A Sequence-to-Sequence Pronunciation Model for Bangla Speech Synthesis

Arif Ahmad
Department of Computer Science and Engineering
Shahjalal University of Science and Technology
Sylhet, Bangladesh
arif_ahmad-cse@sust.edu

Mohammed Raihan Hussain
Department of Computer Science and Engineering
Leading University
Sylhet, Bangladesh
rhzinuk@gmail.com

Mohammad Reza Selim
Department of Computer Science and Engineering
Shahjalal University of Science and Technology
Sylhet, Bangladesh
selim@gmail.com

Muhammed Zafar Iqbal
Department of Computer Science and Engineering
Shahjalal University of Science and Technology
Sylhet, Bangladesh
mzi@sust.edu

Mohammad Shahidur Rahman
Department of Computer Science and Engineering
Shahjalal University of Science and Technology
Sylhet, Bangladesh
rahmanms@sust.edu

***Abstract*— Extracting pronunciation from written text is necessary in many application areas, especially in text-to-speech synthesis. 'Bangla' is not completely a phonetic language, meaning there is not always direct mapping from orthography to pronunciation. It mainly suffers from 'schwa deletion' problem, along with some other ambiguous letters and conjuncts. Rule-based approaches cannot completely solve this problem. In this paper, we propose to adopt an Encoder-Decoder based neural machine translation (NMT) model for determining pronunciations of Bangla words. We mapped the pronunciation problem into a sequence-to-sequence problem and used two 'Gated Recurrent Unit – Recurrent Neural Network's (GRU-RNNs) for our model. We fed the model with two types of input data. In one model we used 'raw' words and in other model we used 'pre-processed' words (normalized by hand-written rules) as input. Both experiments showed promising results and can be used in any practical application.**

Keywords*— encoder-decoder, sequence-to-sequence, gru-rnn, pronunciation model

I. INTRODUCTION

In any language, determining pronunciation from written text is not always a trivial task. If a language have a direct mapping between the orthographic representation and pronunciation, it is called a phonetic language. Bangla, a modern 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 with a phenomenon called 'schwa deletion'. 'Schwa' is an implicit vowel phoneme ('অ' /ɔ/ for Bangla) associated with every consonant of a language. In Bangla, the 'schwa' phoneme is sometimes kept as is, sometimes deleted, and sometimes replaced with 'ও' /o/ vowel. Besides, some letters and conjuncts have ambiguous pronunciation too.

A phonemic representation is the pronounceable form of the written text. Correct phonemic representation is required in many applications, such as text-to-speech synthesis and speech recognition. The oldest approaches of developing 'pronunciation model' were usually 'rule-based'. 'Hand-written' rules were created with the help of phonetic experts to derive the pronunciation of words. The second approach is 'data-driven' approach, where rules were automatically learned from data. Also pronunciation lexicons are sometimes used along with rules. The latest approach of designing pronunciation models is 'statistical' approach. This approach is also data-driven, but these techniques not only learn from data, but use statistical techniques to do so.

Works on solving 'Bangla' pronunciation problems are very rare. So we have looked at the literatures of other similar languages, such as Hindi. Reference [1] used a rule-based approach with additional morphological analysis. They particularly focused on 'schwa deletion' problem. An overview of different pronunciation models were presented in [2] where the author concluded that more research is required in this area. More advanced, statistical experiments are used in recent years, such as [3] and [4]. S. A. Chowdhury et. al. [5] proposed a machine learning based pronunciation model using Conditional Random Fields (CRFs) algorithm and obtained about 85% overall accuracy.

