Grapheme-to-Phoneme Conversion for Thai using Neural Regression Models

Tomohiro Yamasaki
Toshiba Corporation, Research and Development Center, Japan
tomohiro2.yamasaki@toshiba.co.jp

Abstract

We propose a novel Thai grapheme-to-phoneme conversion method based on a neural regression model that is trained using neural networks to predict the similarity between a candidate and the correct pronunciation. After generating a set of candidates for an input word or phrase using the orthography rules, this model selects the best-similarity pronunciation from the candidates. This method can be applied to languages other than Thai simply by preparing enough orthography rules, and can reduce the mistakes that neural network models often make. We show that the accuracy of the proposed method is .931, which is comparable to that of encoder-decoder sequence models. We also demonstrate that the proposed method is superior in terms of the difference between correct and predicted pronunciations because incorrect, strange output sometimes occurs when using encoder-decoder sequence models but the error is within the expected range when using the proposed method.

1 Introduction

Grapheme-to-phoneme conversion (G2P) is the task of converting grapheme sequences into corresponding phoneme sequences. Many languages have the difficulty that some grapheme sequences correspond to more than one different phoneme sequence depending on the context.

G2P plays a key role in speech and text processing systems, especially in text-to-speech (TTS) systems. These systems have to produce speech sounds for every word or phrase, even those not contained in a dictionary. In low-resource languages, it is fundamentally difficult to obtain large vocabulary dictionaries with pronunciations. Therefore, pronunciations need to be predicted from character sequences.

In many languages, each word is composed of syllables and each syllable is composed of characters following the orthography rules of that language. This means that G2P for languages with syllabic orthography rules can be formulated as the task of selecting the best path in a lattice generated for a given input word or phrase if we prepare enough orthography rules to make sure that any lattice generated almost certainly includes the path for the correct pronunciation.

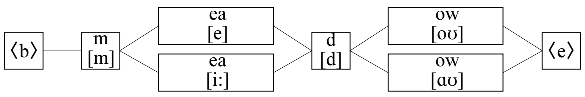

Figure 1: G2P can be formulated as the task of selecting the best path in a lattice. In English, because “ea” can be pronounced as /e/ and /iː/ and “ow” can be pronounced as /oʊ/ and /aʊ/, the word “meadow” has 4 pronunciation candidates, excluding stress position estimation.

As the result of some effort, we prepared Thai orthography rules. Almost all possible paths in a lattice can be generated from these, and each path needs to be evaluated using a phonological language model to select the best path. With this in mind, we propose a novel G2P method based on a neural regression model that is trained using neural networks to predict how similar a pronunciation candidate is to the correct pronunciation. After generating a set of candidates for an input word or phrase using the orthography rules, this model selects the best-similarity pronunciation from the candidates.

In the following sections, we describe the proposed method and explain experiments on a dataset of Thai vocabulary entries with pronunciations collected from Wiktionary. After that, we show that the proposed method outperforms encoder-decoder sequence models in terms of the difference between correct and predicted pronunciations, and demonstrate that incorrect, strange output sometimes occurs when using encoder-decoder sequence models while error is within the expected range when using the proposed method. The code is available at `https://github.com/T0106661`.

2 Related Work

For G2P, converting words into pronunciations does not require as much expertise as needed to prepare correspondences between graphemes and phonemes. Therefore, a method of learning correspondences from a large number of word-pronunciation pairs has been used (van den Bosch and Daelemans, 1993). To solve the problem that graphemes and phonemes often have many-to-many correspondence, a hidden Markov model-based method (Jiampojamarn et al., 2007) and weighted finite-state transducer-based methods (Novak et al., 2012a,b) have been developed. Bisani and Ney (2008) assumed that *graphones* underlie both graphemes and phonemes, and achieved great performance by learning graphones to minimize joint errors.

In addition, there have been some attempts to apply encoder-decoder models to learn end-to-end G2P models. For example, Toshniwal and Livescu (2016) applied a sequence-to-sequence architecture (Sutskever et al., 2014; Luong et al., 2015). Yolchuyeva et al. (2020) and Vesik et al. (2020) applied a transformer architecture (Vaswani et al., 2017) to train models that can deal with English or many other languages.

3 Proposed Method

It is well known that word segmentation (WS) is a necessary preprocessing for languages without word delimiters, such as Chinese, Japanese, and Thai. To solve WS and homograph disambiguation for Thai simultaneously, Tesprasit et al. (2003) enumerated pronunciation candidates for text data with ambiguity and trained a model to select the correct pronunciation from candidates based on the context in which the text data appear.

