GENERATING THAI TRANSCRIPTIONS FOR ENGLISH WORDS

Wirote Aroonmanakun,
Nuttakorn Thapthong, Pakaket Wattuya,
Benjawan Kasisopa, and Sudaporn Luksaneeyanawin
Centre for Research in Speech and Language Processing,
Chulalongkorn University
<*awirote@chula.ac.th*>

Abstract
In a text-to-speech system, a transcription of each word can be either retrieved from the dictionary, or generated by rules or some statistical means. Though the dictionary-based approach can produce the most accurate result, a letter-to-sound conversion module is still necessary for unknown words. This study focuses on producing a module that can automatically transcribe English words into Thai sounds. To do this, 18,690 samples of English words are extracted from the CMU pronunciation dictionary. These samples are classified into different groups. Each group is used for extracting mapping rules for each Thai sound. A machine learning algorithm is implemented to extract conversion rules from these samples. These rules map English letter(s) into a corresponding Thai sound in a specified context. In other words, conversion rules are context sensitive rules, of which the maximum left and right contexts are two letters. Conversion rules are implemented as lookup tables like those of Bosch and Daelemans' (1993). Rules extracted by the algorithm are manually examined and grouped together to make them more generalized. As a result, the number of rules is reduced from 4,120 to 440 rules.

In addition, since Thai syllable structures are different from that of English, phoneme sequences produced by the conversion rules have to be adjusted to comply with the Thai phonological system. Phoneme sequences that are not possible in Thai will be changed, deleted, or split into syllables with respect to the Thai syllable structures. Tones are also assigned to each syllable. Because this grapheme-to-phoneme module is going to be used for generating transcriptions of English words that are not in the dictionary, the module is tested on 1,475 English proper names. The results were judged in terms of degrees of acceptability, adopted from Coker et al.(1990), namely "good", "fair", and "poor",. "Good" means the transcription is what the judge would have said. "Fair" means the transcription sounds all right even the judge would not have said it that way. "Poor" means the transcription sounds bad; no one would have said it. The module can produce the output that is acceptable (fair or good) at the rate of 44%. Though the accuracy rate is not very high, the module is sufficient for generating transcriptions of unknown words.

Introduction

In a text-to-speech system, transcriptions of texts have to be produced by some means. The transcriptions can be generated by letter-to-sound conversion rules, which can be manually written (e.g. Chotimongkol and Black 2000) or automatically extracted from training data (e.g. Bosch and Daelemans 1993). Or they can be generated by using statistical models of

Wilaiwan Khanittanan & Paul Sidwell, eds. *SEALSXIV: papers from the 14th meeting of the Southeast Asian Linguistics Society (2004), Volume 1*. Canberra, Pacific Linguistics, 2008, pp.13-22.

grapheme-to-phoneme conversion (e.g. Tarsaku et al. 2001). With this statistical method, the transcription is the phoneme sequence which has the highest probability. The transcriptions could also be retrieved from a dictionary, in which transcription of each word is manually stored in the dictionary (e.g. Luksaneeyanawin 1989). Though the dictionary-based approach takes longer time to be developed and uses more system resources, it is usually the most accurate one. Nevertheless, it still has to face with unknown word problems, especially for those proper names. Thus, even in a dictionary-based text-to-speech system, there must be a module to handle unknown words.

Since English words are often found within the Thai างกันระหว่ texts, e.g. "ความแตกต่นน้ำงบริษทรถยนต์ Chrysler และ General Motors โดยพิ้วเป็นฐาน แล้นมายาภาพ", a Thai text-to-speech system has to produce not only transcriptions of Thai texts, but also transcriptions of English words. It is possible to use an existing English text-to-speech system to handle only those English words, while a Thai text-to-speech system is used for Thai words. But the speech produced by this method might sound unnatural because users would not expect to hear an English accent mixed within Thai speeches. A more natural solution is to produce transcriptions of English words on the basis of Thai phonological system. In this study, we aim to build a grapheme-to-phoneme conversion module that automatically transcribes English words into Thai sounds.

