of সই (friend) is /sh o iw/, but if G2P system provides output /sh o i/, then it is an error under this category as the output of G2P does not capture the falling diphthong (here, /o iw/).

- **s or sh Confusion:** G2P system provides /s/ in phonetic transcription, where there should be /sh/, and vice-versa. For example, correct phoneme sequence of সংগঠন (organization) is /sh O N g O Th o n/, but if G2P system provides output /s O N g O Th o n/, then it is an error under this category as the output of G2P gives /s/ in place of /sh/.

- **s or ch Confusion:** G2P system provides /s/ in phonetic transcription, where there should be /ch/, and vice-versa. For example, correct phoneme sequence of ছাতা (umbrella) is /ch a t a/, but if G2P system provides output /s a t a/, then it is an error under this category as the output of G2P gives /s/ in place of /ch/.

- **Nasal Confusion:** G2P system does not provide any nasal vowel where there should be a nasal vowel, and vice-versa. For example, correct phoneme sequence of চাঁদ (moon) is /c an d/, but if G2P system provides output /c a d/, then it is an error under this category as the output of G2P gives /a/ in place of /an/.

- **Other Vowel Confusion:** G2P system provides completely different vowel than the corresponding vowel that should ideally be in that position of the phoneme sequence. Note that, in the other error categories, for each position in the phoneme sequence, the generated and ideal phonemes were somehow related. But in this category, at a specific position of the phoneme sequence, the generated and ideal phonemes are completely different. For example, correct phoneme sequence of অধ্যবসায় (perseverance) is /o d dh o b O sh a ew/, but if G2P system provides output /o d dh a b O sh a ew/, then it is an error under this category as the output of G2P gives /a/ in place of /o/ (fourth phoneme).

Algorithm 2 compares a machine-generated lexicon with a reference lexicon (manually verified), where both the lexicons have same grapheme sequences, but the corresponding phoneme sequences may be different. This algorithm counts how many errors of each category are there in the machine generated lexicon. The algorithm takes each entry of the generated lexicon and increases the count of the corresponding error category (if an error is present there).

We train attention mechanism based Transformer model on each of the 4 lexicons and get

**Algorithm 2** Comparing a Generated Lexicon ($gl$) with Reference Lexicon ($rl$)

$N \leftarrow$ total number of entries in each lexicon
$A, B, C, D, E, F, G$ are Open Close Vowel, s or sh, s or ch, Nasal, Dipthong, Other Vowel, and Inherent confusions, respectively, all initially zero
$H$ denotes other errors not captured by the 7 categories, initially zero
vl and wl are lists of vowels and weak vowels respectively
$a.phs \leftarrow$ phoneme sequence of lexicon entry $a$
$i \leftarrow 0$
**while** $i \neq N$ **do**
    $g = gl[i].phs$
    $r = rl[i].phs$
    $M = \min(length(g), length(r))$
    $j \leftarrow 0$
    **while** $j \neq M$ **do**
        $x = g[j]$
        $y = r[j]$
        **if** $x = y$ **then**
            continue
        $(x, y) \leftarrow sorted(x, y)$
        $Total\_error = Total\_error + 1$
        **if** $ocConfusion(x, y)$ **then**
            $A \leftarrow A + 1$
        **else if** $(x, y) = (\text{“}s\text{”}, \text{“}sh\text{”})$ **then**
            $B \leftarrow B + 1$
        **else if** $(x, y) = (\text{“}ch\text{”}, \text{“}s\text{”})$ **then**
            $C \leftarrow C + 1$
        **else if** $x + \text{“}n\text{”} = y$ **then**
            $D \leftarrow D + 1$
        **else if** $x$ in wl or $y$ in wl **then**
            $E \leftarrow E + 1$
        **else if** $x$ in vl and $y$ in vl **then**
            $F \leftarrow F + 1$
        **else if** $removeVowel(g) = removeVowel(r)$ **then**
            $G \leftarrow G + 1$
        **else**
            $H \leftarrow H + 1$
        $j \leftarrow j + 1$
    $i \leftarrow i + 1$

4 G2P models. We find the phoneme representation of $30K$ critical cases using each of the 4 G2P models. Using Algorithm 2, we count the errors of each category for each of the 4 models. We report