In contrast, the proposed method trains a model that predicts how similar each candidate is to the correct pronunciation. Although the main process of the proposed method is language independent, we take Thai as an example in this section.

First, we prepared correspondences between graphemes and phonemes by combining Thai characters consisting of 44 consonants, 15 vowels, and several symbols, resulting in an approximately 300-line program and more than 180,000 entries. The prepared entries consisted of 77 characters, 5,772 syllables, and 31 phoneme symbols, and each syllable was composed of up to 7 phoneme symbols.

Characters	Syllables
เส	/see˩˥/
มอ	/mɔɔ˧/
เสมอ	/sa˧ məə˩˥/
ผล	/phon˩˥/, /phon˩˥ la˧/
รั์	(silent)

Table 1: Examples of Thai characters and syllables. Some vowels are not written and some consonants are treated as if written twice.

Suppose that a dataset of vocabulary entries with pronunciations $D = \{(v_i, p_i) \mid i = 0, \ldots\}$ is given. We begin by tracing characters in each vocabulary v_i one at a time to generate a lattice of nodes corresponding to entries. We then enumerate all possible paths in the lattice from one end to the other, and obtain a set of pronunciation candidates $C_i = \{c_{ij} \mid j = 0, \ldots\}$ by joining syllables assigned to nodes. For example, when we have $v_i =$ "กลางคืน"(night), we obtain the set $C_i = \{$/ka˧ laa˧ ŋa˧ khɯɯn˧/, /klaa˧ ŋa˧ khɯɯn˧/, /klaaŋ˧ khɯɯn˧/$\}$.

After that, we calculate the similarity

$$s_{ij} = s(p_i, c_{ij}) = 1 - \frac{d(p_i, c_{ij})}{\max(|p_i|, |c_{ij}|)}$$

to the correct pronunciation p_i for each candidate c_{ij}, where $d(\cdot,\cdot)$ denotes symbol-based edit distance. This s_{ij} takes a maximum of 1 when $c_{ij} = p_i$ and approaches a minimum of 0 as c_{ij} diverges from p_i. For the previous example, we obtained $p_i =$ /klaaŋ˧ khɯɯn˧/; thus $s($/klaaŋ˧ khɯɯn˧/, /klaaŋ˧ khɯɯn˧/$) = 1$, $s($/klaaŋ˧ khɯɯn˧/, /klaa˧ ŋa˧ khɯɯn˧/$) = 13/16$, for example, were obtained.

Using the similarity defined above, we train a neural regression model that predicts how similar each candidate is to the correct pronunciation. More specifically, we train the model to return the similarity s_{ij} from the encoded vectors of v_i and c_{ij} using RNNs with mean absolute error as the loss function to keep each sample error as small as possible.

Figure 2 shows the model architecture. Vocabulary v is converted into a d_v-dimensional vector by a character embedding layer and a bi-directional

GRU (Bi-GRU). Candidate c is converted into a d_c-dimensional vector by a syllable embedding layer, a phoneme embedding layer, and Bi-GRUs, like the network of Lample et al. (2016). Finally, both vectors are concatenated and converted into similarity s by two dense layers. Each layer dimension can be changed depending on the target language.

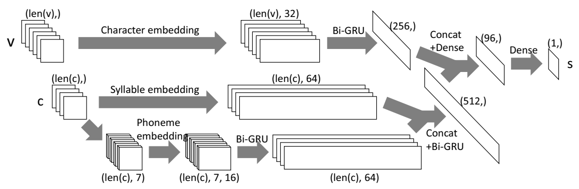

Figure 2: Architecture of the proposed method. Each pair of vocabulary v and candidate c is converted into similarity s. Each layer dimension can be changed depending on the target language.

The model trained in this way is expected to represent the phonological nature of the target language. Therefore, we can predict the pronunciation p' for a given vocabulary v' as follows. As in the training phase, we trace characters in each vocabulary v', generate a lattice, and obtain a set of candidates $C' = \{c'_j \mid j = 0, \ldots\}$. Next, we calculate the similarity s'_j for each pair (v', c'_j) using this model, and find $j_{\max} = \operatorname{argmax}_j s'_j$. Finally, we output the predicted pronunciation $p' = c'_{j_{\max}}$.

As can be seen, both training and predicting processes are language independent. In other words, the proposed method can be applied to languages other than Thai simply by preparing enough orthography rules.

However, one potential problem with the proposed method is that the number of candidates increases exponentially with input length, which can be undesirable for long words and phrases. This is considered in the next section.

