Modified Grapheme Encoding and Phonemic Rule to Improve PNNR-Based Indonesian G2P

Suyanto
Department of Computer Science and Electronics, FMIPA
Gadjah Mada University
Yogyakarta, Indonesia 55281

Sri Hartati
Department of Computer Science and Electronics, FMIPA
Gadjah Mada University
Yogyakarta, Indonesia 55281

Agus Harjoko
Department of Computer Science and Electronics, FMIPA
Gadjah Mada University
Yogyakarta, Indonesia 55281

***Abstract*—A grapheme-to-phoneme conversion (G2P) is very important in both speech recognition and synthesis. The existing Indonesian G2P based on pseudo nearest neighbour rule (PNNR) has two drawbacks: the grapheme encoding does not adapt all Indonesian phonemic rules and the PNNR should select a best phoneme from all possible conversions even though they can be filtered by some phonemic rules. In this paper, a modified partial orthogonal binary grapheme encoding and a phonemic-based rule are proposed to improve the performance of PNNR-based Indonesian G2P. Evaluating on 5-fold cross-validation, contain 40K words to develop the model and 10K words to evaluation each, shows that both proposed concepts reduce the relative phoneme error rate (PER) by 13.07%. A more detail analysis shows the most errors are from grapheme ⟨e⟩ that can be dynamically converted into either /ɛ/ or /ə/ since four prefixes, 'ber', 'me', 'per', and 'ter', produce many ambiguous conversions with basic words and also from some similar compound words with both different pronunciations for the grapheme ⟨e⟩. A stemming procedure can be applied to reduce those errors.**

Keywords*—Modified grapheme encoding; phonemic rule; Indonesian grapheme-to-phoneme conversion; pseudo nearest neighbour rule

I. Introduction

A phonemization or letter-to-sound conversion, more commonly known as grapheme-to-phoneme conversion (G2P), is an important module in both speech recognition and speech synthesis. In general, a G2P is developed using machine learning-based methods, such as instance-based learning [1], table lookup with defaults [1], self-learning techniques [2], hidden Markov model [3], morphology and phoneme history [4], joint multigram models [5], conditional random fields [6], Kullback-Leibler divergence-based hidden Markov model [7]. These methods are commonly very complex and designed to be language independent, but they give varying performances for some phonemically complex languages, such as English, Dutch, French, and Germany.

In [8], an information gain tree (IG-tree) with best guest strategy is proposed to develop a specific G2P for Indonesian language. It gives PER of 0.99% for training set of 9K words and testing set of 1K unseen words. Another Indonesian G2P developed using PNNR [9] produces slightly higher PER than the IG-tree, around 1.07, but it is capable of disambiguating homograph words. The PNNR-based G2P has two drawbacks. Firstly, the designed grapheme encoding does not adapt all Indonesian phonemic rules. Secondly, the PNNR should select a best phoneme from all possible conversions even though they can be filtered using some phonemic rules.

When compare to English, the Indonesian language has a much simpler phonemization rule. An initial study on 50K words, collected from the great dictionary of the Indonesian language (*Kamus Besar Bahasa Indonesia Pusat Bahasa*or KBBI) third edition, released in 2008, developed by *Pusat Bahasa*, shows that it has 26 graphemes, where 15 graphemes (57%) are pronounced as certain phonemes without any exception and the rest 11 graphemes (43%) are pronounced as some possible phonemes with or without a quite common phonemic rule, as listed in table I. In this research, all graphemes are converted into single-phonemic symbol (SPS) to simplify the alignment process. The SPS are deliberately developed using characters available in any computer keyboard to simplify the implementation. But, in this paper any phoneme is written using the IPA symbol.

