TELKOMNIKA, Vol.12, No.2, June 2014, pp. 389~396
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v12i2.1945

Nearest Neighbour-Based Indonesian G2P Conversion

Suyanto[1], Agus Harjoko[2]
[1] School of Computing, Telkom University
Jalan Telekomunikasi Terusan Buah Batu, Bandung 40257, Indonesia
[2] Faculty of Mathematics and Natural Sciences, Gadjah Mada University
Sekip Utara, Bulaksumur, Yogyakarta 55281, Indonesia
*Corresponding author, e-mail: suy@ittelkom.ac.id[1], aharjoko@ugm.ac.id[2]

Abstract

Grapheme-to-phoneme conversion (G2P), also known as letter-to-sound conversion, is an important module in both speech synthesis and speech recognition. The methods of G2P give varying accuracies for different languages although they are designed to be language independent. This paper discusses a new model based on the pseudo nearest neighbour rule (PNNR) for Indonesian G2P. In this model, a partial orthogonal binary code for graphemes, contextual weighting, and neighbourhood weighting are introduced. Testing to 9,604 unseen words shows that the model parameters are easy to be tuned to reach high accuracy. Testing to 123 sentences containing homographs shows that the model could disambiguate homographs if it uses a long graphemic context.Compared to an information gain tree, PNNR gives a slightly higher phoneme error rate, but it could disambiguate homographs.

Keywords*: grapheme-to-phoneme conversion, Indonesian language, pseudo nearest neighbour rule, partial orthogonal binary code, contextual weighting*

1. Introduction

In general, there are three approaches in G2P, linguistic knowledge-based approach, data-driven approach, and a combination of them. The first approach is commonly used for specific language, but it usually has low generalisation for unseen words. Hence, most recent researches employ the second approach because of flexibility and generalisation, such as information gain tree [1], conditional random fields [2], Kullback-Leibler divergence-based hidden Markov model [3], joint multigram models [4], instance-based learning [5], table lookup with defaults [5], neural networks [6], finite state [7] and [8], morphology and phoneme history [9], hidden Markov model [10], and self-learning techniques [11]. These methods are generally designed to be language independent, but the data sets they use are commonly for English, Dutch, and French.

The Indonesian language has relatively simple phonemic rules. A grapheme <u> is generally pronounced as /u/. But,if <u> is preceeded by <a> in some cases, then it should be pronounced as a diphtong /aw/, such as the word '*kerbau*' (buffalo) that is pronounced as /kerbaw/. This is different from English, where a grapheme <u> could be pronounced as /u/, /a/, or /e/ such as in 'put', 'funny', or 'further'.Indonesian language has thirty two phonemes: sixvowels, four diphtongs, and twenty two consonants [12]. Those phonemes and the related English oneswritten using the ARPAbet symbols can be seen in [13].Indonesian has nine affixes: six prefixes and three suffixes [12]. The usage of sufix '*-i*'may produce an ambiguity between a vowel series anda diphtong. For example, a prefix '*meng-*' followed by a root '*kuasa*' (authority) and a suffix '*-i*' produces a derivative '*menguasai*'. The grapheme <*ai*> in '*menguasai*'isa vowel series /a/ and /i/, but <*ai*>in a root '*belai*' (cares) is a diphtong /ay/. There are so many such cases in the Indonesian language that make the G2P conversion quite hard.

Nearest neighbour is quite a good method for many problems. This method performs a high accuracy for isolated sign language character recognition [14], as well as for bankruptcy prediction models [15]. Now, there are so many variations for this method, such as fuzzy k-NN,neighbourhood weighted nearest neighbour,PNNR, etc. In [16], the researchers show that PNNR performs better than the traditional *k*-nearest neighbour classification rule (kNN), the neighbourhood weighted nearest neighbour classification rule (WNN), and the local mean-based learning method (LM) in large training sample and mixture model data situations. But, in

Received December 12, 2013; Revised April 12, 2014; Accepted May 16, 2014

a small training sample and singular model data, PNNR performs better than both kNN and WNN, but it does not outperform the LM.

