Grapheme-to-phoneme Prediction Under Low-resourced Environment

Liang-yu CHEN, Hyukryul YANG, Won Eui HONG
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
{lchenbg, hr.yang, wehong}@cse.ust.hk

Abstract

Speaking English tail words in right pronunciation is hard for non-native learners. In engineering perspective, G2P (grapheme-to-phoneme) prediction under low-resourced environment is challenging issue both in ASR (automatic speech recognition) and TTL (text-to-speech) tasks. Many types of G2P models have been suggested to provide efficient conversion from written words to corresponding phonetics. In this report, we propose a seq2seq G2P model to predict phonetic symbols for English tail words. To achieve satisfactory implementation, the proposed model is compared with baseline RNN introduced from the course. The results show that the model works well over the domain dataset, implicating the possibility of further research extention which can be conducted based on suggested results.

1 Introduction

Global language skill is inevitable in daily life and pronunciation is one of the most important aspect in communication. More accurate speaking improves the quality of conversation, and exact pronunciation is the key to it. Although there are standard dictionaries of phonetic symbols describing pronunciation for the existing English vocabularies, hundreds of new terms are currently being created and gradually being used in everyday life. For example, the pronunciation of rarely used words such as 'Latex', 'agar', 'didlum' can make a trouble for English learners as their second language despite their comprehension of the term is correct. According to Global Language Monitor, a new word is created in every 98 minutes [11]. The neologisms recently created may not be collected in dictionaries in time so that there causes a gap in daily communications between native and non-native English speakers. Though the set of neologisms form a long tail and is waiting to be labeled with phonetic symbols, it does not seem to stop expanding.

For engineering perspective, long tail words (i.e., the words which are not used frequently) take 80% of entire English corpora. Thus any raw dataset of English sentences generated from real rarely contains the tail words. This low-resourced environment often causes a statistical models not to be trained. Thus phonetic symbols have a role to supplement the model by explicitely indicating the exact pronunciations in written format. However, the dictionary of words with phonetic symbols still has its limitation because the tail words are rarely used in raw dataset.

Being inspired by those real-life problems, we come up with a solution to help non-native English speakers to pronounce rarely seen vocabularies in the same way as native English speakers do. To resolve the problems, we developed our idea in two different phases. As the first step, we mapped words and their correcponding phonetic symbols into one-hot vectors. Secondly, we developed our model to make it capable of mapping distinct characters in each English word to corresponding phonetic symbols.

Our proposed model classifies and predicts the phonetic symbols by introducing a sequence to sequence (seq2seq) which maps each English vocabulary to corresponding phonetic symbols in

character level. Given an arbitrary combination of alphabets composing a word, we get an output denoting how the exact word is supposed to be pronounced. This seq2seq model has been used widely through current natural language processing tasks especially in the field of NMT (neural machine translation). As a consequence, our model imitates a translation from English words to phonetic symbols. After training our suggested model, we compared it with GRU RNN model as baseline. Our work is going to be a steppingstone for future work for more advanced G2P techniques. Details are further explained in the following sections.

2 Related Works

The phonetic symbols are used in the area of low-resource speech recognition [4], and transliteration [7]. Though most of speech recognition model deeply implements raw audio resources as major source of input, phonetic symbols can be utilized as meta information. This is because most of dataset follows long tail distribution, so that any words with low usage are hard to be trained by G2P (Grapheme-to-phoneme) model.

G2P is a process of exploiting rules to predict a phonetic symbols for words. As an early step to handle tail words for ARS or TTL tasks, many G2P models were suggested. Somtimes the mapping rules are implicitly embeded into a statistical model of pronunciation dictionary of dataset. Some researches implemented seq2seq G2P conversion based on RNN architecture over low-resourced environment[8].

The metric for evaluating G2P over low-resource is usable when it converges fast, reliably, and stably [13]. Though G2P tasks in current researches aim multi-lingual translation as their applicable area, we implement G2P holding our focus on tail words prediction over English vocabulary.