In table I, the grapheme ⟨a⟩ can be pronounced as four possible phonemes: 1) /ɑ/, generally if it is followed by graphemes those commonly pronounced as consonants, such as a word '*abad*' (century) that is pronounced as /ɑbɑd/; 2) /aɪ/, usually if it is followed by ⟨i⟩ or ⟨y⟩, such as '*abai*' (do not care) that is pronounced as /ɑbaɪ/; 3) a diphtong /aʊ/, commonly if it is followed by ⟨u⟩ or ⟨w⟩, such as '*harimau*' (tiger) that is pronounced as /hɑrimaʊ/; and 4) /ɑ+ʔ/, commonly if it is followed by ⟨a⟩, ⟨e⟩, ⟨i⟩, ⟨o⟩, or ⟨u⟩, such as in '*saat*' (sometime) that is pronounced as /sɑʔɑt/, and '*bait*' (couplet) that is pronounced as /bɑʔit/. The grapheme ⟨a⟩ is not possible to be pronounced as /aɪ/ if it is not followed by ⟨i⟩ nor ⟨y⟩. It also not possible to be pronounced as /aʊ/ if it is not followed by ⟨u⟩ nor ⟨w⟩. Such phonemic rules can be actually used to filter possible conversions so that PNNR can convert a grapheme into a correct phoneme more accurately and faster.

The grapheme ⟨a⟩ is much more frequently pronounced as /ɑ/, up to 54K, among the three other phonemes those are less than 1K. This fact makes ⟨a⟩ can be converted into a correct phoneme easily. But, grapheme ⟨e⟩ can be pronounced as five possible phonemes: /ɛ/, /ə/, /eɪ/, /ɛ+ʔ/, and /ə+ʔ/. The biggest challenge is converting this grapheme into /ɛ/ and /ə/ since it changes so dynamic without certain phonemic rule and they have high frequencies and many ambiguities. For

TABLE I: Twenty Six Indonesian Graphemes, Their Possible Pronunciations in the Single Phonemic Symbol (SPS) and the International Phonemic Alphabet (IPA) as well as Frequencies and Percentages in 50K words containing 385K Graphemes

Number	Grapheme	SPS	IPA	Frequency	Percentage
1	a	a	ɑ	54,859	14.23%
1	a	A	aɪ	979	0.25%
1	a	U	aʊ	624	0.16%
1	a	1	ɑ+ʔ	669	0.17%
2	b	b	b	11,236	2.91%
3	c	c	ʧ	3,667	0.95%
4	d	d	d	8,077	2.10%
5	e	e	ɛ	9,851	2.56%
5	e	E	ə	30,554	7.93%
5	e	Y	eɪ	29	0.01%
5	e	2	ɛ+ʔ	36	0.01%
5	e	3	ə+ʔ	193	0.05%
6	f	f	f	2,114	0.55%
7	g	g	g	6,492	1.68%
7	g	*	*	11,513	2.99%
8	h	h	h	5,769	1.50%
8	h	*	*	245	0.06%
9	i	i	i	26,685	6.92%
9	i	*	*	1,047	0.27%
9	i	4	i+ʔ	30	0.01%
10	j	j	ʤ	3,381	0.88%
11	k	k	k	21,784	5.65%
11	k	K	x	217	0.06%
11	k	*	*	19	0.00%
12	l	l	ɫ	15,465	4.01%
13	m	m	m	19,237	4.99%
14	n	n	n	22,143	5.74%
14	n	G	ŋ	11,779	3.06%
14	n	N	ɲ	3,741	0.97%
15	o	o	ɔ	13,763	3.57%
15	o	O	ɔɪ	56	0.01%
15	o	5	ɔ+ʔ	60	0.02%
16	p	p	p	12,919	3.35%
17	q	k	k	23	0.01%
18	r	r	r	24,709	6.41%
19	s	s	s	17,602	4.57%
19	s	S	ʃ	206	0.05%
20	t	t	t	18,981	4.92%
21	u	u	u	17,926	4.65%
21	u	*	*	623	0.16%
21	u	6	u+ʔ	19	0.00%
22	v	f	f	745	0.19%
23	w	w	w	1,784	0.46%
24	x	s	s	29	0.01%
25	y	y	j	1,124	0.29%
25	y	*	*	2,085	0.54%
26	z	z	z	397	0.10%

example, a word '*reses*' (recess) is pronounced as /rəsɛs/, but '*resesi*' (recession) is pronounced as /rɛsɛsi/. A grapheme ⟨e⟩ in '*berang*' (irascible) is converted into /ɛ/, but ⟨e⟩ in '*berangin*' (windy) should be pronounced as /ə/. Therefore, the grapheme ⟨e⟩ is predicted to produce so many errors.

