Example-Based Grapheme-to-Phoneme Conversion for Thai

Paisarn Charoenpornsawat and Tanja Schultz

Interactive Systems Laboratories
Carnegie Mellon University
paisarn@cs.cmu.edu and tanja@cs.cmu.edu

Abstract

Several characteristics of the Thai writing system make Thai grapheme-to-phoneme (G2P) conversion very challenging. In this paper, we propose an Example-Based Grapheme-to-Phoneme conversion approach. It generates the pronunciation of a word by selecting, modifying and combining pronunciations from syllables from training corpus. The best system achieves 80.99% word accuracy and 94.19% phone accuracy which significantly outperform previous approaches for Thai.

Index Terms: Thai, grapheme-to-phoneme conversion, example-based, letter-to-sound rules, pronunciation

1. Introduction

A pronunciation dictionary is a crucial component for speech synthesis and automatic speech recognition systems. To manually build large dictionaries, it requires language knowledge and is a time consuming task. Moreover it is impossible to create pronunciations for all words in a language on the fly.

To handle this problem, several G2P systems have been proposed such as decision-based tree [1, 2], statistically-based [3, 4] and pronunciation-by-analogy based (PbA) [5, 6 ,7] approaches.

For Thai, one of the best G2P systems was proposed by Tasaku et.al. [8] who proposed a Probabilistic GLR (PGLR) parser technique which gave 72.87% word accuracy. The PGLR parser is a rule-based approach. Thus the accuracy of this system depends on how much language knowledge and effort was given to the system. Chotimongkol and Black [2] proposed decision trees with combining n-gram of phone model. The best result was 75.3% word accuracy which improved about 6.4% absolute on word accuracy from the original letter-to-sound rules in Festival [1].

In this paper, we propose an Example-Based Grapheme-to-Phoneme conversion approach (EBG2P) which is a data-driven technique. This approach generates the pronunciation of a word by modifying and combining pronunciations of words or subwords from a training set.

2. Challenges in Thai G2P

General details about Thai pronunciation can be found in [9, 10]. Thai, an alphabetical language, has 44 characters for 21 consonant sounds, 19 characters for 24 vowel sounds (9 short vowels, 9 long vowels and 6 diphthongs), 4 characters for tone markers (5 tones), special characters, and numbers. Thai has some characteristics that cause problems in G2P system which can be classified as the following:

1. Some vowel letters can appear before, after, above or below a consonant letter e.g. in the word "แมว" (/mae:w/), the vowel character "แ" (/ae:/) appears before the consonant character "ม" (/m/)
2. Some syllables are composed of only consonants, or a single consonant letters without any written vowel symbols e.g. the word "ยก" (/yok/). It consists of only two consonant letters, "ย" (/y/) and "ก" (/k/). There is no letter to represent the vowel /o/.
3. The special character " ์ " called Karan is a deletion character. If it appears above a consonant, it indicates that the consonant will be ignored. Sometimes it does not only delete that consonant but it also deletes the one immediately preceding it or a whole syllable.
4. There is a syllable consisting of a consonant and a vowel which is represented with two or more vowel letters. The consonant letter is inserted among vowel letters. For example, in the syllable "และ" (/lae/) two vowel letters "แ" and "ะ" represent vowel /ae/ and the consonant letter "ล" (/l/) is in the middle.

3. Example-Based G2P Conversion

3.1. Training

First we segment all words from training data into syllables either by a rule-based syllable (RBS) segmenter or a statistically-based syllable (SBS) segmenter. We then create alignments between letters and phones (see 4.2 for the details). From the alignments, we save the G2P mapping statistics of each letter regarding its context (up to +/-1 letter).