This research focuses on developing a new G2P model based on PNNR for the Indonesian language. In this model, a partial orthogonal binary code for graphemes, contextual weighting, and neighbourhood weighting are proposed. This model will be evaluated usinga data set of 47 thousand words and will be compared to the IG-tree method as described in [1].

2. Research Method

Converting a grapheme into a phoneme contextually depends on some other surounding graphemes. The contextual length is varying based on the language. In [5], the optimum contextual lengths for English, Dutch, and French are five graphemes on the left and five graphemes on the right. In other languages with some homographs, the contextual lengths could be longer.

In [1], the calculation of information gain (IG) for 6,791 Indonesian words shows that the focus grapheme has the highest IG (around 3.9). The first graphemes on the right and on the left of the focus have a lower IG than that on the focus, i.e. around 1.1. The IG sharply decreases until the seventh grapheme (around 0.2). Developing a IG-tree usingseven graphemes on the right and the left, commonly written as 7-1-7, produces a phoneme error rate (PER) of 0.99% and a word error rate (WER) of 7.58% for 679 unseen words [1]. This contextual scheme 7-1-7 is adapted in this research.

2.1. Data Preprocessing

The data sets used here are pairs of word (graphemic symbols) and their pronunciation (phonemic symbols). First, each word should be aligned to the corresponding phonemic symbol (see figure 1), where '*' is a symbol for blank (no phoneme). Next, each grapheme occurring in a word is consecutively located as the focus grapheme and the others are located on their appropriate contextual positions as illustrated by figure 2. In the figure, word '*belai*' (cares) is transformed into five patterns.

In this research, each phonemic symbol is designed to have one characterto simplify the alignment process. Table 1 lists all phonemes and their one-character symbols. Phoneme /ng/ is symbolised as /)/ to distinguish it with the phoneme series /n/ and /g/. For instance, two graphemes <n> and <g> in *astringen* should be converted as a phoneme series, not a single phoneme /)/.

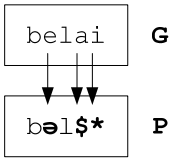

Figure 1. Aligning a word to its phonemic symbols. G = graphemes, P = phonemes

L7	L6	L5	L4	L3	L2	L1	**Focus**	R1	R2	R3	R4	R5	R6	R7	**Class**
*	*	*	*	*	*	*	**b**	e	l	a	i	*	*	*	**b**
*	*	*	*	*	*	b	**e**	l	a	i	*	*	*	*	**ə**
*	*	*	*	*	b	e	**l**	a	i	*	*	*	*	*	**l**
*	*	*	*	b	e	l	**a**	i	*	*	*	*	*	*	**$**
*	*	*	b	e	l	a	**i**	*	*	*	*	*	*	*	*

Figure 2. Locating each grapheme occured in a word as focus and its class. L1 is the first grapheme on the left of the focus and R1 is the first grapheme on the right

Table 1. Indonesian phonemes and their one-character symbols

Phoneme(s)	One-character phonemic symbol
/kh/	(
/ng/	)
/ny/	+
/sy/	~
/ay/	$
/aw/	@
/ey/	%
/oy/	^
/a/ and /?/	1
/e/ and /?/	2
/ə/ and /?/	3
/i/ and /?/	4
/o/ and /?/	5
/u/ and /?/	6

Homographs should be accompanied by one or more other words on the left or right to disambiguate them, as illustrated by Figure 3. The word '*apel*' is a homograph with two different pronunciations, i.e. /*apəl*/ (apple) and /*apel*/ (assembly). The sentence'*mereka memakan buah apel sebelum apel pagi dimulai*' (they eat an apple before the morning assembly) should be included in the data set since some words on the left and right of word '*apel*' are very important to disambiguate that homograph.