Those facts motivate this research to focus on modifying the partial orthogonal binary grapheme encoding used in [9] and incorporating some phonemic rules to reduce the PER of the PNNR-based Indonesian G2P as well as do a more detail evaluation to see its drawbacks.

In the following sections, this paper will discuss how to use PNNR to develop the Indonesian G2P, the proposed modified partial orthogonal binary grapheme encoding and the phonemic rule-based phoneme filtering, the experimental results showing the performance of both proposed concepts, and the conclusion.

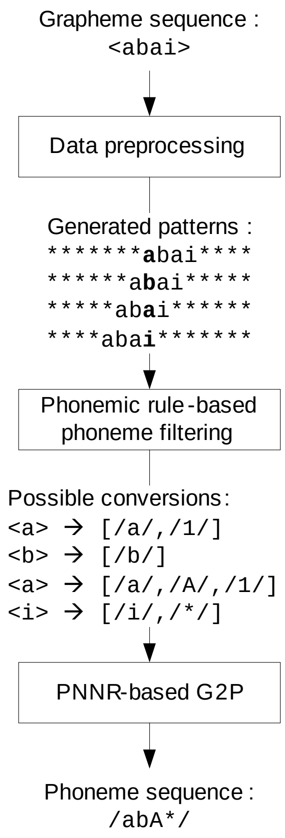

Fig. 1: Block Diagram of Indonesian G2P Using PNNR and Phonemic Rule

II. PNNR-Based G2P

The block diagram of PNNR-Based G2P is illustrated by figure 1. Data preprocessing converts a word (grapheme sequence) into some patterns, where '*' in generated patterns is a symbol for no grapheme and /*/ in possible conversions is a symbol for no phoneme, while /a/, /1/, and /A/ are single phonemic symbols for /ɑ/, /ɑ+ʔ/, and /aɪ/ respectively. The phonemic rule filters some potential conversions to be selected by PNNR, for instance the first grapheme ⟨a⟩ followed by ⟨b⟩ in the given grapheme sequence ⟨abai⟩ is possible to be converted into either /ɑ/ or /ɑ+ʔ/. Finally, the PNNR decides the best conversion of each given grapheme into the possible phonemes.

A. Data Preprocessing

The data used to develop a PNNR-based G2P is a pair of word (sequence of graphemic symbols) and the corresponding pronunciation (sequence of phonemic symbols). The data preprocessing is the sama as described in [9]. First, each grapheme should be mapped into a single phonemic symbol (SPS), as listed in table I. Next, each word should be aligned to the corresponding phonemic symbols. Then, each grapheme occurring in the word is consecutively located as the focus grapheme and the rests on their appropriate contextual positions using contextual length $L = 14$ (7 graphemes on the left and 7 on the right of the focus, as suggested in [9]) as illustrated by figure 1.

B. Modified Grapheme Encoding

The partial orthogonal binary grapheme encoding described in [9] has two problems. Firstly, the distance between a vowel-oriented grapheme and a consonant-oriented grapheme is exactly same as the distance between two consonant-oriented graphemes come from different groups or between a consonant-oriented grapheme and a non-grapheme. This encoding, in some cases, makes the PNNR do a wrong conversion. Secondly, three graphemes ⟨q⟩, ⟨v⟩, and ⟨x⟩ are encoded into three unique group although, in fact, those graphemes are always converted into /k/, /f/, and /s/ respectively.