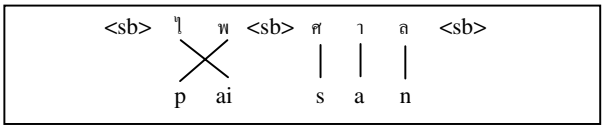

Figure 1: The alignments of letters and sounds

For example, the word "ไพศาล" (/paisan/) can be separated into two syllables, "ไพ"(/pai/) and "ศาล"(/san/). Figure 1 shows the alignments between letters and phones where <sb> is a syllable boundary. From the alignments, we then construct G2P mapping table by adding the following:

- Add the mapping from "ไ" to /ai/

i. where the previous letter is "<sb>" and the next letter is "พ".
ii. where the pervious letter is <sb>.
iii. where the next letter is "พ".

Then we do the same for every character. We also save the count of every mapping.

3.2. Pronunciation Generation

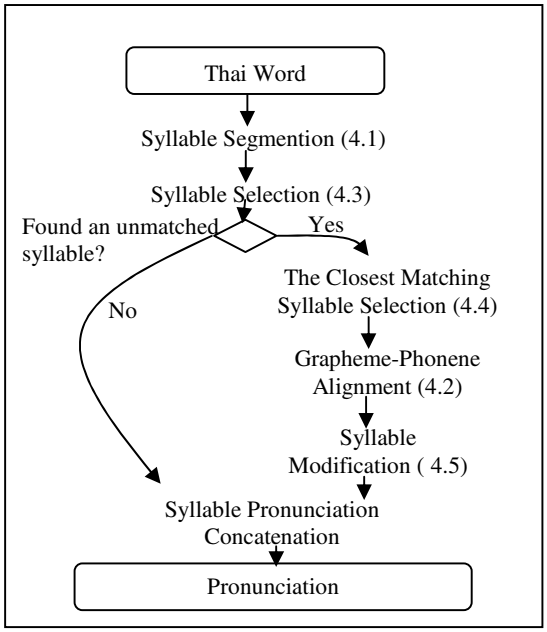

Figure 2: Pronunciation Generation in EBG2P

After the training step, all words in the dictionary were separated into syllables and the G2P mapping table was constructed. We then apply them to predict a pronunciation of a word in question by the following steps (see the Figure 2).

1.) First we have to segment the word in question into syllables by applying the same segmenter as used in the training steps. For example, we would like to predict a pronunciation of "ร้านขายยา" (/raan khaay yaa/). First we separate this word into syllables which are "ร้าน <sb> ขาย <sb> ยา".

2.) Then the longest matching subsequence of these syllables will be selected from the training corpus. For example, if there are the words I.) "ร้าน<sb>ค้า" (/raan khaa/) II.) "ขาย" (/khaay/) and III.) "ขาย<sb>ยา" (/khaay yaa/) in the training corpus, the first and third syllable of the pronunciation will be selected from the word I and word III, respectively. While the second syllable of the pronunciation could be selected from the word II or word III. In this case, the algorithm will choose the first syllable in word I and both syllables from words III.

3.) If every syllable in the source word is matched with a syllable in the training data, the algorithm then retrieves a pronunciation of each matching syllable. If a syllable can be pronounced more than one way, we select the one that has the highest frequency in the training data. Figure 3 shows the pronunciation generation for the word "ร้านขายยา" (/raan khaay yaa/).

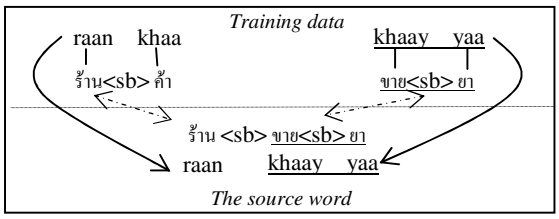

Figure 3: The pronunciation generation (in case all syllables found in the training data)

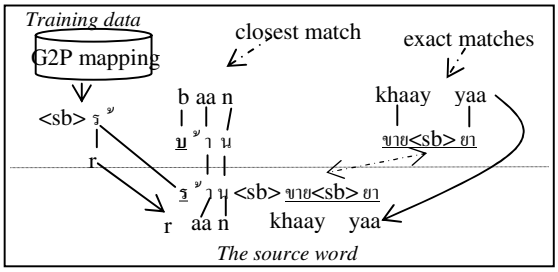

Figure 4: The pronunciation generation (in case a syllable is not found in the training data)

4.) If there is an unmatched syllable in the word, the algorithm will search for the closest matching syllable which has the lowest editing distance and then generate a pronunciation of the unmatched syllable by modifing the pronuciation of the closest matching syllable. In case there are several matching syllbles, we select the one that has the highest freqency in the training data.

