An encoder-decoder based grapheme-to-phoneme converter for Bangla speech synthesis

Arif Ahmad*, Mohammad Reza Selim, Muhammed Zafar Iqbal and Mohammad Shahidur Rahman

Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet-3114, Bangladesh

(Received 28 June 2019, Accepted for publication 29 July 2019)

Abstract: This paper proposes an encoder-decoder based sequence-to-sequence model for Grapheme-to-Phoneme (G2P) conversion in Bangla (Exonym: Bengali). G2P models are key components in speech recognition and speech synthesis systems as they describe how words are pronounced. Traditional, rule-based models do not perform well in unseen contexts. We propose to adopt a neural machine translation (NMT) model to solve the G2P problem. We used gated recurrent units (GRU) recurrent neural network (RNN) to build our model. In contrast to joint-sequence based G2P models, our encoder-decoder based model has the flexibility of not requiring explicit grapheme-to-phoneme alignment which are not straightforward to perform. We trained our model on a pronunciation dictionary of (approximately) 135,000 entries and obtained a word error rate (WER) of 12.49% which is a significant improvement from the existing rule-based and machine-learning based Bangla G2P models.

Keywords: Encoder-decoder, Sequence-to-sequence, GRU-RNN, NMT

PACS number: 43.72.Ja [doi:10.1250/ast.40.374]

1. INTRODUCTION

Grapheme-to-phoneme (G2P) converters are essential components in many speech processing tasks, such as Text-to-Speech Synthesis (TTS) and Automatic Speech Recognition (ASR). In many languages, extracting a phoneme sequence from the orthographic form is not always a trivial task. If there exists a direct mapping between the orthographic representation and the pronunciation, then this language is called a phonemic language. Bangla, an Indo-Aryan language, is not completely a phonetic language. Although most of the Bangla written texts can be pronounced directly without ambiguity, the problem arises for certain letters (e.g. /s/ vs. /ʃ/ for the letter ʃ), and for the pronunciation of implicit vowels (/null/ vs. /o/ vs. /a/). Besides, some conjuncts have ambiguous pronunciations too.

To solve the G2P problem, the oldest approach was to use a digitized pronunciation dictionary (called lexicon), manually developed by lexicographers and linguists. The resource-rich languages, such as English has some publicly available lexicons, for example, the CMUDict [1]. Creating a large lexicon is very expensive, as well as time and memory consuming.

Another early approach was based on data-driven ‘hand-written’ rules which were devised by linguistic experts. Typically, these rules are constructed in the form of $A/C/B \mapsto y$, meaning the letter $C$ is rewritten as phoneme $y$, when $C$ occurs in the context of letters $A$ and $B$. Rule-based approaches usually work well, but there are always exceptions in each rules.

The latest approaches in designing G2P models involve various ‘statistical’ methods. These approaches are also data-driven, but these techniques not only learn from data, but use statistical techniques to do so. The huge improvements in computing power in recent years give us opportunity to apply ‘machine learning’ based methods in problems that could not be solved by rules. Various deep learning models enable us to utilize the immense amount of data and processing power we have in our disposal. We discuss more details about different approaches in Sect. 3.

In this paper, we propose an encoder-decoder based Neural Machine Translation (NMT) model to convert Bangla words into their pronounceable forms. We adopted this NMT model by mapping our problem into a sequence-to-sequence deep learning problem. Our G2P task is

*e-mail: arif.ahmad-cse@sust.edu
distinguished in another important way: whereas the machine translation tasks are scored with the relatively forgiving BLEU scores, in the G2P task, a phonetic sequence must be exactly correct in order to get credit when scored. Our contributions include:

- Unlike other G2P models, our model outputs the phonemic representations of the given words. So we can easily extract any type of phonetic units, such as phonemes, di-phones, tri-phones, syllables, etc. from the output.
- We provide a systematic comparison with existing approaches using a large lexicon that we have developed.
- We intend to make our lexicon publicly available on Github in order to ensure replicability and to enable future research.