Transcriptions of English and Thai

Our Thai text-to-speech system, CU-TTS, is mainly a dictionary-based system. Transcriptions of Thai and English words are manually stored in the dictionary. Transcriptions of both English and Thai are based on the Thai phonological system. But some phonemes like /s/, /f/, /l/, /ch/ are allowed as possible final consonants. Transcriptions of most words would be retrieved from the dictionary. Only transcriptions of words that are not included in the dictionary will be generated by rules. To handle unseen English words, an English grapheme-to-phoneme system is developed. The system here is different from other English grapheme-to-phoneme systems because the transcriptions produced by our system are Thai transcriptions, not English transcriptions. For example, the word "bird" will be pronounced /bəət3/ rather than /'bɜd/.

Since the set of English phonemes are different from Thais, we have to map English phonemes to corresponding Thai phonemes. English transcriptions that already coded in the Carnegie Mellon University Pronouncing Dictionary are used as the basis of English phonemes in this study. (The CMU Dictionary is a machine-readable pronunciation dictionary for North American English that contains over 125,000 words and their transcriptions.) In CMU dictionary, 39 phonemes are used for coding the English transcription (not counting variation for lexical stress). Vowels may carry lexical stress, marked with the number 0-2. (0 = No stress, 1 = Primary stress, 2 = Secondary stress) For example, the word "homework" is transcribed as /HH OW1 M W ER2 K/, "coordinate"as /K OW0 AO1 D IH0 N EY2 T/. Table 1 shows the mapping of English phonemes to Thai phonemes. The second column represents the CMU code for English phonemes. The third column, "IPA", is the IPA symbols of that English phoneme.

The next column, "Thai Pronunciation", is the corresponding Thai phoneme. For some phonemes, a phoneme with or without stress may map to different phonemes in Thai. For example, AA0 (unstressed /ɔ/) maps to /ɔ/ in Thai, while AA1 (stressed /ɔ/) maps to

/aa/ in Thai. Some phonemes, such as B, D, T, map to two different Thai phonemes according to its position in the syllable structure (as an initial or final consonant).

Table 1: *Mapping between English and Thai phonemes*

Phoneme #	CMU	IPA	Thai Pronunciation	Example
1	AA0	ɔ	ɔ	Cod
	AA1	ɑ ▪	aa	Heart
2	AE	æ	ɛ	Bad
3	AH0	ə	ə	About
	AH1	ʌ	a	Bud
4	AO	ɔ ▪	ɔɔ	Cord
5	AW	au	aw	Cow
6	AY	ai	aj	Eye
7	EH	e	e	Bed
	EH0 (R)	æ ▪	ɛɛ	Bare
8	ER	ɜ	əə	Bird
9	EY	e ▪	ee	Day
10	IH	ɪ	i	Bid
	IH0 (R)	ia	ia	Beer
11	IY	i ▪	ii	Bead
12	OW	o ▪	oo	Go
13	OY	ɔi	ɔj	Boy
14	UH	ʊ	u	Good
	UH0	ua	uua	Tour
15	UW	u	uu	Food
16	B	b	#b, p#	Be, Cab
17	CH	tʃ	ch	Etch
18	D	d	#d, t#	Dog, Mad
19	DH	ð	th	Then
20	F	f	f	Fee
21	G	g	k	Green
22	HH	h	h	He
23	JH	dʒ	c	Edge
24	K	k	kh, k	Key, Wok
25	L	l	l	Lab
26	M	m	m	Me

27	N	n	n	No
28	NG	ŋ	ŋ	Sing
19	DH	ð	th	Then
29	P	p	ph, p	Put, Cap
30	R	r	r	Read
31	S	s	s	Sea
32	SH	ʃ	ch	Shed
33	T	t	th, t	Tea, Sit
34	TH	ɵ	th	Thin
35	V	v	w	Van
36	W	w	w	Way
37	Y	j	j	Yard
38	Z	z	s	Zoo
39	ZH	ʒ	ch	Beige

A transcription of every word in the English dictionary then is manually coded using Thai phonemes. The transcription is syllable-segmented and tone-assigned. More than 100,000 words are stored in our English dictionary. Although the process of compiling this English dictionary is time-consuming, it would yield the most accurate transcriptions. When our Thai text-to-speech system processes English words, it will retrieve Thai transcriptions of those words from the dictionary. However, the dictionary cannot contain all possible words, especially for those proper names. A module to convert English words to Thai transcriptions is still necessary for the system.