**Algorithm 3** Procedure: ocConfusion $(x, y)$ (Checks if open close vowel confusion)

$ocSet \leftarrow [(\text{“}O\text{”}, \text{“}o\text{”}), (\text{“}E\text{”}, \text{“}e\text{”}), (\text{“}On\text{”}, \text{“}on\text{”}), (\text{“}En\text{”}, \text{“}en\text{”})]$
**if** $(x, y)$ in $ocSet$ **then**
    **return** True
**else**
    **return** False

**Algorithm 4** Procedure: removeVowel (phoneme_sequence)

**return** phoneme_sequence removing all vowels from it

the results in Table 2 and Figure 1. Here, other error denotes the errors that are not captured by these 7 categories. We see from these results that most of the errors are under Open Close vowel, s or sh, Diphthong, and Inherent Vowel confusions.

## 5 Developing an Improved G2P System for Bangla Language

We develop an improved lexicon and use two machine learning based models trained on our lexicon to develop an improved G2P system for Bangla language.

### 5.1 Developing an Improved Lexicon

For developing an improved lexicon, we include with Google's lexicon the manually verified $30K$ non-trivial or critical entries. Also, we include the $70K$ entries in which all of the 4 models (trained on each of the 4 versions of Google lexicon) unanimously agreed. After removing the repeated entries with same grapheme sequence, our lexicon consists of around $90K$ entries. In case of repeated entries having same grapheme sequence, we carefully keep only the entry that has been manually verified.

### 5.2 G2P Models

We use Neural Sequence to Sequence models. In these models, a conditional distribution of a sequence (here, phoneme sequence) is learned conditioned on another sequence (here, grapheme sequence). We train two following Sequence to Sequence models on our lexicon for G2P conversion:

**LSTM-RNN**: This is a plain Sequence to Sequence model that incorporates an encoder and decoder mechanism. Recurrent Neural Network

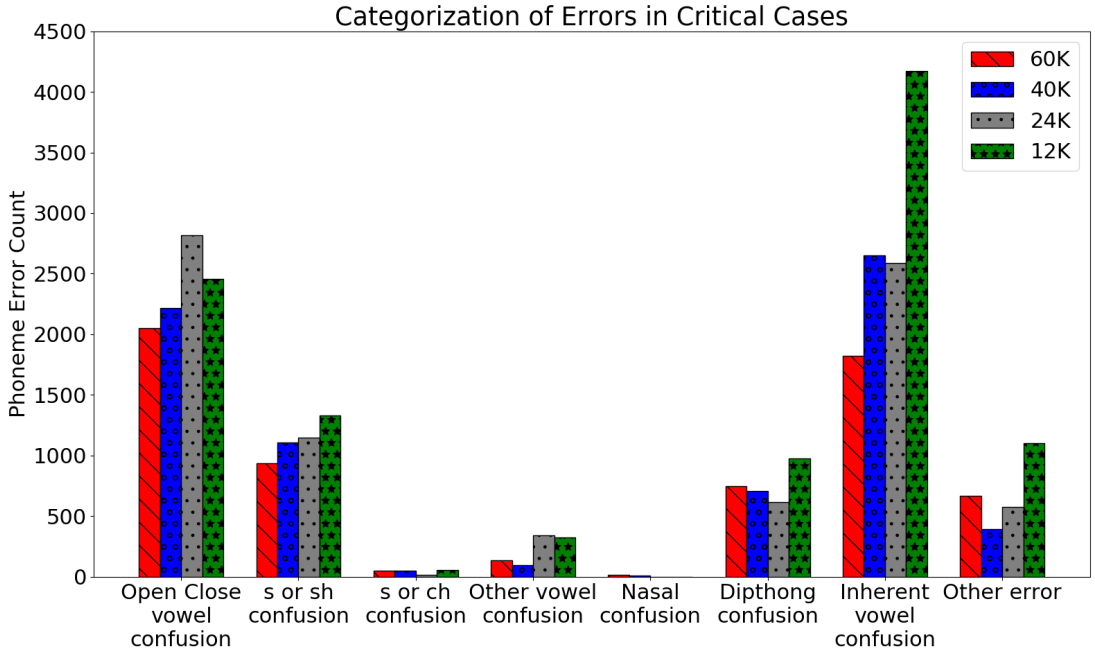


Figure 1: Categorization of errors in critical cases, here each of the $60K, 40K, 24K$, and $12K$ denotes the model trained on that particular lexicon.

(RNN) is usually utilized in encoder and decoder design. For addressing vanishing gradient problem in RNN, Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is used. We follow Yao and Zweig (2015) for implementation.