It is obvious from the above-mentioned works that the older rule-based approaches cannot solve the pronunciation problem completely. Fortunately, the huge improvements in computing power in recent years gives us opportunity to apply 'machine learning' based methods in problems that couldn't 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. In this paper, we have proposed sequence-to-sequence based deep learning model to solve the Bangla pronunciation problem. Although some machine learning techniques had been used in recent years [4] [5], our work is novel in two aspects. First, our model generates the 'phonemic form' of a given word whereas others' generate a list of phonemes. We preferred the phonemic forms over the phoneme lists, because it can be parsed in various ways, such as phones, di-phones, tri-phones, syllables, etc. Second, we have used a sufficiently large amount of data (about 80,000 words in total) to train our deep learning model. A data-set of this magnitude was never used in previous works. We gathered Bangla lexicons from different sources (described in section III) and manually removed the inconsistencies that were present in the data. We plan to release our lexicon publicly, so that people can use it to evaluate any related work.

We have implemented an Encoder-Decoder based [6] Neural Machine Translation (NMT) model to convert Bangla words into their pronounceable forms. We have adopted this NMT model by mapping our problem into a sequence-to-sequence 'deep learning' problem. We describe our work in sub-sequent sections. Section II defines the problem elaborately and discusses the necessity of developing a sequence-to-sequence model for pronunciation task. Section III discusses the process of data preparation. In section IV, we describe the architecture of our model in detail. Section V illustrates the experimental process and results. Finally, section VI concludes the discussion by summarizing the experiments and pointing out future research direction.

II. Review of Bangla Pronuciatoin Problems

As mentioned in the section I, Bangla is not completely a phone language. It struggles mainly in pronouncing the 'schwa' phoneme, i.e. the implicit 'অ' /ɔ/ sound associated with each consonant. Some examples of 'schwa deletion' are: 'বলব' /b o l b o/ ('অ' /ɔ/ deleted from the middle-letter of the word), 'নগর' /n ɔ g o r/ ('অ' /ɔ/ deleted from last letter of the word), 'শহর' /ʃ ɔ h o r/ ('অ' /ɔ/ is replaced with 'ও' /o/ in second letter). A rule-based approach cannot solve the issue, because there is no grammatical rule for pronunciation in Bangla.

Another ambiguous situation arises with the pronunciation of 'স'. It has two pronunciations: /ʃ/ and /s/. It is observed that, most of the 'proper' Bangla words pronounce 'স' as /ʃ/. On the other hand, most of the foreign words adapted to Bangla pronounce 'স' as /s/. But exceptions exist in both the cases.

The vowel phoneme 'এ' /e/ is sometimes replaced with 'এ্যা' /æ/ phoneme. An example is দেখা /d æ kʰ a/ ('এ' /e/ becomes 'এ্যা' /æ/). Besides these, some conjuncts also have ambiguous pronunciations.

III. Data Preparation

A sufficiently large lexicon was needed to train the pronunciation model. We obtained our lexicon from several sources. Google released a public Bangla lexicon [7] of ~60,000 words. They used this lexicon to develop their Bangla speech synthesizer [8]. They annotated the pronunciations in ToBI notation. We have adopted Google's lexicon, but changed the pronunciations into Bangla orthographic form. We have collected another lexicon from Bangla Academy Dictionary, which contains ~40,000 words.

Upon gathering all the lexicons, we compiled a lexicon consisting of ~80,000 words. We split the data in 80-10-10 ratio (80% data for training, 10% data for validation, 10% data for testing).

We have also compiled a lexicon of 'normalized' form of those words. By 'normalized' we mean the removal/replacement of unnecessary letters from the words. For example, the character 'শ' and 'ষ' have the same pronunciation /ʃ /. So we replaced all 'ষ' with 'শ'. The replacement rules are discussed in [9], [10] and [11]. The list of all replacements we used is given in Table I.