Grapheme-to-phoneme Conversion

Grapheme-to-phoneme conversion is a necessary part of any text-to-speech systems. Various approaches have been proposed. It could be a dictionary-based, a rule-based, a statistical based, or a hybrid one. In a rule-based system, letter-to-sound conversion rules can be constructed by hand or derived by a machine. In a statistical based system, a corpus of word-transcription is used for training. Probability of mapping each character to its corresponding phoneme is estimated from this training corpus. The system will use this statistical information to select the transcription with the highest probability. Most of current systems are statistical basis because it is easier to construct and maintain the systems. In addition, the systems do not depend on a specific language as the rule-based systems do. They can be used for any languages as long as an appropriate training corpus is given. Nevertheless, the rule-based systems usually have an advantage in terms of processing speed.

In our Thai text-to-speech system, since most of the transcriptions are in the dictionary, we do not need the grapheme-to-phoneme conversion module to transcribe all words. The module we need will be used to generate transcriptions of a few unknown words. Thus, we do not want a module to be complicated and take much processing time. In this study, we adapted Bosch and Daelemans' (1993) data-oriented rule based grapheme-to-phoneme conversion system. Bosch and Daelemans used a training corpus

which contains pairs of words and transcriptions. In each pair, each character is aligned with its corresponding sound. A learning algorithm will read this corpus and generate context-sensitive letter-to-sound conversion rules. Bosch and Daelemans set the maximum context to be five characters on the left and on the right. For example, in Dutch, when the pair "s|c|h|o|e|n|e|n"-/s|x|#|u|#|n|#|ɔ|#/ is found, rules like s|c|ho -> /x/, sc|h|oen -> #, etc., will be generated. (The symbol # is used for phonetic null. It means no sound is generated from the mapping.) The letter in between | | is the one that maps to the specified sound. Letters before and after | are those left and right contexts. Rules will cease to exist if it conflicts with new data. (The same character in the identical context maps to a different sound.) Thus, rules are not ambiguous. At the end, rules with different contexts sizes are stored in different lookup tables. For example, Table r:0-1-1 is the collection of mapping rules when considering only one character on the right; Table r:2-1-3 is the collection of mapping rules when considering two characters on the left and three characters on the right. Probabilistic rules are also generated from the training data, e.g. a +> /aa/. (The symbol +> is used for probabilistic mapping rules.) Probabilistic rules are chosen from the most frequent mapping of that character. These probabilistic rules will be used when no rules can map the input character. Bosch and Daelemans (1993) randomly selected 18,500 word-pronunciation pairs as the training data for Dutch, and for English. The lookup tables created from the training data contains 27,000 rules or patterns for Ducth, and 35,000 patterns for English.

In our system, since we already built a dictionary which contains pairs of English words and Thai transcriptions, we could use these data as a training set. But, we would need to do the alignment between letters and sounds first. Doing that would be very time-consuming. Since we only need a small module for processing unseen words, we build lookup tables in a much quicker way by using training data that is different from that of Bosch and Daelemans. Because we expect the difficulties of mapping from English characters to Thai sounds are mainly caused from vowels, we randomly selected pairs of words and transcriptions as representatives of each vowel. Table 2 shows the number of words selected for training each vowel pattern. There are totally 20 data sets. For each pair, we only mark characters that map to the selected vowel in that data set. For example, since the character "e" in "payment" is mapped to /e/, this word will be marked as "paym(e)nt". Thus, training data for the sound /e/ will be like "paym(e)nt", "pass(a)ge", "overh(ea)d", etc. Then, a learning algorithm will be used to create mapping rules of that vowel. The program will create mapping rules with different contexts, such as m|e|n -> /e/, ym|e|nt -> /e/, etc. In this study, we set the maximum contexts to be two characters. Only 7 lookup tables and 2 probabilistic mapping tables are used in this study, namely r:0-1-0, r:0-1-1, r:1-1-0, r:1-1-1, r:2-1-0, r:0-1-2, r:2-1-2, g:0-1-0, and g:1-1-1. Table r:x-1-y stands for lookup table in which x characters on the left and z characters on the right are taken into account, and g:x-1-y stands for probabilistic mapping when considering x characters on the left and y characters on the right.