**Transformer Model**: Transformer Model uses attention mechanism. Attention mechanism provides improvement upon plain Sequence to Sequence by easing the flow of information from source sequence to destination sequence. We follow Vaswani et al. (2017) for implementation.

We show the performance of both of these models in Section 6. We observe that Transformer Model provides higher token-level accuracy (lower Word Error Rate) than LSTM-RNN.

## 6 Experimental Results

We run extensive simulations and use two measures for evaluating the performances of the systems:

**Word Error Rate (WER):** For calculating Word Error Rate (WER), we use the following formula:

$$\text{WER} = \frac{E}{T}$$

where $E$ denotes the number of words that have disagreement in their generated phoneme sequence and reference phoneme sequence, and $T$ denotes the total number of words.

**Phoneme Error Rate (PER):** For calculating Phoneme Error Rate (PER), we use the following formula:

$$\text{PER} = \frac{I + S + D}{T}$$

where $I$, $S$, $D$ denote respectively the total number of insertion, substitution, and deletion operations needed for all the words to align the generated phoneme sequence with the reference phoneme sequence for each word. $T$ denotes the total number of phonemes present in all the words.

Our best performing model is Transformer Model. We use batch size of 4096. Our neural network has 3 hidden layers, each containing 256 nodes. We use a computer having 8GB RAM, Intel Core i7 CPU, and Nvidia Geforce 1050 GPU for running all of the simulations. For each model, we run the simulations for around $110K$ iterations taking around $5$ hours.

### 6.1 Performance on Critical Cases

We report the experiment results of Google's lexicon on critical cases in Table 3. We do not report our lexicon here as critical cases are already included in our lexicon.

### 6.2 Performance Comparison In General

For comparing the performances of models trained on our lexicon and Google's lexicon, we randomly

| Error Type | 60K | 40K | 24K | 12K |
|---|---|---|---|---|
| total error | 6415 | 7222 | 8087 | 10400 |
| Open Close Confusion (%) | 32.0 | 30.7 | 34.8 | 23.6 |
| Inherent Vowel Confusion (%) | 28.4 | 36.7 | 32.0 | 40.1 |
| s or sh confusion (%) | 14.6 | 15.3 | 14.2 | 12.8 |
| Dipthong confusion (%) | 11.6 | 9.8 | 7.6 | 9.4 |
| Other Vowel Confusion (%) | 2.1 | 1.3 | 4.2 | 3.1 |
| s or ch confusion (%) | 0.8 | 0.7 | 0.2 | 0.5 |
| Nasal Confusion (%) | 0.2 | 0.1 | 0 | 0 |
| Other Error (%) | 10.4 | 5.4 | 7.1 | 10.6 |

Table 2: Error classification of $30K$ critical cases, here each of the four rightmost columns denotes the model trained on that particular lexicon.

| Lexicon | Model | WER (%) | PER (%) |
|---|---|---|---|
| Google | LSTM-RNN | 25.7 | 3.26 |
| | Transformer Model | 23.6 | 2.71 |

Table 3: Performance on Critical Cases

take 9000 entries from our manually verified $30K$ critical cases as test set. We use this test set for evaluating all the models. Though our actual lexicon contains these 9000 entries, we do not keep them in our lexicon while doing the experiments to fairly evaluate the performances of the lexicons. For both lexicons, we keep 90% of the lexicon in train set and remaining 10% in validation set. Table 4 shows the result. Models trained on our lexicon outperforms those trained on Google's lexicon by a significant margin. Moreover, Transformer Model performs better than LSTM-RNN.

Figure 2 and Table 5 categorize the errors of systems trained on 3 types of lexicons (Romanized version of our lexicon is discussed in section 6.3) by using Algorithm 2. Here, we report the results of Transformer Model only as it has been better performing that LSTM-RNN in our experiments. We observe most of the errors are related to Open Close vowel, s or sh, Diphthong, and Inherent Vowel confusions - this finding also conforms to Figure 1 and Table 2, which were error categorization of critical cases.

| Lexicon | Model | WER (%) | PER (%) |
|---|---|---|---|
| Google | LSTM-RNN | 17.1 | 2.32 |
| | Transformer Model | 14.8 | 1.88 |
| Our Lexicon | LSTM-RNN | 10.5 | 1.42 |
| | Transformer Model | 9.8 | 1.33 |