TABLE I. Replace Rules for Normalization

Letter	Replaced with	Example
ঈ	ই	ঈগল=ইগল
ঈ-কার	ই-কার	রাণী=রাণি
ঊ	উ	ঊহ্য=উহ্য
ঊ-কার	উ-কার	শূন্য=শুন্য
ঋ	রি	ঋতু=রিতু
ঐ	ওই	ঐক্য=ওইক্য
ঐ-কার	ও-কার + ই	কৈ=কোই
ঔ	ওউ	ঔষধ=ওউশধ
ঔ-কার	ও-কার + উ	কৌশল=কোউশল
ৎ	ত্	মহৎ=মহত্
ঙ	◌ং	গাঙ=গাং
ণ	ন	হরিণ=হরিন
য	জ	যখন=জখন
ষ	শ	মহিষ=মহিশ

IV. Model Architecture

A. *Building the Encoder-Decoder Model*

Our Sequence-to-Sequence model uses the Encoder-Decoder Architecture. We have adopted a neural machine translation model [12] to perform the pronunciation conversion. Fig. 1 shows a simplified flow diagram of our model. We have conducted our experiment on two types of inputs: 'raw' words and 'normalized' words. We will compare the outcomes of the two experiments in the next section. 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. 2.

B. *Work-flow of the Model*

Let's consider a Bangla word "মানুষ" /m a n u ʃ/ which should be converted to its phonemic form "মানুশ্". We first convert the entire vocabulary of our data-set to integer-tokens so the text "মা নু ষ" becomes [12, 54, 197].

Each of these integer-tokens is then mapped to an embedding-vector with e.g. 128 elements, so the integer-token 12 could for example become [0.12, -0.56, ..., 1.19] and the integer-token 54 could for example become [0.39, 0.09, ..., -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 "মা নু শ্" is padded with special markers "ssss" and "eeee" to indicate its beginning and end, so the sequence of integer-tokens becomes [2, 12, 54, 79, 3]. During training, the decoder will be given this entire sequence as input and the desired output sequence is [12, 54, 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 "মা" (integer 12), and then map the letter "মা" to the letter "নু" (integer 54), and so forth.

C. *Hyperparameter Tuning*

We started our experiment with initial Hyperparameter settings. After adjusting the parameters for a few iterations of the model, and studying the observations of [5], we tuned the parameters as follows. 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 modelling. Both the encoder and decoder have 3 layers of GRU cells. We ran 20 epochs on every iteration of model training. Increasing the