Based on those facts, the partial orthogonal binary grapheme encoding in [9] is modified as follow:

1) All graphemes and non-graphemes are divided into three main categories: vowel-oriented graphemes, consonant-oriented graphemes, and non-graphemes. Vowels and consonants are designed to have 6 different bits since they are contextually so different in a word (sequence of grapheme). For example, a grapheme ⟨a⟩ followed by a vowel ⟨i⟩ in '*pantai*' (beach) should be pronounced as /aɪ/, but it should be converted into /ɑ/ when it is followed by a consonant ⟨s⟩ '*pantas*' (feasible). But, vowels and non-graphemes as well as consonants and non-graphemes have 5 different bits because they have relatively lower differences. For example, a non-graphemic symbol ⟨-⟩ in '*berang-berang*' (beaver) make grapheme ⟨e⟩ pronounced as /ə/, but grapheme ⟨e⟩ in '*berang*' (irascible) and '*memberang*' (become angry) should be pronounced as /ɛ/.
2) In each main category, some small groups developed by considering the pronunciation similarity of those graphemes in manner and place of articulation as described in [10], [11]. Graphemes in a same group are designed to have 2 different bits since they are contextually so similar in a word, for instance a grapheme ⟨n⟩ followed by either ⟨g⟩ or ⟨k⟩ should be converted into /ŋ/ such as '*bang*' (brother) and '*bank*' (bank) those pronounced as /bɑŋ/. Graphemes in different groups are designed to have 4 different bits as they are contextually quite different, for instance a grapheme ⟨e⟩ preceded by ⟨p⟩ should be converted into /ə/ such as in word '*peran*' (role) that pronounced as /pərɑn/, but it should be converted into /ɛ/ when it is preceded by ⟨s⟩ such as in '*seran*' (strip) that pronounced as /sɛrɑn/. Therefore, graphemes ⟨k⟩ and ⟨g⟩ are grouped into a group, but ⟨p⟩ and ⟨s⟩ are in a different group.
3) Three graphemes ⟨q⟩, ⟨v⟩, and ⟨x⟩ are encoded into the same binary code as graphemes ⟨k⟩, ⟨f⟩, and ⟨s⟩ since they are always converted into /k/, /f/, and /s/ respectively.
4) Three non-graphemes ⟨-⟩, ⟨*⟩, and ⟨space⟩ are encoded into the same binary codes since they have a same function in a word, i.e. no grapheme.

The modified partial orthogonal binary encoding for 26 graphemes and 3 non-graphemes (*, -, and space) are listed in table II. Based on this encoding, two graphemes in a same small group have two different bits and the Euclidean distance between them is $\sqrt{2}$. Those in different small group, but in a same main category, have four different bits and their Euclidean distance is 2. A vowel and a non-grapheme or a consonant and a non-grapheme have five different bits and their Euclidean distance is $\sqrt{5}$. A vowel and a consonant have six different bits and their Euclidean distance is $\sqrt{6}$. This encoding is expected to improve the capability of PNNR in classifying a pattern of grapheme sequence into a correct phoneme.

TABLE II: Modified Partial Orthogonal Binary Encoding for 26 Graphemes and 3 Non-graphemes

Grapheme	Modified partial orthogonal binary code
a	001100000000000000000000000000000000000000000
e	001010000000000000000000000000000000000000000
i	001001000000000000000000000000000000000000000
o	001000100000000000000000000000000000000000000
u	001000010000000000000000000000000000000000000
b	**110000001100000000000000000000000000000000000**
p	**110000001010000000000000000000000000000000000**
t	110000000001100000000000000000000000000000000
d	110000000001010000000000000000000000000000000
k	**110000000000001100000000000000000000000000000**
q	**110000000000001100000000000000000000000000000**
g	**110000000000001010000000000000000000000000000**
c	110000000000000001100000000000000000000000000
j	110000000000000001010000000000000000000000000
f	**110000000000000000001100000000000000000000000**
v	**110000000000000000001100000000000000000000000**
s	110000000000000000000001100000000000000000000
x	110000000000000000000001100000000000000000000
z	110000000000000000000001010000000000000000000
m	**110000000000000000000000001100000000000000000**
n	110000000000000000000000001010000000000000000
h	**110000000000000000000000000001100000000000000**
r	110000000000000000000000000000000001100000000
l	**110000000000000000000000000000000000011000000**
w	110000000000000000000000000000000000000110000
y	**110000000000000000000000000000000000000001100**
*	010000000000000000000000000000000000000000011
-	010000000000000000000000000000000000000000011
space	010000000000000000000000000000000000000000011

C. Phonemic Rule-based Phoneme Filtering

The fifteen phonemic rules, to filter some potential phonemes to be selected by PNNR, are listed in table III. The first column is premise evaluating the focus grapheme (FG) and the first contextual graphemes on the left (L1) and right (R1). The second one is consequent containing one or more impossible phonemes (IP) to filter the possible conversions.