ASR and TTS systems attain their accurate pronunciations both by a lexicon dictionary and a G2P model. Dictionary-augmented model is a LSTM implemented to force-align network, which is a kind of finite state machine-based character modeling, to reduce dimension in a batch. The goal is also to predict pronunciation for tail words. [2]. Non-sequential greedy decoding method based on convolutional seq2seq model was suggested achieving state-of-the-art performance in G2P [3]. For each English word $w^{(\ell)}$ and its corresponding phonetic symbol $p^{(\ell)}$, a pair $(w^{(\ell)}, p^{(\ell)})$ is a data point. To be implemented in the RNN model, each $w^{(\ell)}$ is encoded into one-hot vector $\mathbf{x}^{(\ell)}$, and corresponding $p^{(\ell)}$ is encoded into another one-hot vector $\mathbf{y}^{(\ell)}$, so the whole data set is like $\mathcal{X} = \{\mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)}\}_{\ell=1}^{|\mathcal{X}|}$.

2.1 Word Embedding: Character Skip-grams

Since the map $\phi_x : \{w^{(\ell)} \rightarrow \mathbf{x}^{(\ell)}\}$ is defined considering its pronounciation of $w^{(\ell)}$, it should be disassambled into its characters. One simple approach to represent the map, is encoding the set of characters into one-hot vector. However, as the example illustrates,

$$w^{(1)} = (\text{‘ } n \text{ ’,‘ } o \text{ ’,‘ } s \text{ ’,‘ } e \text{ ’}),\ p^{(1)} = (\text{‘ } n \text{ ’,‘ } o \text{ ’,‘ } \upsilon \text{ ’,‘ } z \text{ ’})$$
$$w^{(2)} = (\text{‘ } o \text{ ’,‘ } n \text{ ’,‘ } e \text{ ’,‘ } s \text{ ’}),\ p^{(2)} = (\text{‘ } w \text{ ’,‘ } \Lambda \text{ ’,‘ } n \text{ ’,‘ } z \text{ ’})$$

$w^{(1)}$ and $w^{(2)}$ share the same characters, while the sequences are different so the meaning and pronunciation as well. If the words are vectorized in one-hot, the above case could not be resolved. Thus the sequence of character should be considered. The character n-grams is a method to build ϕ_x. It converts a sequence of characters into a set of n-grams. It enables us to compare each sequence in efficient way. For example, previous works implemented the character n-grams for anti-spam filtering [9] or plagiarsm detection [14]. More advanced scheme is Skip-gram [10], which is originally one of a word embedding technique used to learn vector representations of words. It could be implemented in character-based embeddings as well. For example, the Skip-gram of 1 context windows:

$$(\text{‘ } n \text{ ’,‘ } o \text{ ’,‘ } s \text{ ’,‘ } e \text{ ’}) \rightarrow ([\text{‘ } n \text{ ’,‘ } s \text{ ’}], \text{‘ } o \text{ ’}), ([\text{‘ } o \text{ ’,‘ } e \text{ ’}], \text{‘ } s \text{ ’})$$

The window size is a parameter thus more investigation is needed to specify it. The Skip-gram neural network model maximizes the log-likelihood which is given by;

$$\sum_{t=1}^{T} \sum_{a \in C_t} \log p(c_a | c_t),$$

where C_t is the context window of the center character c_t, and T is the length of the word. The probability $\log p(c_a|c_t)$ is softmax with scoring function. The same scheme holds for $\phi_y : \{p^{(\ell)} \rightarrow \mathbf{y}^{(\ell)}\}$, but works inversely ϕ^{-1} when implemented in the model.

2.2 Model Building: Supervised Learning for Classification

For $\mathcal{X} = \{\mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)}\}$, we assume there is a map $\psi_\theta : \{\mathbf{x}^{(\ell)} \rightarrow \mathbf{y}^{(\ell)}\}$ called ground truth. The problem suggested in this proposal is to build $\hat{\psi}_{\hat{\theta}}$ where $\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, P(\mathbf{y}^{(\ell)}|\mathbf{x}^{(\ell)})$ so that $E\big[\psi_\theta(\mathbf{x}^{(\ell)}), \hat{\psi}_{\hat{\theta}}(\mathbf{x}^{(\ell)})\big]$ is minimized. $E(\cdot)$ is generally set to Cross-entropy in many classification problems, but we need more investigation to fix the clear shape of it. Fig 1 depicts the workflow of the model. We are going to define $\hat{\psi}$ in more detail in the following sections.