The remainder of the paper is structured as follows. Section 2 defines the Bangla pronunciation problem elaborately and discusses the necessity of developing a sequence-to-sequence model for G2P task. Section 3 presents the previous approaches to solve G2P problems. Section 4 discusses the process of data preparation. In Sect. 5, we describe the architecture of our model in detail. Sections 6 and 7 illustrates the experimental process and results, respectively. Section 8 concludes the discussion by summarizing the experiments and by pointing out some scopes for future research.

2. REVIEW OF BANGLA PRONUNCIATION PROBLEMS

Bangla is a rich and influenced language, which has evolved through the passage of time and adopted many words from several foreign languages. It has been originated from Sanskrit or Devnagari language, but the phonetic variation of Bangla is higher than any other language originated from Sanskrit. Therefore, the number of issues that should be taken care in G2P conversion are large and complicated.

A major issue in pronouncing Bangla words is determining the implicit vowel phoneme (schwa) associated with each consonant. The pronunciation of the implicit vowel can be anything among /ə/, /o/ and /null/. Some examples of schwa-deletion (and ‘retention’) are: ‘কল’ /bolbo/ (/ə/ is deleted from the second letter, and changed to /o/ in first and third letter), ‘সম’ /sarna/ (/ə/ is deleted from the last letter), ‘অথচ’ /atɔkɔ/ (there is no schwa-deletion, but in the last two letters, /ə/ is changed to /o/). A rule-based approach cannot solve this issue, because there is not specific grammatical rules defined for schwa-deletion in Bangla.

Another ambiguous situation arises with the pronunciation of ‘শ’. It has two pronunciations: /ʃ/ and /s/. It is observed that, in most of the proper Bangla words, ‘শ’ is pronounced as /ʃ/ (as in ‘সিল’ /ʃil/). On the other hand, in most of the foreign words adopted in Bangla, ‘শ’ is pronounced as /s/ (as in ‘ইসলাম’ /islam/, an Arabic word). But exceptions exist in both cases.

Studies show that more than 50% errors occurred in G2P module are caused by the above two issues [2]. There remain a few more issues though. The vowel phoneme ‘া’ /æ/ is sometimes replaced with ‘আ’ /e/ phoneme. An example is ‘দেশ’ /dækша/. Besides, some conjuncts have also ambiguous pronunciations.

3. RELATED WORK

Many attempts had been taken for solving grapheme-to-phoneme problems in the past few decades. We have seen extensive works with various approaches — from rule-based approaches to the current state-of-the-art deep-learning based approach. We first give a review of the current status of the modern G2P models for different languages. Then we review the related works for Bangla G2P conversion completed till date, including manual and rule-based approaches.

Most of the G2P works done in the past pertain to English language. Chen et al. [3] proposed several machine learning models such as a conditional maximum entropy model, a joint maximum entropy n-gram model and a joint maximum entropy n-gram model with syllabification. The authors reported results using Phoneme Error Rate (PER) and Word Error Rate (WER) metrics. For the CMU pronunciation dictionary (known as CMUDict) [1] they reported best PER is 1.4% and WER is 8.4%. Taylor [4] proposed a Hidden Markov Model (HMM) approach to solve the G2P problem, and achieved an accuracy of 61.08% on a sufficiently large data-set. The joint-sequence models proposed by Bisani et al. [5] also performed well.

In recent years, neural network approaches have also been proposed for G2P problems. For example, Bilcu [6] investigated different types of neural network structures and found that multi-layer perceptrons performed best.

G2P conversion can also be considered as a machine translation problem where we need to translate source graphemes into target phonemes. Yao and Zweig [7] applied first such method using bi-directional Long-Short-Term-Memory neural networks (LSTMs) and reported a WER of 23.55% on CMUDict. Their experiment outperformed the existing joint-sequence model [5].

Hybrid models were also found effective. For example, Wu et al. [8] combined a joint n-gram model with a conditional random fields (CRF) model, and Hahn et al. [9] combined a basic joint n-gram model with a decision tree model. Rao et al. [10] used deep bi-directional LSTM (DBLSTM) with a connectionist temporal classification (CTC) layer. They combined this model with a joint n-gram model and obtained a WER of 21.3% on CMUDict.
An extensive cross-language studies have been conducted by many researchers. Deri and Knight [11] conducted one such study using an adaptive approach where they explored 229 languages. For Bangla, their reported WER is 66.2%.