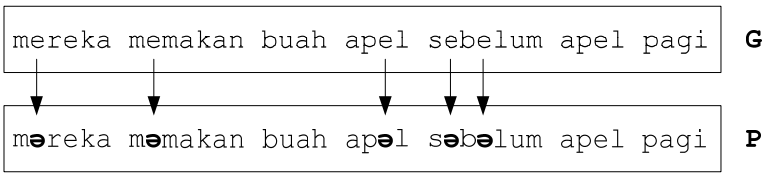

Figure 3. Graphemes in a sentence and the aligned phonemic symbols

2.2. Partial orthogonal binary code

The grapheme encoding used for neural network-based G2P is usually full of orthogonal binary code, such as in [6]. Here a partial orthogonal binary code is proposed by considering a categorisation of Indonesian phonemes based on (1) articulation manners: stop, fricative, nasal, trill, lateral, or semivowel; (2) articulation area: bilabial, labiodental, alveolar, palatal, velar, or glotal; and (3) the condition of vocal cords: voiced or unvoiced. The partial orthogonal binary codes for Indonesian graphemes are listed in table 2. Based on the codes, two graphemes in the same category have two different bits and their euclidian distance is $\sqrt{2}$. But, those in different categories have four different bits and their euclidian distance is 2.

Table 2.Partial orthogonal binary code for all Indonesian graphemes and three other symbols: '*' (blank or no grapheme), '-' (dash), and 'space'.

Grapheme	Binary code
a	11000000000000000000000000000000000000000000000000
e	10100000000000000000000000000000000000000000000000
i	10010000000000000000000000000000000000000000000000
o	10001000000000000000000000000000000000000000000000
u	10000100000000000000000000000000000000000000000000
b	**00000011000000000000000000000000000000000000000000**
p	**00000010100000000000000000000000000000000000000000**
t	00000000011000000000000000000000000000000000000000
d	00000000010100000000000000000000000000000000000000
k	**00000000000011000000000000000000000000000000000000**
g	**00000000000010100000000000000000000000000000000000**
c	00000000000000011000000000000000000000000000000000
j	00000000000000010100000000000000000000000000000000
f	**00000000000000000011000000000000000000000000000000**
v	**00000000000000000010100000000000000000000000000000**
s	00000000000000000000110000000000000000000000000000
z	00000000000000000000101000000000000000000000000000
m	**00000000000000000000001100000000000000000000000000**
n	**00000000000000000000001010000000000000000000000000**
h	00000000000000000000000011000000000000000000000000
x	**00000000000000000000000000110000000000000000000000**
q	00000000000000000000000000011000000000000000000000
r	**00000000000000000000000000001100000000000000000000**
l	00000000000000000000000000000110000000000000000000
w	**00000000000000000000000000000011000000000000000000**
y	00000000000000000000000000000011000000000000000000
*	**00000000000000000000000000000000110000000000000000**
-	00000000000000000000000000000000001100000000000000
space	**00000000000000000000000000000000000011000000000000**

2.3. Contextual weight

As described in [1] and [5], graphemes that closer to the focus have a higher IG. Thus, a contextual weighting used here is an exponential function in order to approach the trend of the IG. The weight of the *i*-th contextual grapheme on the left or the right is formulated by equation 1, where *L* is the contextual length. The focus grapheme is the 0-th context, which has the maximum weight.

$$w_i = p^{L-i+1} \quad (1)$$

Figure 4. illustrates the function for varying *p*. It is clear that the greater the *p,* the sharper the slope of the function. The *p* should be chosen precisely to balance the weighting. The trend of what is described in [1] could be approached by the Figure 4.