Figure 1: The Conceptual Workflow of the Proposed Solution

3 Data Set

The Dataset is from Kaggle.com [16]. It contains a list of pairs of word and its pronunciation over 134,000 indexes of North American English. For example,

nose N OW1 Z
ones W AH1 N Z

where N nasal, OW vowel, Z fricative, W semivowel, AH vowel, 1 liquid. The file size is 3.6MB. Preprocessing is not required since the dataset is already well-refined. The specification of dataset is in table 1 shows that

Table 1: Dataset

File	Size	Example
cmudict.dict	3.4M	abuse AH0 B Y UW1 S
cmudict.phones	382B	AH vowel
cmudict.symbols	281B	AY0
cmudict.vp	1.7K	)end-paren EH1 N D P ER0 EH1 N

The total number of samples in dataset is 124,814. Total dataset is splitted in the ratio of $7:3$ and assigned to train set and test set each. Also, for the validation to select parameters, we splitted train set again in the same way. The size of each dataset is summarized in Table 2.

Table 2: Size of Dataset

Train Dataset	Test Dataset	Validation Train dataset	Validation Test datset
87,369	37,445	61,158	26,211

4 Method

Our goal is basically finding the map $\psi_\theta : \{\mathbf{x}^{(\ell)} \rightarrow \mathbf{y}^{(\ell)}\}$. Note that each of $\mathbf{x}^{(\ell)}$ and $\mathbf{y}^{(\ell)}$ are English words, equivalently sequences of characters. I.e., $\mathbf{x}^{(\ell)} = \{x_1^{(\ell)}, x_2^{(\ell)}, \cdots, x_{k-1}^{(\ell)}, x_k^{(\ell)}\}$, where k varies. Thus Multi Layer Perceptrons (MLP), which requires fixed length of input and outputs, could not be implemented. The **sequence to sequence (seq2seq)** over RNN (Recurrent Neural Network) ocould work on it.

4.1 Recurrent Neural Network (RNN)

Consider sequence to sequence model. When a sequence of inputs $(x_1^{(\ell)}, x_2^{(\ell)}, \cdots, x_{k-1}^{(\ell)}, x_k^{(\ell)})$ are given, a standard RNN produces a sequence of outputs by following expression.

$$h_t = \sigma(W^{hx} x_t^{(\ell)} + W^{hh} h_{t-1} + b_h), \text{ where } y_t^{(\ell)} = W^{yh} h_t$$

h_t describes hidden state at time t. W and b weights and bias at each index, respectively. $\sigma(\cdot)$ is softmax as an activation function. RNN where input and the output sequences have different lengths, as it continues to output at every timestep when input is coming in. However, in the case of pronunciation, characters which come later could affect the phonetic symbols of the preceding characters. Therefore, we set RNN as the baseline method, with zero paddings to make the sequence length of the output same as which of the inuput.

4.2 RNN Encoder-decoder

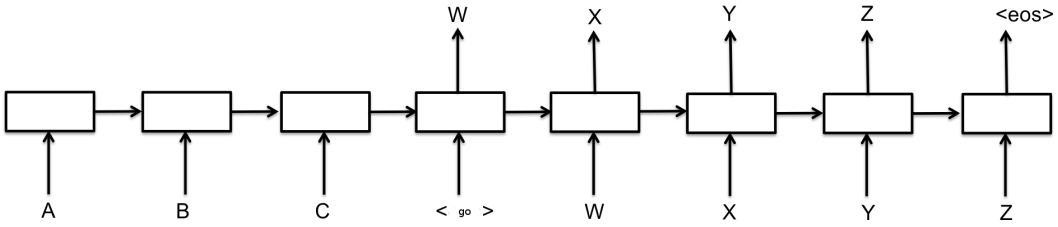

Figure 2: RNN Encoder-Decoder Diagram.