Works on developing Bangla G2P models are sparse. So we looked at the literature of other similar languages, such as Hindi. Narasimhan et al. [12] used a rule-based approach with additional morphological analysis. They particularly focused on solving the ‘schwa-deletion’ problem. The study of Thu et al. [13] provides a good comparison using machine learning methods for G2P conversion of the Burmese language—a another under-resourced language. They experimented and evaluated seven different algorithms using a small amount of labeled data-set. These include Support Vector Machines (SVM), Conditional Random Fields (CRF), Joint-Sequence Models (JSM), Weighted Finite-state Transducers (WFST), and Recurrent Neural Networks (RNN).

Now we discuss the G2P models developed for Bangla. To our best knowledge, the existing rule-based Bangla G2P systems are as follows: Mosaddeque et al. [14] adopted the linguistic rules proposed in the Bangla Academy pronunciation dictionary [15]. Their implemented G2P system achieves an accuracy of 97.01% on seen corpus of 736 words. On an unseen corpus of 8,399 words, it achieves an accuracy of 89.48%. Basu et al. [16] proposed another rule-based system, which takes into account three types of information, i.e. orthographic information with their preceding and succeeding context, Parts-of-Speech (POS), and the semantic content. Their system consists of only 21 rules with which it achieves an accuracy of 91.48% on 9,294 words from 1,000 sentences. Ghosh et al. [17] proposed a sub-word based approach for G2P conversion where they used sub-words (suffixes) to reduce the size of lexicon. They achieved an accuracy of 80%.

The use of statistical methods for solving Bangla G2P problem is extremely rare. Google released their Bangla Text-to-Speech (TTS) system [2] in 2016, where they applied machine learning to develop a G2P model. They prepared a lexicon of (approximately) 65,000 words, among which 37,000 were used to train the G2P model. They achieved a WER of 18.5%. Chowdhury et al. [18] proposed a G2P model using CRFs. They used two public lexicons and trained them separately. They achieved a WER of 40.61% on Google lexicon [2] and a WER of 9.69% on CRBLP lexicon [19].

The limitations of existing rule-based efforts for Bangla G2P are that they are hardly comparable due to the lack of non-shareable resources and/or not maintaining the same evaluation condition or performance metrics. In our work, we try to overcome those limitations by maintaining a widely used performance metric, which is Word Error Rate (WER). Besides, we applied an encoder-decoder deep neural network model for the first time to solve Bangla G2P problem. The encoder-decoder model has the significant advantage of not requiring a separate alignment step between the input graphemes and output phonemes. We describe our experiments in detail in the subsequent sections.

4. DATA PREPARATION

Preparing the data-set was one of the most challenging task for this research. We did not have a large lexicon in Bangla (like CMUDict for English) to start our experiments with. Previous research attempts in solving Bangla G2P problem used very small data-sets, many of which were not possible to obtain. Fortunately, Google released a Bangla lexicon (approximately) 65,000 words in 2016 [20]. We used that lexicon as our starting point. We describe our lexicon creation and text normalization processes in the following subsections.

4.1. Lexicon Creation

We carefully curated a Bangla lexicon of (approximately) 135,000 words from the following sources:

- The Google lexicon mentioned above contained 65,000 words, among which 5,000 were English words that occur occasionally in Bangla texts. We excluded those English words, and took the rest 60,000 words for our lexicon.
- We took another 15,000 words from a Bangla pronunciation dictionary called ‘Bangla Uccharon Ovidhan’ created by Bangla Academy [15].
- We collected the rest 60,000 words by crowd-sourcing. The people who volunteered for this task were students of Linguistics and Computer Science. They had been guided thoroughly before contributing to the project. These 60,000 words were taken from a list of most frequently used Bangla words.