For p = 1.6, the weight of the first contextual grapheme is 26.8435 and the sum of the second to the seventh graphemes is 42.0726. It means that further contextual graphemes have quite high importance in deciding the output class. But, if p = 4.0, the weight of the first contextual grapheme is 16,384 and the sum of the others is 5,460. This makes the first contextaul graphemes having very high importance and the further ones are almost useless. It is quite easy to reason that p = 2 will be very good since the weight for the first contextual grapheme is 128 and the sum of the others is 126. Thus, a pattern with wrong graphemes on the first left and the first right context has a low distance. But, if it has correct graphemes on the second to the seventh context, its distance could be quite similar to the patterns that have a correct grapheme either on the first left or on the first right context. However, this is just a prediction. The optimum value of *p* could be some values around 2.0.

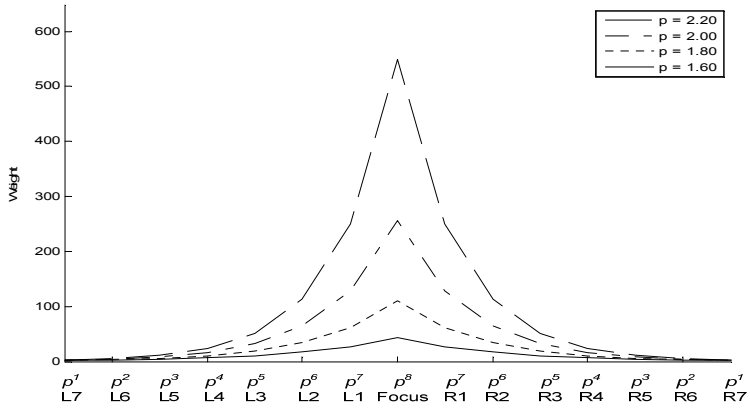

Figure 4. Four examples of contextual weightswith varying*p*and*L* = 7

2.4. Pseudo nearest neighbour rule

PNNR works simply by calculating the total distance of the *k* nearest neighbours in all classes and then deciding a class with a minimum total distance as the output. The *k* nearest neighbours are weighted gradually based on their rankings of distance in ascending order. In [16], the neighbourhood weight for *j*-th neighbour is formulated as one divided by *j*. It is clear that the closest neighbour has a weight of 1, and the weights gradually decrease for the further neighbours. Thus, the furthest neighbour has the lowest weight.

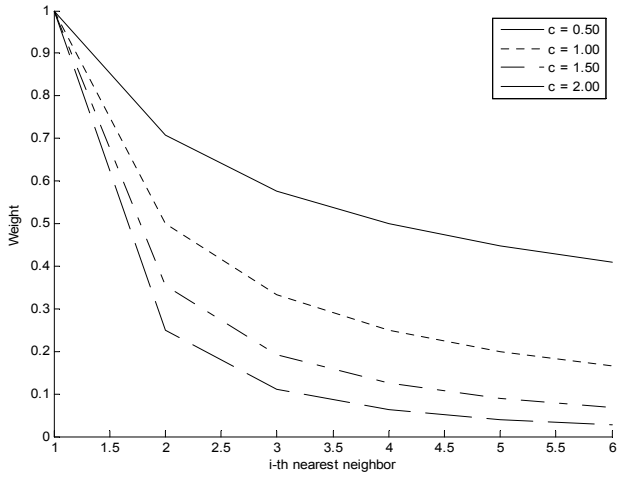

Figure 5. Neighbourhood weights for varying *c*

PNNR as in [16] is adopted in this research because of its high performance for large data sets. But, the neighbourhood weight formula *u* is slightly modified by introducing a constant *c* as a power for the distance ranking to produce varying gradual neighbourhood weights as described in equation 2. Thus, the greater *c,* the sharper decreasing of weight as illustrated by figure 5. It is clear that if *c* = 1.0, then u_j is the same as the formula described in [16].

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

3. Results and Discussion

The data set used here contains 47 thousand pairs of words (and some sentences) and their pronunciation collected from the great dictionary of the Indonesian language(*Kamus Besar Bahasa Indonesia Pusat Bahasa*, abbreviated as KBBI) fourth edition, released in 2008,

developed by *Pusat Bahasa*. The data set is divided into three groups: 60% train set, 20% validation set, and 20% test set.