RNN Encoder-Decoder was proposed to tackle the weakness of standard RNN [5], and is called 'Sequence to Sequence' (Seq2seq) [15]. RNN Encoder-Decoder is divided into two parts, encoder and decoder. The encoder consumes input sequence $(x_1^{(\ell)}, \cdots, x_k^{(\ell)})$ and encode it into fixed length vector c. Consequently, the decoder produces $(y_1^{(\ell)}, \cdots, y_{k'}^{(\ell)})$ as an output when c is given. Note that $\mathbf{k} \neq \mathbf{k}'$ for some cases. This RNN Encoder-decoder scheme can be expressed in following equations.

For encoder,

$$h_t = f(x_t^{(\ell)}, h_{t-1})$$
$$c = q(\{h_1, \cdots, h_k\})$$

c is a vector generated from the sequence of the hidden states. $f(\cdot)$ and $q(\cdot)$ are nonlinear functions [1]. Long Short-Term Memory (LSTM) [15] or Gated Recurrent Unit (GRU) [5] are generally used for f and q.

For decoder,

$$h_t = f(y_{t-1}^{(\ell)}, h_{t-1}, c)$$
$$p(y_t^{(\ell)} | y_1^{(\ell)}, \cdots, y_{t-1}^{(\ell)}, c) = g(h_t, y_{t-1}^{(\ell)}, c)$$

$g(\cdot)$ should be the activation function which outputs probability. Finally, the probability of output sequence is,

$$p(y_1^{(\ell)}, \cdots, y_{k'}^{(\ell)} | x_1^{(\ell)}, \cdots, y_k^{(\ell)}) = \prod_{t=1}^{k'} p(y_t^{(\ell)} | y_1^{(\ell)}, \cdots, y_{t-1}^{(\ell)}, c)$$

where the input sequence is given. Therefore, the objective function of RNN Encoder-Decoder model is as below:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \frac{1}{N} \sum_{\ell=1}^{N} \log p_\theta(\mathbf{y}^{(\ell)} | \mathbf{x}^{(\ell)})$$

By utilizing this RNN Encoder-Decoder model, the map $\hat{\psi}_\theta : \{\mathbf{x}^{(\ell)} \to \hat{\mathbf{y}}^{(\ell)}\}$ fits the training set.

5 Experiments

The experiment is done on Ubuntu 16.04, Python 3.6.3, and PyTorch 0.4.1. For the hardware spec, we used one Nvidia GeForce GTX 1080ti. The training time is 3176 seconds for RNN model and 6922 seconds for Seq2seq model.

5.1 Preprocessing

Before the training phase begins, we excluded illegar characters which are not included in the set of English alphabets. The minimum and maximum length of embedding are set to 2 and 20, respectively. 20 is set for the maximum length of word among the input dictionary. For exception handling, we skipped words of which length are out of the boundary. Trimming empty words is done when dataset is loaded.

Each character and phonetic symbols are embeded into one-hot vectors with legnth 30 and 87, respectively. 30 stands for English alphabets plus 4 special characters (‘ ’, ‘.’, ‘-’, ‘ ’ ’). 87 stands for the number of phonetic symbols to predict. The whole dataset is transformed into one-hot tensors, and saved as .npy compression.

5.2 Model Architecture

The figure 3 depicts the GRU RNN model.

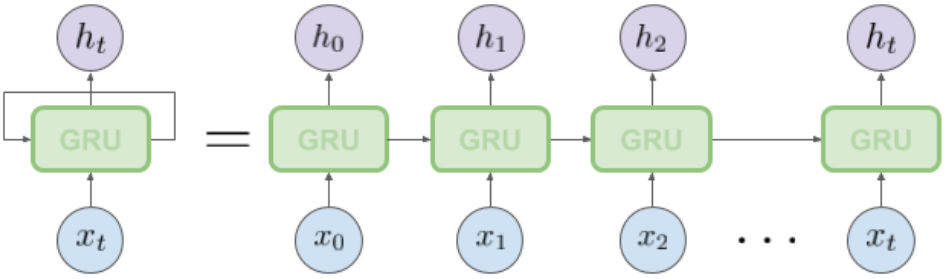

Figure 3: GRU RNN

The architecture for seq2seq with GRU is shown as the figure below, which consists a single GRU unit for each time step followed by a softmax activation. One GRU unit is composed of these building blocks. Update gate $z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$, the reset gate $r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$, Memory content $h'_t = tanh(Wx_t + r_t \odot Uh_{t-1})$, and Final memory content $h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t$.