Table 2: *Number of training words for each vowel.*

CMU	IPA	Thai Pronunciation	No. of data
AA0	ɔ	ɔ	485
AA1,AA2	ɑː	aa	902
AE	æ	ɛ	970
AH0	ə	ə	2082
AH1	ʌ	a	439
AO	ɔː	ɔɔ	1167
AW	au	aw	291
AY	ai	aj	285
EH	e	e	1567
EH0 (R)	æː	ɛɛ	82
ER	ɜ	əə	1915
EY	eː	ee	761
IH	ɪ	i	1610
IH0 (R)	ia	ia	220
IY	iː	ii	1397
OW	oː	oo	4182
OY	ɔi	ɔj	46
UH	ʊ	u	76
UH0	ua	uua	17
UW	u	uu	196

To reduce the number of rules and ensure that rules are not accidental, we select only rules or patterns that occur more than once in the training data. There are 4,120 rules at this step (step-1). Next, we delete rules that could also be produced by the default mappings (probabilistic rules g:0-1-0). For example, since the default mapping of the letter "a" is /ɛ/, we can safely delete all rules that map "a" to /ɛ/ in all lookup tables. After deleting rules that have the same output as the default mapping of "a", "e", "i", "o", "u" (step-2), the number of rules reduces to 2,549 patterns. At this point, we only have mapping rules for vowels. Thus, we manually add 62 default mapping rules for consonant letters (not a,e,i,o,u) (step-3). Then, we delete rules that are redundant. These are mapping rules that produce the same sound as the probabilistic mapping rules g:1-1-1 (step-4). For example, if we have r|i|l +> /i/, we could delete rules like rr|i|ll -> /i/, er|i|ll -> /i/, etc. At this step, the number of rules reduces to 1,548 patterns. After that, rules are manually deleted, or modified to reflect the generalization as much as possible (step-5). Special symbols, C, V, and X are used to represent groups of consonant letters, vowel letters, and

any letters respectively. The final outcome contains 440 patterns for mapping (including probabilistic mapping).

These mapping rules will be used by a grapheme-to-phoneme module. The module will try rules in the mapping tables in the following order: r:2-1-2, r:2-1-0, r:0-1-2, r:1-1-1, r:1-1-0, r:0-1-1, r:0-1-0, g:1-1-1, and g:0-1-0. Each character in the unknown word will be mapped to a corresponding sound, with respect to the patterns. Since we did not create the mapping rules for consonants in the same way as we did with vowels (by learning from training data), it is possible that the output might have a cluster that is not possible for Thai, such as "st", "bj", "nj", "mj", etc. For example, the program will map the word "star" to /sthaa/ by using the following rules s+>/s/, t+>/th/, C|a|r+>/aa/, V|r|^ -> #. (The symbol "^" is used for indicating word boundary.) But this phoneme output is not Thai pronunciation. Thus, the program has to readjust sound sequence by means of deletion, insertion, or modification. In this example, /sthaa/ is changed to /sa taa/. Then, each syllable will be assigned tones. (The numbers 0-4 are used for mid, low, falling, high, and rising tones respectively.)