A grapheme is subject to two or more rules so that number of possible conversion to be lower. For example, grapheme ¡a¿ can be converted into four differenet phonemes: /ɑ/, /aɪ/, /aʊ/, or /ɑ+ʔ/. But, ¡a¿ in grapheme sequence ¡aku¿ is subject to both rule 1 and 2 so that the grapheme is not possible to be converted into phoneme /aɪ/ nor /aʊ/. Hence, ¡a¿ in the grapheme sequence ¡aku¿ is only to be converted into /ɑ/ or /ɑ+ʔ/.

D. Pseudo Nearest Neighbour Rule

The PNNR used here works in the same way as in [9]. It selects the best possible phonemes by finding the minimum total distance between the current pattern and the possible phonemes (classes) taking into account the *k* closest patterns. The total distance is calculated using equation 1, where u_j is the weight for the *j*-th neighbour, *L* is the contextual length, d_{li}

TABLE III: Phonemic Rules to Filter Some Potential Phonemes (Conversion Classes), where FG is the Focus Grapheme, IP is Impossible Phoneme(s), L1 and R1 is the First Contextual Grapheme on the Left and the Right Respectively

Premise	Consequent
FG is ⟨a⟩ and R1 is not member {⟨i⟩,⟨y⟩}	IP is /aɪ/
FG is ⟨a⟩ and R1 is not member {⟨u⟩,⟨w⟩}	IP is /aʊ/
FG is ⟨e⟩ and R1 is not member {⟨i⟩,⟨y⟩}	IP is /eɪ/
FG is ⟨e⟩ and R1 is not member {⟨a⟩,⟨e⟩,⟨i⟩,⟨o⟩,⟨u⟩}	IP is {/ɛ+ʔ/,/ɛ+ʔ/}
FG is ⟨g⟩ and L1 is not member {⟨n⟩}	IP is /*/
FG is ⟨i⟩ and L1 is not member {⟨a⟩,⟨e⟩,⟨o⟩}	IP is /*/
FG is ⟨i⟩ and R1 is not member {⟨a⟩,⟨e⟩,⟨o⟩}	IP is /i+ʔ/
FG is ⟨k⟩ and R1 is not member {⟨h⟩}	IP is /x/
FG is ⟨n⟩ and R1 is not member {⟨c⟩,⟨j⟩,⟨s⟩}	IP is /ɲ/
FG is ⟨n⟩ and R1 is not member {⟨g⟩,⟨k⟩}	IP is /ŋ/
FG is ⟨o⟩ and R1 is not member {⟨i⟩,⟨y⟩}	IP is /ɔɪ/
FG is ⟨s⟩ and R1 is not member {⟨y⟩}	IP is /ʃ/
FG is ⟨u⟩ and L1 is not member {⟨a⟩}	IP is /*/
FG is ⟨u⟩ and R1 is not member {⟨a⟩,⟨e⟩,⟨o⟩}	IP is /u+ʔ/
FG is ⟨y⟩ and L1 is not member {⟨n⟩,⟨s⟩}	IP is /*/

and d_{ri} are the distances of the *i*-th contextual grapheme on the left and right calculated using the modified partial orthogonal binary grapheme encoding, and w_i is the weight for the *i*-th contextual grapheme.

$$T = \sum_{j=1}^{k} u_j \sum_{i=1}^{L/2} (d_{li} w_i + d_{ri} w_i) \tag{1}$$

The neighbourhood weight for the *j*-th neighbour, u_j, is formulated by equation 2, where *c* is an power constant as introduced in [9].