And softmax acitvaation function on a K-dimensional vector is defined as:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

where j denotes the output units, so $j = 1, ..., K$.

The model reads the 30-dimensional input and outputs a 87-dimensional vector, with their sequence length equivelent to 20. However, the sequence of output is supposed to be of length 21 according to the dataset. Consequently, zero-padding schemes are applied to the model output to append a 87-dimensional vector and make the sequence length the same as the ground truth data.

The main part of the seq2seq model consists of encoder and decoder. Each of them has hidden layer of size 256. The sizes of input and output layer are 30 and 87, respectively. Both encoder and decoder implements Adam optimizer with learning rate 10^{-3}. Each input contains SOS (Start of Sentence) is set to ’\t’, EOS (End of Sentence) is set to ’\n’. For each hidden layer, GRU (Gated Recurrent Unit) is applied. GRU is a memory unit similar to LSTM (Long-short term memory), however it have exhibited better performance over small size of dataset [6].

Our proposed model contais one GRU unit in encoder, and another GRU unit in decoder in symmetric manner. hidden units and output layer is fully connected layer of the form $y = xA^T + b$, which linearly maps input features into output features. Lastly, we used LogSoftmax $\hat{y}_i$ as the last layer, which is simply a logarithm after softmax function.

$$\hat{y}_i = \log \frac{\exp(x_i)}{\sum_{k=1}^{K} \exp(x)}$$

The loss function is NLL (negative log likelihood) loss.

$$E^{(\ell)}(\mathbf{w} \mid \mathbf{x}^{(\ell)}, y^{(\ell)}) = -\sum_i y_i^{(\ell)} \log \hat{y}_i^{(\ell)}$$

The figure 4 depicts the seq2seq model with GRU.

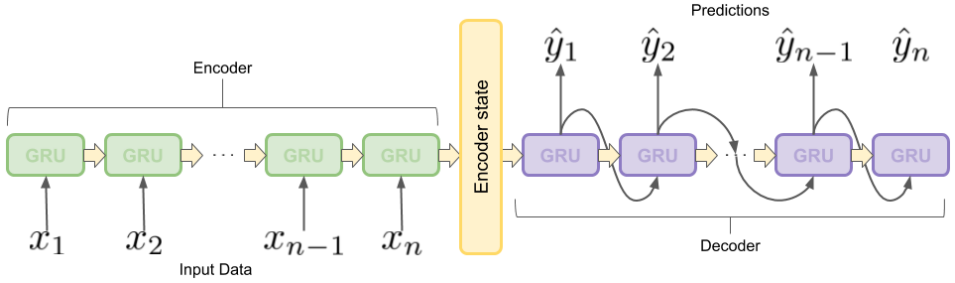

Figure 4: Seq2seq with GRU

5.3 Hyperparameters

We perfomed holdout validation to choose the best parameter setting. The validation space for learning rate is {0.1, 0.01, 0.001}, while optimizers are from {SGD, Adam}. The result of holdout validation is attached in the Table 3. The overall model settings including derived learning rate and optimizer as hyperparameters are presented in Table 4.

Table 3: Holdout Validation Results for Test Loss

	0.01 & SGD	0.01 & Adam	0.001 & SGD	0.001 & Adam	0.0001 & SGD	0.0001 & Adam
GRU RNN	1.7976	**0.7021**	2.2977	1.6666	4.3301	1.9571
GRU seq2seq	1.8388	**1.3443**	1.7137	1.6397	2.3793	1.6601

Table 4: Model Settings

BATCH_SIZE	EPOCHS	MIN_DICT_WORD_LEN	MAX_DICT_WORD_LEN	OPTIMIZER	LEANING_RATE
512	10	2	20	Adam	10^{-2}

5.4 Performance Measure

Since the length of output sequence may vary so that it may not exactly match the length of the sequence of ground truth. For measurement, we implemented BLEU (Bilingual Evaluation Understudy) score, which is often used for machine translation tasks, reported as fitting well with human judgement [12].