After gathering all the data from the above mentioned sources, we obtained a reasonably large lexicon of (approximately) 135,000 pronunciations. A manual checking and cleaning was also performed to ensure the consistencies among the entries and to eliminate human errors. Then we split the data in 94-3-3 ratio (94% data for training, 3% data for cross-validation, 3% data for testing), which is a standard distribution for modern deep learning experiments. Table 1 shows the distribution of our lexicon data.

Now we have a fairly large Bangla lexicon in our possession. We name it SUST-LEX and we intend to release it for public use. We hope that Bangla research community will find it useful and also contribute to the further improvements (and addition) to this lexicon as well.
4.2. Text Normalization

We normalized the source-words of the lexicon in order to improve performance of the G2P converter. Although the word “normalization” usually refers to the conversion of non-standard words (NSW) into the orthographic forms, we use it here to indicate the removal/replacement of unnecessary letters from the source-words. We observed that many pairs of letters and/or diacritics in Bangla have the same pronunciations. For example the letters য and য have the same pronunciation /j/. So we replaced all the occurrences of য with য. Similarly the letters হ and হ, and their diacritics চ and চ have the same pronunciation /i/. The replacement rules are discussed in [21] and [22]. The list of all replacement rules are mentioned in Table 2.

5. MODEL ARCHITECTURE

In this section, we present our approach that uses a sequence-to-sequence recurrent neural network (RNN) model to predict phonemic representation from grapheme sequence. It is to be noted that, there are two ways to transcribe speech. The phonemic representation/transcription involves representing speech using just a unique symbol for each phoneme of the language. The goal of a phonemic transcription is to record the ‘phonemes as mental categories’ that a speaker uses, rather than the actual spoken variants of those phonemes that are produced in the context of a particular word. Phonetic representation transcription on the other hand specifies the finer details of how sounds are actually made. So, for example, a non-English speaker trained in the IPA could look at a phonetic transcription, and know how to pronounce it accurately without knowing the rules about English phonemes.

5.1. Building the Encoder-Decoder Model

We have adopted an encoder-decoder based neural machine translation (NMT) model [23] to perform the G2P conversion task. Figure 1 shows a simplified flow diagram of our model.

At first the input text goes through a ‘normalizer’ that performs some trivial replacements mentioned in Sect. 4.2. We observed that ‘normalized text’ performs better than ‘raw text’ in our model.

Both the encoder and decoder networks are Recurrent Neural Networks (RNNs) with Gated Recurrent Unit (GRU) cells [24]. We preferred the GRU cells instead of popularly used LSTM cells, because GRU can help the model converge faster in the training phase. GRU cells are aimed to solve the vanishing gradient problem which comes with a standard RNN. To solve this vanishing gradient problem of GRU uses, so called, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction.

The detailed functionality of GRU cells can be found in the paper of its inventors [24]. Here we add a brief description of the working of a GRU cell for quick reference. A functional diagram of standard GRU cell is given in Fig. 2.

Table 2 Replacement rules for text normalization.