$$u_j = \frac{1}{j^c} \tag{2}$$

The graphemic contextual weight used here is an exponentially decaying function as proposed in [9]. It is formulated by equation 3, where w_i is the weight for the *i*-th contextual grapheme, *p* is an exponential constant, and *L* is the graphemic contextual length distributed equally into left and right of the focus. Thus, the first contextual grapheme has the maximum weight since it is the most important grapheme in deciding the best phoneme, whereas the last one has the minimum.

$$w_i = p^{L/2-i+1} \tag{3}$$

III. Experimental Results

The dataset used in this research is a set of 50K words with corresponding pronunciation (phonemic symbols) collected from the great dictionary of the Indonesian language (*Kamus Besar Bahasa Indonesia Pusat Bahasa*or KBBI) third edition, released in 2008, developed by *Pusat Bahasa*. To get a valid evaluation, the PNNR-based G2P model is tested using 5-fold cross-validation. First, the dataset is randomly split into five subsets {1, 2, 3, 4, 5} of 10K different words each. Next, five datasets: A, B, C, D, and E, are developed to contain 40K for parameter tuning and 10K for evaluation. Dataset A consists of {1, 2, 3, 4} for parameter tuning and {5} for evaluation, dataset B consists of {1, 2, 3, 5} for parameter tuning and {4} for evaluation, and so on.

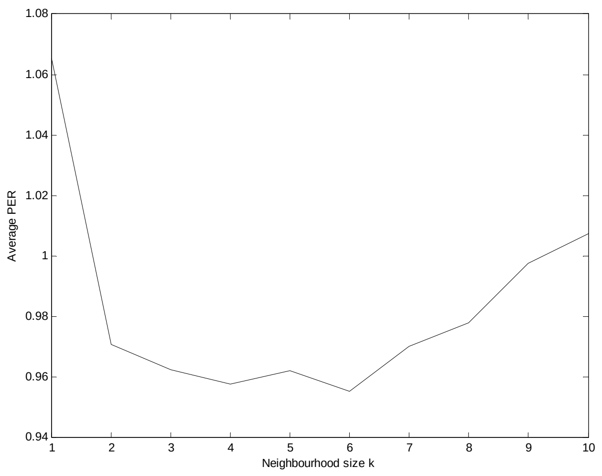

Fig. 2: Average PER of PNNR-based G2P for Varying Neighbourhood Size *k*

A. Optimum Parameters

Three parameters of PNNR, the neighbourhood size *k*, the power constant of neighbourhood weight *c*, and the exponential constant for graphemic contextual weight *p*, are tuned using the five fold datasets. The first parameter to be tuned is *k* since it is so varying based on the problem that it is difficult to be predicted, while the optimum *c* is assumed to be 1.0 and the optimum *p* is predicted to be 2.0. Here the PNNR is evaluated for varying *k* with *c* = 1.0 and *p* = 2.0. The result in figure 2 shows that when *k* = 1 produces very high PER since considering only one neighbour can lead PNNR to be a too general classifier. It also gives high PER when considers so many neighbours that make it too specific in classifying a pattern. The PNNR produces the lowest PER on *k* = 6.

Next, the PNNR with *k* = 6 and *p* = 2.0 is analyzed using varying *c*. The result illustrated by figure 3 shows that when *c* is less than 1.0 the PNNR yields high PER since a low *c* makes the closest neighbour has quite similar distance to the further ones. It also produces high PER when *c* is 1.7 or more since a high *c* makes the closest neighbour has too high distance and the further ones are too low. It produces the lowest PER when *c* = 1.6.

The PNNR with *k* = 6 and *c* = 1.6 is then evaluated using varying *p*. The result in figure 4 shows that very small *p*, less than 1.7, make the PNNR yields high PER because the closest contextual graphemes has a similar importance to the further ones. It gives the lowest average PER, around 0.93%, when *p* = 1.9.

B. Modified Grapheme Encoding and Phonemic Rule

Next, the PNNR-based G2P using modified grapheme encoding and phonemic rule is cross-validated using five fold datasets. The experimental result in table IV shows that it significantly reduces both PER, up to 13.07%, and WER, around 12.13%, relative to the Indonesian PNNR-based G2P proposed in [9].