Table 4: Performance Comparison In General

| Error Type | Our Lexicon | Romanized Lexicon | Google Lexicon |
|---|---|---|---|
| Total Error | 1337 | 1406 | 2961 |
| Inherent Vowel Confusion (%) | 35.6 | 34.5 | 33.8 |
| Open Close Confusion (%) | 30.1 | 30.4 | 27.9 |
| s or sh confusion (%) | 13.3 | 13.0 | 12.1 |
| Diphthong confusion (%) | 10.7 | 9.8 | 13.1 |
| Other Vowel Confusion (%) | 1.9 | 4.3 | 2.7 |
| s or ch confusion (%) | 0.3 | 0.2 | 0.2 |
| Nasal Confusion (%) | 0.2 | 0.43 | 0.6 |
| Other Error (%) | 7.9 | 7.4 | 9.7 |

Table 5: Performance comparison on different error categories. Here each of the three rightmost columns denotes the model trained on that particular lexicon.

### 6.3 Effect of Romanization

For doing experiments on the effect of romanization on G2P conversion, we romanized all grapheme sequences in our lexicon to prepare romanized counterpart of our lexicon. During romanization, each grapheme symbol in Bangla is replaced with a single English letter except that if a consonant grapheme is not followed by a vowel grapheme, "O" was added after the romanized symbol of that consonant as the roman symbol for "অ", which is usually inherently pronounced in such cases. All the symbols used for romanization were completely disjoint to avoid any ambiguity in the lexicon. Figure 3 shows the WER and PER