4 Experimental Setup and Results

We collected 18,066 Thai vocabulary entries with pronunciations from Wiktionary as an experimental dataset. The vocabulary consisted of not only words but also phrases. We then converted the pronunciations described in a Thai-specific way into International Phonetic Alphabet sequences. For entries without pronunciations but with syllable boundaries, we determined their pronunciations when all candidates for each boundary were uniquely generated. For entries where the pronunciations were unable to be determined, two workers fluent in Thai described the correct pronunciations. The average length of each data was 7.42 in characters, 2.47 in syllables, and 12.48 in phoneme symbols.

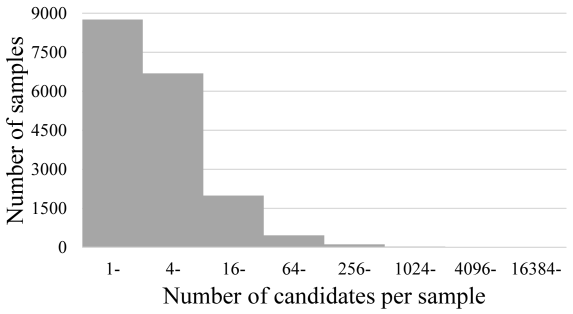

Figure 3: Distribution of the number of candidates. On average, the model has to select the correct pronunciation from about 20 candidates.

First, we examined the number of candidates generated and the coverage rate of the correct pronunciations. Figure 3 shows the distribution of the number of candidates. As seen in the figure, the number is usually less than 10 and seldom greater than 100. In fact, the minimum, mode, median, mean, and maximum were 1, 2, 4, 19.6, and 19,242 respectively. This means that the average stayed slightly less than 20, although the lattice generated for longer input tends to have many branches and can cause an exponential increase in candidates. For the coverage rate, 17,720 of 18,066 correct pronunciations were included in the sets of candidates and 346 were not.

Next, we evaluated our method compared with three G2P baseline models available from SIGMORPHON (2020), namely, a pair ngram model (fst) and two encoder-decoder sequence models (encoder-decoder and transformer, hereinafter ab-

Models	Accuracy	Difference	
		Ave	Max
fst	.670 ± .014	.619	12.3
enc-dec	.932 ± .012	.313	78.9
xformer	.911 ± .015	.360	42.5
ours	.931 ± .006	.217	8.7

Table 2: Performance comparison with three G2P baseline models for Thai. Difference is calculated across all entries. Each result is the average of 10-fold cross-validation on the test data, and thus maximum difference may not be an integer.

breviated to enc-dec and xformer). Table 2 shows the results of 10-fold cross-validation on the test data. For each fold in the experiments, we used 8/10 of entries for training, 1/10 for validation, and 1/10 for testing. We also used accuracy and the difference between correct and predicted pronunciations counted by phoneme symbols as evaluation metrics. In other words, accuracy is the percentage of 0-difference entries.

As a result, 1.17×10^6 parameters were trained. Each training run was composed of about 40 epochs and each epoch took about 13 minutes on 1 GPU (Titan V, 11GB). The absence of decoders in our model is a possible reason why the number of parameters is small and the training time is short.

Table 2 shows that our method achieved high accuracy, small average difference, and small maximum difference, and the accuracy in particular is comparable to those of enc-dec and xformer. In contrast, the low accuracy and large average difference of fst indicate that an ngram model was not able to sufficiently learn the Thai phonological nature, and large maximum differences of enc-dec and xformer indicate the well-known problem of neural network models returning good output when they work, but sometimes making mistakes when they do not.

	การเลือกตั้งประธานาธิบดีฝรั่งเศส (French presidential election)
fst	/kaan˧ lɯak˥˩ taŋ˥˩ pra˩ thaa˧ naa˧ thi˥ bɔɔ˧ dii˧ fa˧ raŋ˥˩ seet˩/
enc-dec	/kaan˧ lɯak˥˩ taŋ˥˩ pra˩ thaa˧ naa˧ thip˥ dii˧ fa˧ raŋ˥˩ pra˩ thaa˧ naa˧ thip˥ dii˧ fa˧ raŋ˥˩ pra˩ thaa˧ naa˧ thip˥ dii˧ fa˧ raŋ˥˩ pra˩ thaa˧/
xformer	/kaan˧ lɯak˥˩ taŋ˥˩ pra˩ thaan˧ baa˧ dii˧/
ours	/kaan˧ lɯak˥˩ taŋ˥˩ pra˩ thaa˧ naa˧ thi˥ ba˧ dii˧ fa˧ raŋ˥˩ seet˩/
correct	/kaan˧ lɯak˥˩ taŋ˥˩ pra˩ thaa˧ naa˧ thip˥ dii˧ fa˧ raŋ˥˩ seet˩/

Table 3: Examples of outputs with errors. The encoder-decoder sequence models sometimes included syllable repetitions and sometimes lacked syllables. For comparison, fst and our method returned outputs with fewer errors.