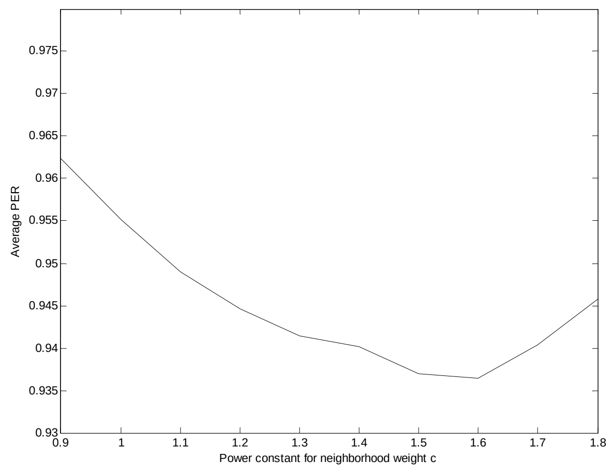

Fig. 3: Average PER of PNNR-based G2P for Varying Power Constant for Neighbourhood Weight *c*

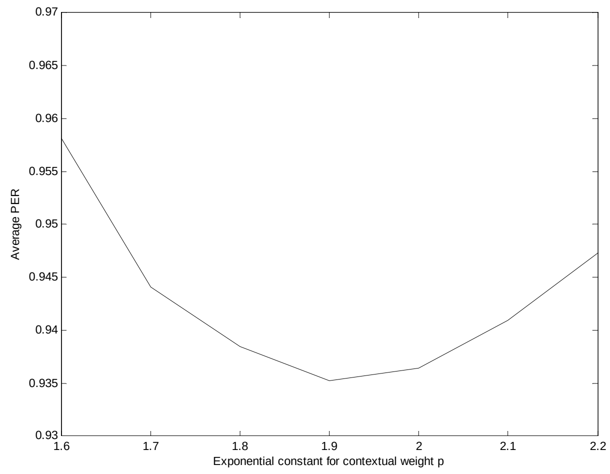

Fig. 4: Average PER of PNNR-based G2P for Varying Exponential Constant for Contextual Weight *p*

TABLE IV: Advantage of Modified Grapheme Encoding and Phonemic Rule for PNNR-based G2P

PNNR-based G2P	PER (%)	WER (%)
Without phonemic rule	1.07	7.67
With phonemic rule and modified grapheme encoding	0.93	6.74

C. Most Errors

As predicted in the introduction, the grapheme ⟨e⟩ produces most errors, up to 82%. This is caused by two problems: 1) Four prefixes, 'ber', 'me', 'per', and 'ter', produce many ambiguous conversions with basic words. For example, a grapheme ⟨e⟩ in a basic word '*berang*' (irascible) is converted into /ɛ/, but ⟨e⟩ in a derivative word '*berangin*' (windy) should be pronounced as /ə/ since 'ber' is a prefix for the basic word '*angin*' (wind) that is always pronounced as /bər/. A grapheme ⟨e⟩ in a basic word '*memang*' (indeed) is converted into /ɛ/, but ⟨e⟩ in a derivative word '*memangsa*' (to prey) should be pronounced as /ə/ since 'me' is a prefix for the basic word '*mangsa*' (prey) that is always pronounced as /mə/. A grapheme ⟨e⟩ in a basic word '*peroksida*' (peroxide) is converted into /ɛ/, but ⟨e⟩ in a derivative word '*perokok*' (smoker) should be pronounced as /ə/ since 'pe' is a prefix for the basic word '*rokok*' (cigarette) that is always pronounced as /pə/. A grapheme ⟨e⟩ in a basic word '*pering*' (tuberculosis) is converted into /ɛ/, but ⟨e⟩ in a derivative word '*teringat*' (remembered) should be pronounced as /ə/ since 'ter' is a prefix for the basic word '*ingat*' (remember) that is always pronounced as /tər/; 2) A grapheme ⟨e⟩ in some similar compound words can be dynamically pronounced as either /ɛ/ or /ə/, such as a word '*termoelektris*' (thermo-electric) that is pronounced as /tərmoɛłɛktris/, but 'termoelektrisitas' (thermo-electricity) should be pronounced as /tɛrmoɛłəktrisitɑs/. A word '*reses*' (recess) is pronounced as /rəsɛs/, but '*resesi*' (recession) is pronounced as /rɛsɛsi/. Such cases are very hard to be solved by the PNNR.