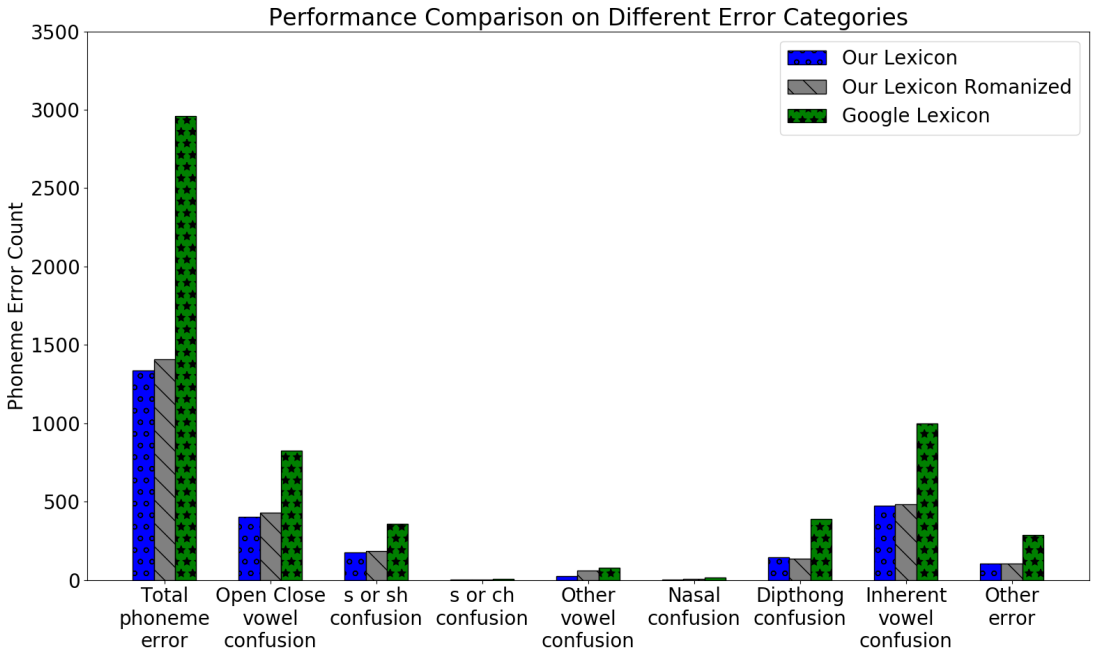


Figure 2: Performance Comparison on Different Error Categories

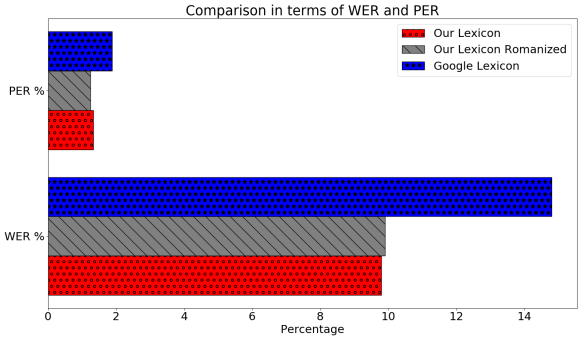


Figure 3: Comparison in terms of WER and PER

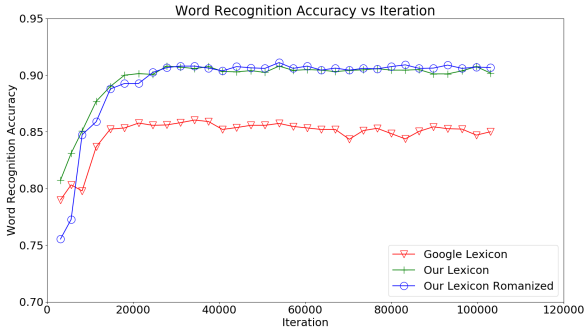


Figure 4: Word Recognition Accuracy vs Iteration

of systems trained on 3 types of lexicons. Here, we report the results of Transformer Model only as it has been better performing than LSTM-RNN in our experiments. Both versions of our lexicon perform better (lower WER and lower PER) than Google's lexicon. Also romanization does not significantly increase or decrease the performance.

Figures 4 and 5 show respectively the Word Recognition Accuracy (1−WER) and Phoneme Recognition Accuracy (1−PER) with respect to number of iterations run during simulation. Both versions of our lexicon perform better (higher Word Recognition Accuracy and higher Phoneme Recognition Accuracy) than Google's lexicon. Figure 6 shows Negative Log Perplexity vs number of iterations. Both versions of our lexicon provide higher negative log perplexity than Google's lexicon.

### 6.4 Effectiveness of Our Identified Critical Cases

In this section, we want to establish that the improved performance of our lexicon comes not only from the increase in number of training samples, but also due to the fact that the critical cases identified by our novel methodology have been added as training samples. For this, we prepare a new training lexicon by combining a portion of the Google lexicon with a portion of our identified critical cases. As we have kept $9K$ entries from the critical cases as our test set, we take the remaining $21K$ critical cases and combine them with the randomly taken $39K$ entries from Google lexicon to prepare a new lexicon of size $60K$. While taking entries from Google lexicon, we ensure that we do not take any repeated entry that has already been in the critical cases and added to the new lexicon.

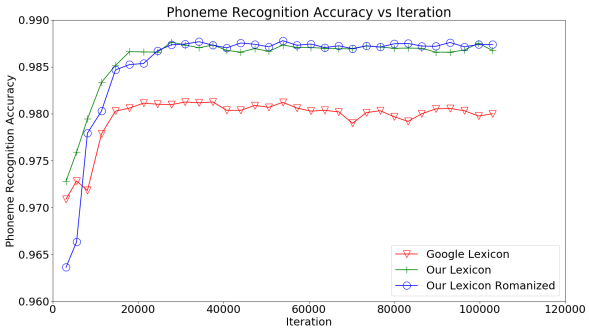


Figure 5: Phoneme Recognition Accuracy vs Iteration

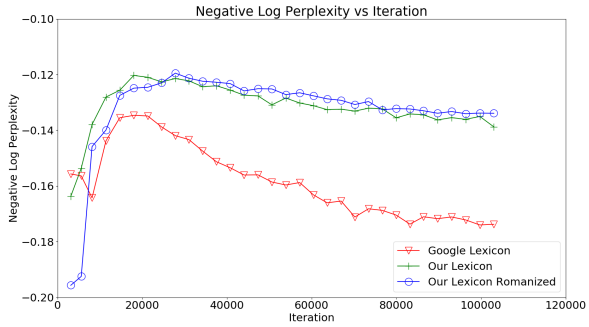


Figure 6: Negative Log Perplexity vs Iteration

| Lexicon | Model | WER (%) | PER (%) |
|---|---|---|---|
| Google | LSTM-RNN | 17.1 | 2.32 |
| | Transformer Model | 14.8 | 1.88 |
| New Lexicon | LSTM-RNN | 12.6 | 1.54 |
| | Transformer Model | 11.2 | 1.49 |

Table 6: Effectiveness of critical cases. Both lexicons are of size $60K$. New Lexicon consists of $21K$ critical cases and $39K$ entries from Google lexicon.