From the Thai transcriptions encoded in the English dictionary, we tried to extract tone patterns of words with different number of syllables, e.g. 2-syllable words, 3-syllable words, etc. But there seem to be no unique tone sequence for each word type. For example, for 2-syllable words, the tone sequence can be 00 ("begin"-/bii0 kin0/), 01 ("import"-/?im0 pɔɔt1/), 02 ("bonny"-/bɔɔn0 nii2/), 03 ("aircraft"-/?ɛɛ0 craaf3/), 10 ("smile"-/sa1 maaj0/), 11 ("bishop"-/bi1 chɔɔp1/), 12 ("fader"-/feet1 dəə2/, 20 ("vietnam"-/wiiat2 naam0/, 21 ("weirdness"-/wiiat2 nees1/, 30 ("workday"-/wəək3 dee0/), 31 ("abbot"-/?ɛp3 bot1/), 32 ("beatle"-/bii3 thəl2/), 33 ("accept"-/?ɛk3 sep3/, etc. However, we create rules of tone assignment by frequency of occurrences. Tone patterns that occur most often are used as the default one. By doing this, though the program cannot assign correct tones to all words, the tone assigned should be acceptable for a large number of words. The following is the tone assignment rule for 2-syllable words.

Live + Live & Short vowel => 0 2
Live + Live & Long vowel => 0 0
Live + Dead & Short vowel => 0 1
Live + Dead & Long vowel => 0 2
Dead + Live => 3 2
Dead + Dead & Short vowel => 3 1
Dead + Dead & Long vowel =>3 2

There are seven tone patterns for 2-syllable words. Tones are assigned on the basis of live/dead syllables and short/long vowels. A syllable is a dead one if it ends with one of the following sounds, /p/, /t/, /k/, /f/, /s/, /ch/; or it ends with a short vowel. A syllable is a live syllable if it ends with the following sound /m/, /n/, /ŋ/, /w/, /j/, /l/; or it ends with a long vowel. For example, if the first syllable is a live syllable, its tone will be 0. If the next syllable is also a live one with short vowel, the tone will be 2. Tone assignment rules for 3 or 4syllable words are also created in the similar way.

Experiments

Since the English grapheme-to-phoneme module is designed to generate Thai transcriptions of any unknown words, which are usually proper names, the module is tested on 1,475 English proper names. The transcription output is expected to be fairly acceptable rather than perfect. Therefore, we evaluate the performance of the module by judging the results in terms of degrees of acceptability, adapted from Coker et al.(1990), namely "good", "fair", and "poor". The label "good" means the transcription is what the judge would have said. "Fair" means the transcription sounds all right even the judge would not have said it that way. "Poor" means the transcription sounds bad; no one would have said it. A graduate student in the Linguistics program is asked to judge the result using this scale. For example, the transcription /ʔɛɛ0 leek1 sɛɛn0 driia2/ generated for the word "Alexandria" was judged as "poor"; /khris3 thii2/ generated for "Christie" was judged as "fair"; /baa0 baa0 raa2/ generated for "Barbara" was judged as "good". Table 3 shows the number of items judged.

Table 3: *The result of judgment*

Good	417	28.27%
Fair	237	16.07%
Poor	821	55.66%
Total	1475	100.00%

It can be seen from Table 3 that 44% of the outputs are acceptable (fair or good). But 56% are not acceptable. Since generating appropriate tones is a difficult task and we want to evaluate the letter-to-sound conversion rules, the judge was asked to re-evaluate the result again by ignoring tones. The result from this judgment would better reflect the performance of the English grapheme-to-phoneme module. The result is shown in Table 4.

Table 4 shows that the result is a bit better. The number of acceptable phoneme sequences is up to 55%, while the number of unacceptable is 45%. Though the accuracy is not up to the level of 80% as reported in other systems, the module is sufficient to the task in this study, given that pronunciations of 44% of unseen words are acceptable, and the rests are at least Thai pronunciations.

Table 4: *The result of judgment when ignoring tones*

Good	537	36.41%
Fair	278	18.85%
Poor	660	44.75%
Total	1475	100.00%

Discussion

This study has demonstrated a quick and easy way to create a grapheme-to-phoneme conversion system. Instead of using a corpus which is fully aligned between characters and sounds, we prepared training data only on the difficult mappings, which are vowels in this study. Alignment is marked only for characters that produce those vowel sounds. Letter-to-sound conversion rules then are automatically generated by a machine. Rule patterns

generated from the machine can immediately be used by the conversion program. However, we chose to manually collapse rules by hand since we want the module to be very small. Compared to previous research on English grapheme-to-phoneme conversion, which has the number of rules up to 35,000 patterns (Bosch and Daelemans 1993), the number of rules in this study is only 440 patterns. Though the accuracy of the module is not high, it is sufficient to the task. Because most of English words' transcriptions are already stored in the dictionary, we only want the module to generate Thai transcriptions for a few unknown words. These words are expected to be at least pronounceable in Thai. And if the pronunciation really sounds bad, users can add the correct pronunciation in the user dictionary.