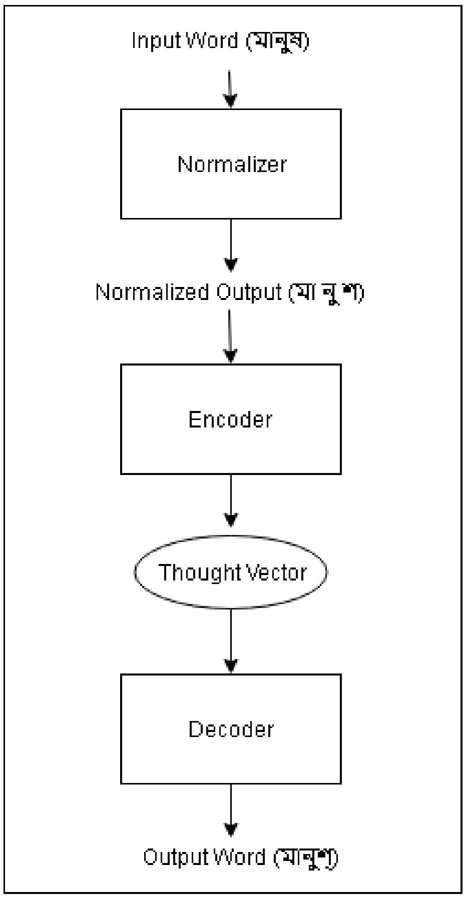

Fig. 1: Flow diagram of the encoder-decoder model

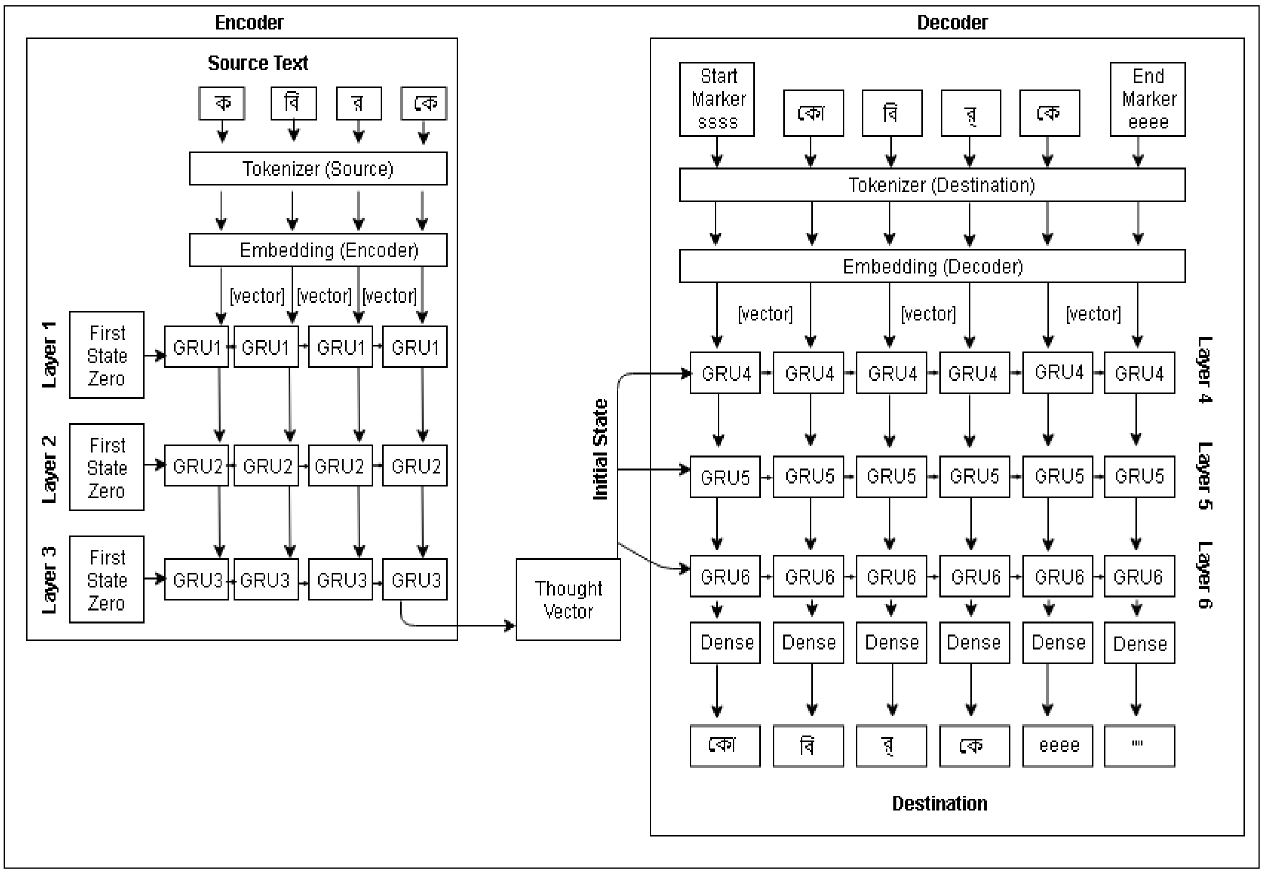

Fig. 2: Architecture of sequence-to-sequence model

epochs more than this did not perform any better. The hyperparameters used in our model are summarized in Table II.

TABLE II. HYPERPARAMETERS USED IN THE MODEL

Hyperparameter	Value
Embedding Dimension	128
State Size	512
Batch Size	500
Number of Epochs	20
RNN Cell Variant	GRU
Encoder Depth	3
Decoder Depth	3
Encoder	Bidirectional
Decoder	Bidirectional

V. RESULTS & PERFORMANCE ANALYSIS

Our model is different from other pronunciation models, in terms of output. Usually a pronunciation model takes 'word' as input and generates 'phonemes' as output. But in our model, we generate the 'phonemic representation' of the input words, instead of phonemes.

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, syllables, etc. Some examples of our pronunciation model is given in table III.