First, the PNNR is trained using the train set. Next, the trained PNNR is validated using a validation set to get the optimum values forthe three parameters: *k* (neighbourhood size), *p* (contextual weight), and *c* (neighbourhood weight).

3.1. Optimum parameters

First, the optimum valueof*k*should be found since it is quite hard to predictthis parameter. This is performed by using a partial orthogonal binarycode and by assuming the optimum value for *p* is 2.0 (based on the mathematical calculation as described in sub-section 2.3) and the optimum value for *c* is 1.0 (based on the experimental results in [16]). Analysing thetrain set gives the minimum number of patterns in the smallest class as 11. Hence, in this experiment, *k* could be 1 to 11. A computer simulation shows that the PER is high (1.222%) when *k* = 1 since the new pattern in the validation set should be similar to only one pattern in the decision class. It means the PNNR is very specific in deciding the output class. The PER is also high (1.089%) when *k* is 11 that showsthe PNNR is too general. The optimum *k* is 6,which produces the lowest PER (1.065%) and also the lowest WER (7.449%).

Next, the optimum value of *p*is searched using *k*= 6 and *c*= 1.0. The simulation shows that the PER is very high, i.e. 1.153% and 1.132%, when *p* is low (1.50)and*p* is high (4.00) respectively. The optimum *p* is 1.90 whichgives the lowest PER (1.058%).

The optimum value of *c* is then investigated using*k*= 6 and *p* = 1.90 (based on the previous experiments). The simulation shows that the PER is very high, i.e. 1.092% and 1.122%, when *c* is small (0.50)and*c* is big (2.50) respectively.The optimum *c* is 1.07, which produces PER = 1.053%. The greater the *c,* the lower the values of distant neighbours.

Finally, the PNNR with optimum values for those parameters, *k* = 6, *p* = 1.90, *c* = 1.07, and the partial orthogonal binary code, are tested to 9,604 unseen words (75,456 graphemes). The PNNR produces PER = 1.07% and WER = 7.65%, which is quite similar to those of the train set (PER = 1.053%). It shows that the PNNR has very good generalisation capability. These resultsare slightly better than those using the PNNR with a full orthogonal binary code, which produces a PER of 1.08% and WER of 7.68%.

3.2. Homographsdisambiguation

Contextual length *L* could be short or long to see if it could disambiguate homographs. To see the effect of contextual length, 369 sentences containing homographs are added to the train set. Then, 123 unseen homographs are tested to the PNNR using a partial orthogonal code, *k* = 6, *p* = 1.90, *c* = 1.07, and varying *L*. The results are illustrated by figure 6. The WER is very high (54.472%) when *L* = 1. The PNNR reaches optimum for*L* = 8, 9, or 10 with WER =1.63%.