Acknowledgments

This research is only a small part of the project "Text to Speech and Automatic Speech Recognition for Wireless and Wire Line Value Added Services", which is developed by the Centre for Research in Speech and Language Processing, Chulalongkorn University, for the Sun System Co.Ltd. in 2003.

References

Bosch, A. van den and W. Daelemans. 1993. Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, pages 45-53, Utrecht, Netherland.

Chotimongkol, Ananlada and Alan W Black. 2000. Statistically trained orthographic to sound Models for Thai, In *Proceedings of ICSLP 2000*, Beijing, China October 2000.

Coker, C., K. Church and M. Liberman, 1990. Morphology and Rhyming: Two Powerful Alternatives to Letter-to-Sound Rules for Speech Synthesis. In *Proceedings of the Conference on Speech Synthesis, European Speech Communication Association.*

Luksaneeyanawin, Sudaporn. 1989. A Thai Text to Speech System. In *Proceeding of the Conference of the Regional Workshops on Computer Processing of Asian Languages*, pages 305-315, Asian Institute of Technology.

Meknavin, Surapant and Boonserm Kijsirikul. 2000. Thai Grapheme-to-Phoneme Conversion. In Burn-ham, Denis, et.al., editors, *Interdisciplinary Approaches to Language Processing: The International Conference on Human and Machine Processing of Language and Speech.* NECTEC: Bangkok.

Tarsaku, Pongthai, Virach Sornlertlamvanich, and Rachod Thongpresirt. 2001. Thai Grapheme-to-Phoneme Using Probabilistic GLR Parser. In *Proceedings of Eurospeech 2001*, Aalborg, Denmark, Sept 2001.

Appendix : Examples of the result

Word	Transcription	Judgment1	Judgment2 (ignore tone)
Abraham	ʔɛɛ0^brɛɛ0^hɛɛm2^	poor	poor
Ackworth	ʔɛk3^khwɔɔt1^	good	good
Adam	ʔaa0^daam0^	good	good
Alan	ʔaa0^lɛɛn0^	good	good
Alexander	ʔaa0^leek1^sɛɛn0^dəə2^	fair	good
Alfred	ʔɛɛl0^freet1^	good	good
Baker	bee0^khəə0^	fair	good
Baldwin	bɔɔl0^wiin0^	fair	good
Blackwell	blɛk3^khweel2^	poor	poor
Buckingham	bak3^khiŋ0^hɛɛm2^	fair	good
Cameron	khee0^məə0^roon2^	fair	fair
Campbell	khɛɛm0^beel0^	good	good
Christie	khris3^thii2^	fair	fair
Christina	khris3^thii0^naa2^	good	good
Dennis	deen0^nis1^	good	good
Duncan	duun0^khɛɛn0^	poor	poor
Edmund	ʔeet3^muun2^	fair	fair
Edward	ʔeet3^wɔɔt1^	poor	poor
Fabian	fɛɛ0^biian0^	good	good
Fleetwood	fliit3^wuut1^	fair	good
Fleming	flee0^miŋ2^	fair	fair
Fortune	fɔɔ0^thuun0^	good	good
Geoffrey	cii0^ʔaa0^ʔoof3^free2^	poor	poor
Gifford	kif3^fɔɔt1^	good	good
Gilbert	kiil0^bəət1^	good	good
Harris	haa0^ris1^	poor	poor
Harte	haa0^thee0^	poor	poor
Harvard	haa0^waat1^	good	good
Mary	maa0^rii0^	poor	poor
Obson	ʔoop3^saan2^	poor	poor