For example, if the syllable "ร้าน" (/raan/) is not found in the training data, the algorithm will select the closest one. In this case, it could be "บ้าน" (/bann/). Then the algorithm generates the alignments between the letters and phones. To generate, a pronunciation of "ร้าน", the algorithm replaces the letter "บ" (/b/) with "ร" (/r/) and also replaces the phone of "บ" with the phone of "ร". The phone of "ร" is the one that has the highest frequency selected from the G2P mapping table regarding to its context (+/-1 letter). Figure 4 shows the pronunciation generation of "ร้านขายยา" (/rann kaay yaa/) when the syllable "ร้าน" (/rann/) is not found in the training data.

5.) Finally the algorithm combines the pronunciation of all syllables together. The result is the pronunciation of the source word.

4. EBG2P Components

In this section, we will describe the components of our EBG2P system in more detail: 1.) Syllable segmentation, 2.) Grapheme-phoneme alignment, 3.) Syllable selection 4.) The closest matching syllable selection and 4.) Syllable modification.

4.1. Syllable Segmentation

Our approach requires syllable boundaries in the training and test data. To produce syllable boundaries, we investigated two different approaches. The first is a rule-based approach that uses 43 hand-crafted rules described in [11]. It can achieve about 98% accuracy. However this requires language knowledge. We

therefore investigated a language independent technique to automatically segment words into syllables without supervised training data.

First we collect all substring frequencies from the training data. Then we use entropy information to predict the syllable boundaries using the following criteria.

We define right conditional entropy information which is shown in equation 1.

$$RE(y) = -\sum_{\forall z \in A} p(z \mid y) * \log_2 p(z \mid y) \quad (1)$$

The variable "y" is a substring in a word and "z" is the next letter after the substring "y", and A is the set of all letters in the language.

Applying the right conditional entropy to syllable segmentation is simple and straightforward. The algorithm will produced a syllable boundary when the $RE(y)$ decreases or when $RE(y)$ is 0. Figure 5 shows the syllable segmentation algorithm where S_{ij} is a substring in a word starting at the position I and ending at the position j, C_j is the character at position j in a word and $RE(S_{ij})$ is the right conditional entropy of a substring S_{ij}.

4.2. Grapheme-Phoneme Alignment

The purposes of aligning graphemes to phonemes are 1.) to construct statistics of converting graphemes to phonemes and 2.) to generate a pronunciation of an unseen syllable from the training syllables. To generate grapheme-phoneme alignments, we first manually construct a mapping table which allows all possible alignments of letters and phones which is the same technique that used in [1]. We additionally extend the alignment algorithm to handle cross alignments between consonants and vowels. Then we apply from the mapping table to align letters and phones by the longest matching technique.

The longest matching technique will first select the one that has the most number of phones first. For example, the word "สบาย" can be pronounced /sabaay/. The fist letter "ส" can be mapped to /s/ or /sa/. In this technique, it will select the longest one first which is /sa/ and then do the same for the rest of letters. So the results will be "ส(/sa/)", "บ(/b/)", "า(/aa/)", and "ย(/y/)".