We then compare the performance of this new lexicon with the Google lexicon, both of which are of same size ($60K$), on our test set. The results are in Table 6. The results clearly show that even in the case of same sized lexicons, our identified critical cases can significantly improve the performance as evidenced by the lower WER and lower PER than those for the Google lexicon.

## 7 Conclusion and Future Works

In this paper, we have identified the critical cases, categorized the errors, and increased the current state-of-the-art accuracy in G2P conversion for Bangla language by developing an improved lexicon. In future, we will do classification of errors using unsupervised clustering approaches. Tables 2 and 5 both show that most of the errors occur in Open Close Vowel, s or sh, Diphthong, and Inherent Vowel confusion categories. In future, we will focus on each category to devise novel methodology for mitigating the errors of that category and in turn further increase the accuracy of G2P system in Bangla.

## Acknowledgements

This research work is conducted at the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET) and is supported by Samsung Research.

## References

air. 2015. https://svn.code.sf.net/p/cmusphinx/code/branches/cmudict/cmudict-0.7b.

Firoj Alam, SM Murtoza Habib, and Mumit Khan. 2011. Bangla text to speech using festival. In *Conference on Human Language Technology for Development*, pages 154–161.

Binoy Barman. 2009. A contrastive analysis of english and bangla phonemics. *Dhaka University Journal of Linguistics*, 2(4):19–42.

Joyanta Basu, Tulika Basu, Mridusmita Mitra, and Shyamal Kr Das Mandal. 2009. Grapheme to phoneme (g2p) conversion for bangla. In *Speech Database and Assessments, 2009 Oriental COCOSDA International Conference on*, pages 66–71. IEEE.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Stanley F Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Eighth European Conference on Speech Communication and Technology*.

Jamil Chowdhury, editor. 2016. *Adhunik Bangla Ovidhan*. Bangla Academy, Bangla Academy, Dhaka - 1000.

Shammur Absar Chowdhury, Firoj Alam, Naira Khan, and Sheak RH Noori. 2017. Bangla grapheme to phoneme conversion using conditional random fields. In *Computer and Information Technology (ICCIT), 2017 20th International Conference of*, pages 1–6. IEEE.

Aliya Deri and Kevin Knight. 2016. Grapheme-to-phoneme models for (almost) any language. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 399–408.

Krishnendu Ghosh, Ramu V Reddy, NP Narendra, S Maity, SG Koolagudi, and KS Rao. 2010. Grapheme to phoneme conversion in bengali for festival based tts framework. In *8th international conference on natural language processing (ICON)*. Macmillan Publishers.

Alexander Gutkin, Linne Ha, Martin Jansche, Knot Pipatsrisawat, and Richard Sproat. 2016. Tts for low resource languages: A bangla synthesizer. In *LREC*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Young-Bum Kim and Benjamin Snyder. 2012. Universal grapheme-to-phoneme prediction over latin alphabets. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 332–343. Association for Computational Linguistics.

Franco Mana, Paolo Massimino, and Alberto Pacchiotti. 2001. Using machine learning techniques for grapheme to phoneme transcription. In *Seventh European Conference on Speech Communication and Technology*.

Benjamin Milde, Christoph Schmidt, and Joachim Köhler. 2017. Multitask sequence-to-sequence models for grapheme-to-phoneme conversion. *Proc. Interspeech 2017*, pages 2536–2540.

Ayesha Binte Mosaddeque, Naushad UzZaman, and Mumit Khan. 2006. Rule based automated pronunciation generator.

J. R. Novak. 2012. https://github.com/AdolfVonKleist/Phonetisaurus.

Josef R Novak, Nobuaki Minematsu, and Keikichi Hirose. 2013. Failure transitions for joint n-gram models and g2p conversion. In *INTERSPEECH*, pages 1821–1825.

Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4225–4229. IEEE.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. *arXiv preprint arXiv:1610.07796*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ye Kyaw Thu, Win Pa Pa, Yoshinori Sagisaka, and Naoto Iwahashi. 2016. Comparison of grapheme-to-phoneme conversion methods on a myanmar pronunciation dictionary. In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 11–22.

Shubham Toshniwal and Karen Livescu. 2016. Jointly learning to align and convert graphemes to phonemes with neural attention models. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 76–82. IEEE.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. *arXiv preprint arXiv:1605.03832*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.