<table>
<thead>
<tr>
<th>Letter/Symbol</th>
<th>Replaced with</th>
<th>Pronunciation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ভ</td>
<td>ভ</td>
<td>/j/</td>
<td>ভাস → ভাস /id/</td>
</tr>
<tr>
<td>টি</td>
<td>টি</td>
<td>/j/</td>
<td>টানি → টানি /tani/</td>
</tr>
<tr>
<td>ভ</td>
<td>ভ</td>
<td>/u/</td>
<td>ভাম → ভাম /uma/</td>
</tr>
<tr>
<td>ধ</td>
<td>ধ</td>
<td>/u/</td>
<td>ধু না → ধু না /duna/</td>
</tr>
<tr>
<td>ঙ</td>
<td>ঙ</td>
<td>/i/</td>
<td>ঙাম → ঙাম /ima/</td>
</tr>
<tr>
<td>ঞ</td>
<td>ঞ</td>
<td>/i/</td>
<td>ঞাম → ঞাম /ima/</td>
</tr>
<tr>
<td>ড়</td>
<td>ড়</td>
<td>/ai/</td>
<td>ডাই কাও → ডাই কাও /daio/</td>
</tr>
<tr>
<td>঻</td>
<td>঻</td>
<td>/o/</td>
<td>ওই → ওই /oi/</td>
</tr>
<tr>
<td>টো</td>
<td>টো</td>
<td>/ou/</td>
<td>টাইনো → টাইনো /tino/</td>
</tr>
<tr>
<td>ম</td>
<td>ম</td>
<td>/a/</td>
<td>মান → মান /man/</td>
</tr>
<tr>
<td>ন</td>
<td>ন</td>
<td>/n/</td>
<td>নারী → নারী /nari/</td>
</tr>
<tr>
<td>খ</td>
<td>খ</td>
<td>/k/</td>
<td>খারাবর → খারাবর /kharabor/</td>
</tr>
<tr>
<td>ষ</td>
<td>ষ</td>
<td>/f/</td>
<td>মানুষ → মানুষ /manuf/</td>
</tr>
</tbody>
</table>
At first, the update gate $z_t$ is calculated with the following equation:

$$ z_t = \sigma(W^{(c)}x_t + U^{(c)}h_{t-1}) $$

Here $\sigma$ denotes a sigmoid function. $W^{(c)}$ and $U^{(c)}$ are weight vectors for the particular unit. $h_{t-1}$ is the output of the previous unit. After calculating the update gate, the reset gate is used from the model to decide how much of the past information to forget. To calculate it, we use:

$$ r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) $$

This formula is the same as the one for the update gate. The difference comes in the weights and the gate’s usage. After calculating the gates, the ‘current memory content’ will use the reset gate to store the relevant information from the past. It is calculated as follows:

$$ h'_t = \tanh(Wx_t + r_t \odot Uh_{t-1}) $$

Here $\odot$ denotes the Hadamard product (element-wise multiplication). Finally, as a last step, the network needs to calculate $h_t$ — a vector which holds information for the current unit and passes it down to the network. In order to do that, the update gate is needed. It determines what to collect from the current memory content — $h'_t$ and what from the previous steps — $h_{t-1}$. That is done as follows:

$$ h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t $$

The above process shows how GRUs are able to store and filter information using their update and reset gates. That eliminates the vanishing gradient problem since the model is not washing out the new input every single time but keeps the relevant information and passes it down to the next time steps of the network. If carefully trained, they can perform extremely well even in complex scenarios.

5.2. Work-flow of the Model

The model is split into two parts: An encoder which maps the source-text to a “thought vector” that summarizes the text’s contents, which is then input to the second part of the neural network that decodes the “thought vector” to the destination-text. The neural network cannot work directly on texts. So first we need to split the input word into a sequence of letters and convert each letter to an integer-token using a tokenizer. But the neural network cannot work on integers either, so we use a so-called Embedding Layer to convert each integer-token to a vector of floating-point values. The embedding is trained alongside the rest of the neural network to map letters with similar semantic meaning to similar vectors of floating-point values. The full architecture of our model is illustrated in Fig. 3.

Let’s consider a Bangla word “মানুষ” /manu/ which should be converted to its phonemic form “মানুষ” /ma nu/. We first convert the entire vocabulary of our data-set to integer tokens so the text “মানুষ” becomes [22, 43, 185]. Each of these integer-tokens is then mapped to an embedding-vector with e.g. 128 elements, so the integer-token 22 could for example become $[0.12, -0.56, \ldots, 1.19]$ and the integer-token 43 could for example become $[0.39, 0.09, \ldots, -0.12]$, and so on. These embedding-vectors can then be input to the Recurrent Neural Network, which has 3 GRU-layers. The last GRU-layer outputs a single vector—the ‘thought vector’ that summarizes the contents of the source word, which is then used as the initial state of the GRU-units in the decoder-part. The destination-text “মানুষ” /ma nu/ is padded with special
markers “ssss” and “eeee” to indicate its beginning and end, so the sequence of integer-tokens becomes [2, 22, 43, 79, 3]. During training, the decoder will be given this entire sequence as input and the desired output sequence is [22, 43, 79, 3] which is the same sequence but time-shifted one step. We are trying to teach the decoder to map the “thought vector” and the start-token “ssss” (integer 2) to the next letter “া” (integer 22), and then map the letter “া” /ma/ to the letter “ Butt ” /nu/ (integer 43), and so forth.