4.3. Syllable Selection

To select a pronunciation of each syllable of a word in question, the algorithm tries to match syllables of the word in question with syllables in the training data. The longest matching syllables will be selected first. For example, suppose we try to generate a pronunciation of a word **"คุณภาพชีวิต" (/khun na phab chee wit/)** which could be separated into **"คุณ (/khun na/) ภาพ(/phab/) ชี(chee/) วิต(/wit/)"**. The algorithm will retrieve pronunciations of the all syllables from training data. Suppose there are four words in training data which are

I.) "คุณ(/khun/) ตา(/taa/)",
II.) "คุณ(/khun na/) ภาพ (/phab/)",
III.) "ภาพ (/phab/) วาด (waat)", and
IV.) "ชี(/chee/) วิต(/wit/)".

```
Len = length of the word
i=0;j=1;
  while (j < len){
     if (RE(S_ij) > RE(S_ij-1){
        if (RE(S_ij+1) < RE(S_ij)) {
           S_ij is a syllable
             i=j+1; j=i+1;
             continue;
          }
     }else if (RE(Sij)==0){
          if (j<len){
            S_ij-1 is a syllable;
            i=j; j=i+1;
             continue;
        }
     }
     j++;
 }
```

Figure 5: The Syllable Segmentation Algorithm

The first syllable ("คุณ") of the source word can be selected from the word I or II. However in word II, the next syllable also matches with the second syllable of the source word. The algorithm then selects the longest matching syllables which are in the first two syllables in the word II. Finally the algorithm matches the other two syllables of the source word with both syllables in the word IV. Therefore the pronunciation will be **/khun na phab chee wit/** which is correct.

The advantage of choosing the longest matching syllables is that it can retrieve an appropriate pronunciation which is better than selecting from a shorter one. From the previous example, if we select the first syllable from I , the second syllable from III and the rest from IV, the pronunciation of the source word will be **/khun phab chee wit/** which is not correct.

4.4. The Closest Matching Syllable Selection

In case a syllable of the word in question is not found in the training database, the algorithm will select a syllable which is the most similar to that syllable. We apply an editing distance to measure a similarity between two syllables. Only a substitution operation is used in calculation. The cost of a substitution depends on whether a letter replaced by a letter from a different or the same class. If it is replaced by a letter from the same class, the cost of substitution for each letter is 0.5. Otherwise it is 1. In this paper, we classify letters into three classes: 1.) Consonants 2.) Vowels and 3.) Tone markers.

4.5. Syllable Modification

After we obtained the closest matching syllable, we retrieve its pronunciation and then we make alignments between the letters and phones. We modify this pronunciation by substituting a phone of an unmatched letter in the source syllable. This phone is selected from the G2P mapping table regarding to its context.

For example, suppose we try to generate a pronunciation of the syllable "เขา" (/**kh**aw/) from the syllable "เรา" (/**r**aw/). First we have to make alignments of letters and phones of the syllable "เรา" which are "เ(aw) ร(r) า(-)" and then transform the syllable "เรา" to "เขา" by substituting the letter "ร" with "ข". Also the phone aligned with "ร" has to change to a phone of "ข". To select the phone of "ข", we look up in the G2P mapping table.

Example-Based G2P				Letter-to-Sound Rules in Festival (LTS)			
RBS segmenter		SBS segmenter		With reordering rules		Without reordering rules	
Phone (%)	Word (%)	Phone (%)	Word (%)	Phone (%)	Word (%)	Phone (%)	Word (%)
94.19	80.99	85.29	65.32	86.30	58.80	84.52	56.70

Table 1: The accuracies of different approaches

From the table we will get that letter "ข" in between "เ" and "า" is mostly mapped to /kh/. Finally we will substitute /r/ in /raw/ with /kh/. So a pronunciation of the syllable "เขา" will be /khaw/.

5. Experiments

For our experiments, we used a pronunciation dictionary of about 10,000 words. It was initially generated by LTS and then manually edited by linguists. To evaluate our algorithm, we randomly split the dictionary into two parts. The first part is about 90% of the data for training and the rest is for testing.

Because Thai writing system allows vowel letters precede consonant letters and the alignment technique in LTS does not support cross alignments. We then applied about 10 simple heuristic rules to reorder vowels and consonants. Comparing the performance between EBG2P and LTS, the EBG2P with SBS segmenter greatly outperforms the LTS without applying reodering rules on word accuracy. While EBG2P with RBS segmenter produced much better results than LTS with applying reodering rules on both phone and word accuracies.