6 Results

The result in table 5 shows the model's performance.

Table 5: Models

Model	Train Loss	Test Loss	Test BLEU
GRU RNN	0.484	0.526	0.367
Seq2Seq	0.187	0.867	0.425

Since the length of output sentence varies, precision score does not fit to this task. The evaluation is done with BLEU (BiLingual Evaluation Understudy) Score[12]. This is a variation of precision comparing translated sentence against several translated sentences.

Figure 5 illustrates the negetive log loss of the model for training set, while figure 6 the BLEU Score of the model of RNN model. Figure 7 illustrates the negetive log loss of the model for training set, while figure 8 the BLEU Score of the proposed seq2seq model.

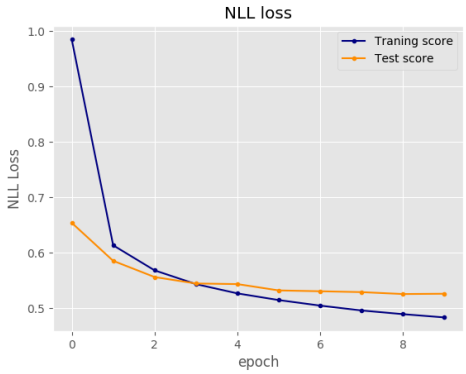

Figure 5: NLL Loss for RNN model

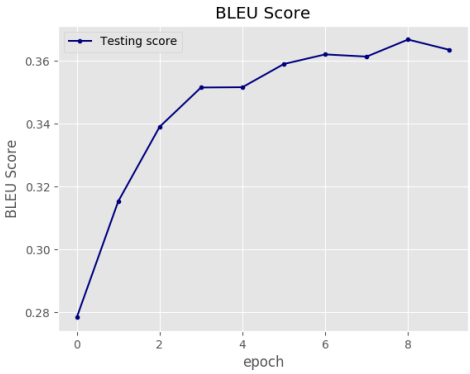

Figure 6: BLEU Score for RNN model

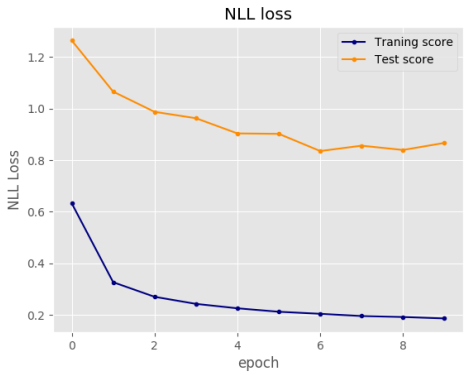

Figure 7: NLL Loss for seq2seq model

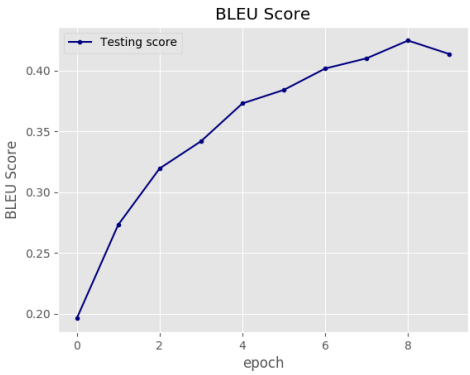

Figure 8: BLEU Score for seq2seq model

Table 6 shows the running example of GRU RNN and Seq2seq model, respectively.