As shown in Table 3, further investigation revealed that the outputs of the encoder-decoder sequence models sometimes included unnatural syllable repetitions and sometimes lacked syllables in the middle, which are undesirable for TTS systems because they might give the impression that the system is failing. In contrast, our method was able to reduce such mistakes because all candidates followed the orthography rules.

4.1 Additional Experiments

To confirm that the main process of our method can be applied to other languages, we performed additional experiments on the Japanese Hiragana dataset available from SIGMORPHON (2021). This dataset consisted of 10,000 entries and the average length of each data was 4.21 in characters, 6.53 in syllables, and 16.43 in phoneme symbols.

As the result of some effort, we prepared Japanese Hiragana orthography rules. The prepared entries consisted of 85 characters, 405 syllables, and 45 phoneme symbols, and each syllable was composed of up to 6 phoneme symbols.

Models	Accuracy	Difference	
		Ave	Max
fst	.915 ± .002	.187	19.0
enc-dec	.925 ± .006	.161	14.8
xformer	.921 ± .006	.169	13.2
ours	.945 ± .002	.103	14.0

Table 4: Performance comparison with three G2P baseline models for Japanese Hiragana.

Table 4 shows performance comparison with three G2P baseline models. As can be seen, our method also achieved high accuracy and small average difference. However, the maximum differences of enc-dec and xformer are comparable to that of our method. A possible reason why the encoder-decoder sequence models worked well is that the number of long inputs in this dataset was smaller compared with Thai.

5 Conclusion and Future Work

In this study, we proposed a novel Thai G2P method based on neural regression models. We confirmed that the model trained using neural networks to predict the similarity was able to select the correct pronunciations from candidates. The accuracy was .931 and the difference between correct and predicted pronunciations was .217 on average and 8.7 at maximum.

This means that the performance of our proposed method was comparable to that of encoder-decoder sequence models and superior in terms of the difference between correct and predicted pronunciations. In particular, error is within the expected range when using our proposed method. Use of neural regression models not only for G2P but also for summarization and generation opens the possibility that neural network models could reduce strange mistakes.

Our proposed method has the strength that it can be applied to any language by preparing enough orthography rules. However, it also has the weakness of the number of candidates increasing exponentially with input length, which can be a concern for languages with many exceptional orthography rules, such as English.

A method for reducing candidates might thus be needed. There may be an efficient solution to find the correct pronunciation using a given candidate and the predicted similarity. However, these studies and experiments are left as future work.

References

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

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, Rochester, New York. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012a. WFST-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 45–49, Donostia–San Sebastián. Association for Computational Linguistics.

Josef R. Novak, Nobuaki Minematsu, Keikichi Hirose, Chiori Hori, Hideki Kashioka, and Paul R. Dixon. 2012b. Improving wfst-based G2P conversion with alignment constraints and RNNLM n-best rescoring. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, pages 2526–2529. ISCA.

SIGMORPHON. SIGMORPHON Shared Tasks.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Virongrong Tesprasit, Paisarn Charoenpornsawat, and Virach Sornlertlamvanich. 2003. A context-sensitive homograph disambiguation in Thai text-to-speech synthesis. In *Companion Volume of the Proceedings of HLT-NAACL 2003 - Short Papers*, pages 103–105.

Shubham Toshniwal and Karen Livescu. 2016. Jointly learning to align and convert graphemes to phonemes with neural attention models. In *2016 IEEE Spoken Language Technology Workshop, SLT 2016, San Diego, CA, USA, December 13-16, 2016*, pages 76–82. IEEE.

Antal van den Bosch and Walter Daelemans. 1993. Data-oriented methods for grapheme-to-phoneme conversion. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht, The Netherlands. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Kaili Vesik, Muhammad Abdul-Mageed, and Miikka Silfverberg. 2020. One model to pronounce them all: Multilingual grapheme-to-phoneme conversion with a transformer ensemble. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 146–152, Online. Association for Computational Linguistics.

Wiktionary. Wiktionary, the free dictionary.

Sevinj Yolchuyeva, Géza Németh, and Bálint Gyires-Tóth. 2020. Transformer based grapheme-to-phoneme conversion. *CoRR*, abs/2004.06338.