When comparing the results of EBG2P using two different syllable segmenters, the EBG2P with RBS segmenter outperforms the EBG2P with SBS segmenter about 9% absolute on phone accuracy and about 16% absolute on word accuracy. These show that the syllable information has an enormous influence on this approach. Thus it is worth to compare the accuracy between two syllable segmenters

We then compare the accuracy of the SBS segmenter with the RBS segmenter in term of precisions and recalls. On the average of several training and test tests, the precision is about 81% and the recall is about 63%. Moreover unseen syllable rates in the test set using the SBS and RBS segmenter are 19.25% and 10.38%, respectively. The analysis of the results shows that the SBS segmenter usually generates longer syllables than the RBS segmenter. Also the entropy information inefficiently predicts syllable boundaries when a syllable occurs only once or very low frequency in the training corpus. Therefore, we have to improve the SBS segmenter and the SBS segmenter also needs more training data to reduce the unseen syllable rates.

6. Conclusions

From the experiment results, they show that our EBG2P technique produce impressive results and outperform decision trees approaches. However, our algorithm needs to know syllable boundaries to give the best performance. From the results, the EBG2P accuracies decrease a lot when using the statistically-based syllable segmenter but it is still comparable to Festival on phone accuracies and produce better results on word accuracies. Furthermore it would be interesting to apply syllable information on the LTS in Festival.

The final targets of this approach are to make the EBG2P approach fully portable to another language and to automatically extract all useful information from the training data. Currently this is the first state of developing EBG2P. For the next steps, we plan to generate all important information automatically from the training data: 1.) a mapping table, 2.) syllable segmentation, and 3.) the similarity cost table.

7. Acknowledgements

We would like to thank Alan Black and Ananlada Chotimongkol for providing us the scripts to do Thai G2P in Festival, and Sanjika Hewavitharana for the useful comments.

8. References

[1] Alan W Black, Kevin Lenzo. and Vincent Pagel. Issues in Building General Letter to Sound Rules. The 3rd ESCA Workshop on Speech Synthesis, Jenolan Caves, Australia. 1998.

[2] Ananlada Chotimongkol and Alan W Black. Statistically trained orthographic to sound Models for Thai, In Proceedings of ICSLP 2000.

[3] Stanley F. Chen. Conditional and Joint Models for Grapheme-to-Phoneme Conversion. The 8th Eurospeech. Geneva, Switzerland. 2003.

[4] Paul Taylor. Hidden Markov Models for Grapheme to Phoneme Conversion. In Proceedings of Interspeech 2005, Lisbon, Portugal.

[5] Michale J. Dedina and Howard C. Nusbaum. PRONOUNCE: a program for pronunciation by analogy. Computer Speech and Language 5 , 55--64. 1996.

[6] Francois Yvon. Grapheme-to-Phoneme conversion using multiple unbounded overlapping chunks. In Proceedings of NeM-LaP-2, Ankara, Turkey. 1996.

[7] Yannick Marchand and Robert I. Damper. A Multi-Strategy Approach to Improving Pronunciation by Analogy. Computational Linguistics 26(2). 2000.

[8] Pongthai Tarsaku, Virach Sornlertlamvanich and Rachod Tongprasert. Thai Grapheme-to-Phoneme Using Probabilistic GLR Parser. In Proceedings of Eurospeech Aalborg, Denmark. 2001.

[9] Virongrong Tesprasit, Paisarn Charoenpornsawat and Virach Sornlertlamvanich. A Context-Sensitive Homograph Disambiguation in Thai Text-to-Speech Synthesis. In Proceedings of HLT-NAACL 2003, Edmonton, Canada. 2003.

[10] Wirote Aroonmanakun and Wanchai Rivepiboon. A Unified Model of Thai Word Segmentation and Romanization. In Proceedings of The 18th PACLIC, Tokyo, Japan. 2004.

[11] Doungkaew Sawamipak. Developing software for analyzing Thai syntax in Unix. Thamasart University Publishing, Bangkok, Thailand. 1990.