5.3. Hyperparameters

We started our experiment with the recommended hyperparameter settings in [23]. After adjusting the parameters for a few iterations of the model, and studying the observations of [18], we tuned the parameters as follows.

The embedding layer consists of a 128 dimension vector. Both the encoder and decoder have three GRU layers. We used GRU cells over LSTM cells because GRU doesn’t need a memory unit, is simpler, easier to modify and trains faster. Also, GRUs perform better on relatively smaller and medium sized data set when doing language modeling. Both the encoder and decoder have 3 layers of GRU cells. We ran 30 epochs on every iteration of model training. Also, early stopping was enabled so that the model would stop training if the cost is not improved for five epochs. This would prevent the ‘over-fitting’ of the training data. Increasing the epochs more than this did not perform any better. We trained the data in mini-batches with each batch having 256 data-points. The hyperparameters used in our model are summarized in Table 3.

6. EXPERIMENTS

Our model is different from other G2P models, in terms of output. Usually a G2P model takes word as input and generates a sequence of phonemes as output. But in our model, we generate the ‘phonemic representation’ of the input word, instead of phonemic sequences. We choose this because phonemic representation has a wide range of application areas. For example, in a text-to-speech synthesis system, the phonemic representation can be used.
to parse the text in various forms of units e.g. phones, di-phones, tri-phones, syllables, etc. Some examples of our G2P model’s output are given in Table 4.

7. PERFORMANCE ANALYSIS AND DISCUSSION

To evaluate the performance of our system, we computed word error rate (WER) in the different phases of the experiment. The formula for computing WER is shown in Eq. (5).

\[
WER = \frac{100 \times \text{No. of words with phoneme error}}{\text{total words}}
\] (5)

After training the model with the tuned hyperparameters mentioned in Sect. 5.3, we obtained a WER of 5.21% on training data and a WER of 12.49% on test data. After analyzing the erroneous outputs, we found that, errors occurred mostly on longer words. This is because the frequency of longer words is much less in the data-set. Also, the lower WER in the training set indicates over-fitting, which can be reduced by including regularization in the model.

We compared our model’s performance with the existing best G2P models for Bangla. It was difficult to make an extensive comparison due to the lack of common test data and evaluation metric. Table 5 shows that, our model clearly outperforms the existing state-of-the-art rule-based and ML-based models. To our knowledge, our model is the first sequence-to-sequence implementation for the Bangla G2P problem. The Bangla research community can extend this approach further by improving the data-set and model architecture.

We have used our G2P model in an existing Bangla text-to-speech system Subachan [25] and observed an improved performance. We have conducted a Mean Opinion Score (MOS) test, a popular subjective evaluation test for assessing the naturalness of TTS systems. We gathered 30 volunteers who would participate in a listening test. All the participants were undergraduate students, aged from 20 to 26. The listeners listened to 20 random sentences generated by Subachan (with and without using our G2P model) and rated the systems in a scale of 5. The original system obtained a MOS of 3.25 out of 5. The new system using our G2P model obtained a MOS of 3.57, which is a significant 9.85% improvement from the previous one.

8. CONCLUSION

In this paper, we presented a novel approach to solve Bangla grapheme-to-phoneme problem. We tackled the problem by adopting a machine translation model in contrast to traditional joint G2P modeling approach. Our encoder-decoder model does not require explicit grapheme-to-phoneme alignment, thus reduces pre-processing works significantly. We also developed a large Bangla lexicon of more than 135,000 entries which we plan to release for future researches. We incorporated our model into an existing text-to-speech system [25], and obtained satisfactory results. To improve the model further, future researches should focus on enhancing the data-set and approaching hybrid models which were found effective in other languages.

REFERENCES