Table 6: Running Examples

Words	GRU RNN	Seq2seq
Latex	L **AE1** T EH2 K S	L **EY1** T EH2 K S
Dit-yan Yeung	D IH1 T Y AE1 N Y **UW1** NG	D IH1 T Y AE1 N Y **AH1** NG
Chalres	CH **AE1** R L **L**	CH **AA1** R L **Z**
Hyukryul Yang	HH **Y** K R **IY0** Y L L Y AE1 NG	HH **IH1** K R **AH0** L Y AE1 NG
Won eui Hong	W **AH1** N Y UW1 HH **N** NG	W **AA1** N Y UW1 **IH0** HH **AA1** NG
ELMO	EH1 L M OW0	EH1 L M OW0
Pickachu	P **AY1** K AA1 **K UW2**	P **IY0** K AA1 **CH UW0**
Starbucks	S T AA1 R B **AH0** K S	S T AA1 R B **IH0** K S
Mcdonalds	M AH0 K D AA1 N AH0 L **D** Z	M AH0 K D AA1 N **D** AH0 L Z
Valkyrie	V **AE1** L K **AY1** R IY0	V **AE2** L K **R IY1** R IY0

7 Discussions

The result shows that seq2seq model outperforms RNN model based on the BLEU score metrics. Moreover, inferring from the random examples we sampled in the previous table, seq2seq model gives a better performance overall.

Nevertheless, deriving from the experiment outcome, the test NLL loss of seq2seq model is higher than which of the rnn model can result from the fact that the loss function we adopted calculates the loss with all dimensions of the input, where some of the dimensions may be redundant. As a consequence, seq2seq can predict well while ignoring errors from redundant dimensions which lead to a high contribution to the loss. Instead, RNN may perform better on minimizing the loss yet not doing well on the final metric. This can be the intrinsic limitations on adopting NLL loss to maximize the BLEU score as they are not directly related to each other.

8 Conclusion

In this report, we presented a seq2seq prediction model to match with corresponding ground truth of phonetic symbols. The seq2seq model with one-hot vector encoding shows better results based on our evaluation metric of the model comparing to which of the simple RNN model, as we expected. This implies on this grapheme to phoneme task, the properties of seq2seq model still hold.

Moreover, we tried to predict phonemes of words which are not used commonly(Table 6). We obtained reasonable results from both models but the result of seq2seq model seems more promising.

9 Additional Information

The hyperlink below is to the YouTube video of our project:

https://youtu.be/9QODUNV1E50

The hyperlink below is to the GitHub page which contains the dataset.

https://github.com/Hyeokreal/ML_FINAL_DATA

The hyperlink below is to the Google Slide which contains the presentation slides.

https://goo.gl/fWZKEX

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Antoine Bruguier, Anton Bakhtin, and Dravyansh Sharma. Dictionary augmented sequence-to-sequence neural network for grapheme to phoneme prediction. *Proc. Interspeech 2018*, pages 3733–3737, 2018.

[3] Moon-jung Chae, Kyubyong Park, Linhyun Bang, Soobin Suh, Longhyuk Park, Namju Kimt, and Longhun Park. Convolutional sequence to sequence model with non-sequential greedy decoding for grapheme to phoneme conversion. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2486–2490. IEEE, 2018.

[4] Dongpeng Chen and Brian Kan-Wing Mak. Multitask learning of deep neural networks for low-resource speech recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(7):1172–1183, 2015.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[7] Wei Gao, Kam-Fai Wong, and Wai Lam. Phoneme-based transliteration of foreign names for oov problem. In *International Conference on Natural Language Processing*, pages 110–119. Springer, 2004.

[8] Preethi Jyothi and Mark Hasegawa-Johnson. Low-resource grapheme-to-phoneme conversion using recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5030–5034. IEEE, 2017.

[9] Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. Words versus character n-grams for anti-spam filtering. *International Journal on Artificial Intelligence Tools*, 16(06):1047–1067, 2007.

[10] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308, 2014.

[11] THE GLOBAL LANGUAGE MONITOR. no-of-words-english. `https://www.languagemonitor.com/global-english/no-of-words/`.

[12] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[13] Dravyansh Sharma. On training and evaluation of grapheme-to-phoneme mappings with limited data. *Proc. Interspeech 2018*, pages 2858–2862, 2018.

[14] Efstathios Stamatatos. Intrinsic plagiarism detection using character n-gram profiles. *threshold*, 2(1,500), 2009.

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

[16] Rachael Tatman. Cmu pronouncing dictionary. `https://www.kaggle.com/rtatman/cmu-pronouncing-dictionary/version/1#cmudict.dict`.