IV. Conclusion

A 5-fold cross-validation shows the modified partial orthogonal binary grapheme encoding and phonemic rule are capable of reducing the relative PER by 13.07%. A detail analysis shows the most errors, up to 82%, come from grapheme ⟨e⟩ that can be dynamically converted into either /ɛ/ or /ə/ since four prefixes, 'ber', 'me', 'per', and 'ter', produce many ambiguous conversions with basic words and also from some similar compound words with both different pronunciations for grapheme ⟨e⟩, such as a word '*reses*' (recess) that is pronounced as /rəsɛs/ but '*resesi*' (recession) is pronounced as /rɛsɛsi/. In the future, a stemming procedure can be incorporated to find the basic words from the derivatives with the four prefixes in order to reduce such errors.

Acknowledgment

The first author is an employee of Telkom Foundation of Education (*Yayasan Pendidikan Telkom*, YPT) as a lecturer at School of Computing, Telkom University (former: Telkom Institute of Technology). Now he is pursuing a doctoral program on Computer Science, Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Gadjah Mada University. This work is supported by YPT with grant number: 15/SDM-06/YPT/2013.

References

[1] A. V. D. Bosch and W. Daelemans, "Data-oriented methods for grapheme-to-phoneme conversion," in *The sixth conference on European chapter of the Association for Computational Linguistics (EACL)*. Morristown, NJ, USA: Association for Computational Linguistics, 1993, pp. 45–53. [Online]. Available: http://portal.acm.org/citation.cfm?doid=976744.976751

[2] F. Yvon, P. B. D. Mareu, C. Alessandro, M. Bagein, G. Bailly, F. Be, S. Foukia, V. Auberge, J. Goldman, E. Keller, D. O. Shaughnessy, V. Pagel, B. Zellner, F. Sannier, and J. Ve, "Objective evaluation of grapheme to phoneme conversion for text-to-speech synthesis in French," *Computer Speech & Language*, vol. 12, no. 4, pp. 393–410, 1998.

[3] P. Taylor, "Hidden Markov Models for Grapheme to Phoneme Conversion," in *INTERSPEECH*, 2005, pp. 1973–1976.

[4] U. D. Reichel and F. Schiel, "Using Morphology and Phoneme History to improve Grapheme-to-Phoneme Conversion," in *INTERSPEECH*. ISCA, 2005, pp. 1937–1940.

[5] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434–451, 2008.

[6] D. Wang and S. King, "Letter-to-sound Pronunciation Prediction Using Conditional Random Fields," *IEEE Signal Processing Letters*, vol. 18, no. 2, pp. 122–125, 2011.

[7] R. Rasipuram, M. M. Doss, and L. Epfl, "Combining Acoustic Data Driven G2P and Letter-to-Sound Rules for Under Resource Lexicon Generation," in *INTERSPEECH*. ISCA, 2012, pp. 1818–1821.

[8] A. Hartoyo and Suyanto, "An improved Indonesian grapheme-to-phoneme conversion using statistic and linguistic information," *International Journal Research in Computing Science (IJRCS)*, vol. 46, no. 1, pp. 179–190, 2010.

[9] Suyanto and A. Harjoko, "Nearest neighbour-based Indonesian G2P conversion," *Telkomnika (Telecommunication, Computing, Electronics, and Control)*, vol. 12, no. 2, pp. 389–396, 2014.

[10] H. Alwi, S. Dardjowidjojo, H. Lapoliwa, and A. M. Moeliono, *Tata Bahasa Baku Bahasa Indonesia (The Standard Indonesian Grammar)*, 3rd ed. Jakarta: Balai Pustaka, 1998.

[11] A. Chaer, *Fonologi Bahasa Indonesia (Indonesian Phonology)*. Jakarta: Rineka Cipta, 2009.