TABLE III. OUTPUT OF THE PRONUNCIATION MODEL

Input word	Normalized word	Output word
মহিষ	মহিশ	মোহিশ্
যাযাবর	জাজাবর	জাজাবর্
কৃষিঋণ	কৃশিরিন	কৃশিরিন্
ঐকমত্য	ওইকমত্য	ওই্কোমত্তো
সদাচরণ	সদাচরন	শদাচরোন্

As mentioned earlier, we conducted our experiment on two types of input words. At first, we trained our model with 'raw' words and got the accuracy of 87.34% on test sets. Then we normalized the input words by some hand-written rules and trained the model with these normalized words. Here we got an accuracy of 89.57%. In training with normalized words, the training-set accuracy was increased significantly due to overfitting. This problem could be solved by increasing the size of data set. Since we could not get more data, we used 'regularization' to handle this situation. Also, we analysed the erroneous outputs manually and found some patterns where errors occurred frequently which can be solved by Hyperparameter tuning. Table IV summarizes the performance of our experiments.

TABLE IV. PERFORMANCE OF PRONUNCIATION MODEL

Type of Input	Training Set Accuracy	Test Set Accuracy
Raw Words	90.82%	**87.34%**
Normalized Words	96.37%	**89.57%**

VI. CONCLUSION

In this paper, we implemented a deep neural network based pronunciation model for Bangla language. We successfully adopted an NMT model to map our problem. Although the accuracy of the model has not achieved perfection yet, it can be used in any real application. For example, in text-to-speech (TTS) systems, users actually hear continues speech and are not bothered by erroneous pronunciation of just a few words. We tested our model in an existing TTS system 'Subachan' and our listeners did not complain about the pronunciation. Having mentioned that, there remains some scope to improve the performance of the model. Our error analysis indicates that, further preprocessing and hyperparameter tuning can result in slightly better performance. Also, adding an 'attention' layer in our model would significantly improve the accuracy of the model.

REFERENCES

[1] B. Narasimhan, R. Sproat, and G. Kiraz, "Schwa-deletion in Hindi text-to-speech synthesis" in International Journal of Speech Technology (2004) 7: 319.

[2] T. Svendsen, "Pronunciation modeling for speech technology," in 2004 Intl. Conference on Signal Processing and Communications, 2004. SPCOM '04., Bangalore, India, 2004, pp. 11-16.

[3] P. Taylor, "Hidden Markov models for grapheme to phoneme conversion", in INTERSPEECH-2005, 1973-1976..

[4] K. Prahallad, A. W. Black and R. Mosur, "Sub-Phonetic Modeling For Capturing Pronunciation Variations For Conversational Speech Synthesis," 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, Toulouse, 2006, pp. I-I.

[5] S. A. Chowdhury, F. Alam, N. Khan and S. R. H. Noori, "Bangla grapheme to phoneme conversion using conditional random fields," 2017 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, 2017, pp. 1-6.

[6] D. Britz, A. Goldie, M.T. Luong, and Q. Le, "Massive exploration of neural machine translation architectures" in arXiv preprint arXiv:1703.03906, 2017.

[7] Online: https://github.com/googlei18n/language-resources/blob/master/bn/data/lexicon.tsv, accessed on August 12, 2018.

[8] A. Gutkin, L. Ha, M. Jansche, K. Pipatsrisawat, and R. Sproat, "TTS for low resource languages: A bangla synthesizer." in LREC, 2016.

[9] Alam, Firoj, S. M. Murtoza Habib and Mumit Khan. "Text Normalization System for Bangla." (2009).

[10] Rashid, M., Hussain, M. and Rahman, M. (2010). Text Normalization and Diphone Preparation for Bangla Speech Synthesis. *Journal of Multimedia*, 5(6).

[11] A. Naser, D. Aich and M. R. Amin, "Implementation of Subachan: Bengali text to Speech Synthesis Software," *International Conference on Electrical & Computer Engineering (ICECE 2010)*, Dhaka, 2010, pp. 574-577.

[12] Online: https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/21_Machine_Translation.ipynb, accessed on August 12, 2018.