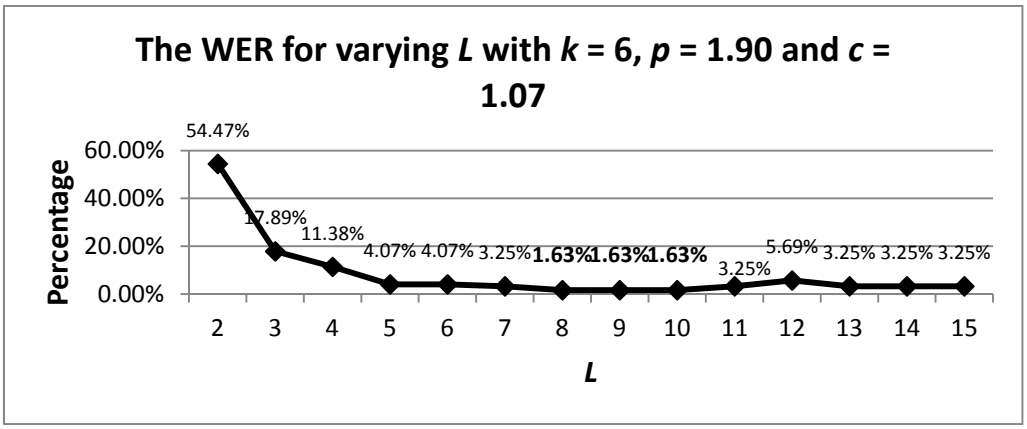

Figure 6. The WER for varying *L*, from 1 to 15

Based on those experimental results, the optimum values for both *p* and *c* are quite easy to be tuned andthey could be predicted mathematically. But, the optimum value for *k* is

quite hard to be predicted since there is no mathematical tool to predict. Some values of *k*, from 5 to 8, produce a similar PER. It means this parameter is not sensitive.

3.3. Comparison of PNNR to IG-tree

The optimum values for all parameters of the PNNR are *k* = 6, *p* = 1.90, *c* = 1.07, and encoding = partial orthogonal binary code. Testing to 9,604 unseen Indonesian words shows that the PNNR has a very good generalisation ability with PER = 1.07% and WER = 7.65%. These results are slightly worse than that of the IG-tree in [1] that produced PER = 0.99% and WER = 7.58%. But, it should be noted that the IG-tree was tested to only 679 unseen Indonesian words. The PNNR is capable of disambiguating homographs, but the IG-tree is not.The IG-tree should be helped by a text-categorisation method to disambiguatehomographs [1].

Table 3.Comparison of the PNNR to the IG-tree

Comparison	IG-tree	PNNR-based
Number of words in testing set	679	9,504
PER	0.99%	1.07%
WER	7.58%	7.65%
Could disambiguate homographs?	No	Yes

3.4. The disadvantages of PNNR-bsed G2P

A PNNR with no linguistic knowledge will have a problem. There are some cases where converting graphemes to the phonemes should not occur because of the dependency of other graphemes on the left or on the right. For instances, see Figure 7.

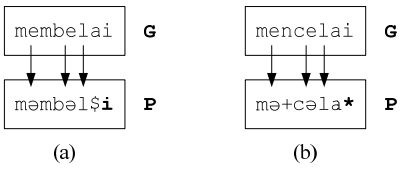

Figure 7. Two examples of wrong G2P conversion

A word '*membelai*' (cares) should be converted into /məmbəl$*/, but the PNNR converted it to /məmbəl$i/. In this case, the grapheme <i> should not be converted to a phoneme /i/ since the left grapheme<a> had been converted to /$/. The grapheme <i> should be converted to /*/. The word '*mencelai*' (to deprecate) should be converted into /mə+cəlai/, but the PNNR converted it to /mə+cəla*/. In this case, the grapheme <i> should not be converted to /*/ since the left grapheme<a> had been converted to /a/. This PNNR has not been designed to handle such cases so it produces some wrong conversions. To solve the problems, some linguistic knowledge could be incorporated into the PNNR. For examples: diphtong /$/ could occur if grapheme <a>is followed by either <i> or <y>; diphtong /@/ occurs if grapheme <a>is followed by either <u> or <w>; diphtong /%/ occurs if grapheme <e>is followed by either <i> or <y>; diphtong /^/ occurs if grapheme <o>is followed by either <i> or <y>; phoneme /(/ occurs if <k> or <c>is followed by <h>; phoneme /)/ occurs if <n>is followed by <g> or <k>; phoneme /+/ occurs if <n>is followed by <y>, <c>, or <j>; phoneme /~/ occurs if <s>is followed by <y>; phoneme /1/, /2/, /3/, /4/, /5/, and /6/ occur if grapheme <a>, <e>, <e>, <i>, <o>, <u>is followed by another constrained vowel respectively.

The PNNR needs as many train setsas possible to be stored as patterngroups. To decide an output class, the PNNR should find the *k* nearest neighbours. Hence, the processing time in the PNNR is relatively longer than that in neural networks or rule-based methods. But, this problem could be solved by an indexing technique.

Contextual weighting used here is an exponential function that is equally used for both left and right contexts. This could be modified to follow the trend of the IG, where the right contexts commonly have a slightly higher IG than the left ones as described in [1]. Hence, contextual weighting function in equation 1 could be split to be two different functions, for the

right and the left context. Next, *p* for the right context could be tuned slightly greater than for the left one.

4. Conclusion

The optimum values for both contextual weight *p* and neighbourhood weight *c* are easy to be tuned since they are not very sensitive. They also could be predicted mathematically. But, the optimum value for neighbourhood size *k* is quite hard to be predicted because there is no mathematical tool to predict.The contextual length *L* could be quite long to disambiguate homographs. Given some representative of enough training sentences, the PNNR-based G2P could convert words into pronunciation symbols. Compare to the IG-tree, the PNNR gives a slightly higher PER, but it could disambiguate homographs. Some linguistic knowledge could be incorporated to improve the accuracy of the PNNR-based G2P.

Acknowledgment

The first author is now a doctoral student in Computer Science Program, Faculty of Mathematics and Natural Sciences, Gadjah Mada University. He 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). This work is supported by YPT with grant number: 15/SDM-06/YPT/2013.

References

[1] Hartoyo A, Suyanto. An improved Indonesian grapheme-to-phoneme conversion using statistic and linguistic information. *International Journal Research in Computing Science (IJRCS).* 2010; 46(1): 179-190.

[2] Dong W, Simon K. Letter-to-sound pronunciation prediction using conditional random fields. *IEEE Signal Processing Letters.* 2011; 18(2): 122-125.

[3] Ramya R, Mathew MD. *Acoustic data-driven grapheme-to-phoneme conversion using KL-HMM*, International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2012 IEEE International Conference on. 2012: 4841-4844.

[4] Maximilian B, Hermann N. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*. 2008; 50(5): 434-451.

[5] Bosch A, Daelemans W. *Data-oriented methods for grapheme-to-phoneme conversion*. Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics, Utrecht, The Netherlands.1993.

[6] Sejnowski TJ, Rosenberg CR. Parallel networks that learn to pronounce English text. *Complex Systems*. 1987: 145-168.

[7] Bouma G. *A finite state and data-oriented method for grapheme-to-phoneme conversion*. Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference, Seattle, Washington. 2000: 303-310.

[8] Caseiro D, Trancoso I, Oliveira L, Viana C. *Grapheme-to-phone using finite-state transducers*. Proceeding IEEE Workshop on Speech Synthesis, Santa Monica, CA, USA. 2002.

[9] Reichel, Uwe D, Florian S, *Using morphology and phoneme history to improve grapheme-to-phoneme conversion*. Proceedings of Interspeech. 2005: 1937-1940.

[10] Taylor P. *Hidden Markov Models for grapheme to phoneme conversion*. Proceedings of Interspeech. 2005: 1973-1976.

[11] Yvon, F. *Self-learning techniques for grapheme-to-phoneme conversion*. Proceeding of the 2nd Onomastica Research Colloquium. London. 1994.

[12] Hasan A, Soenjono D, Hans L, Anton MM. *Tata bahasa baku bahasa Indonesia (The standart Indonesian grammar)*. Jakarta. Balai Pustaka. 1998.

[13] Sakriani S, Konstantin M, Satoshi N. *Rapid development of initial Indonesian phoneme-based speech recognition using the cross-language approach.* Proceeding O-COCOSDA. Jakarta. Indonesia. 2005: 38-43.

[14] Santosa PI. Isolated Sign Language Characters Recognition. *TELKOMNIKA*. 2013; 11(3): 583-590.

[15] Arieshanti I, Purwananto Y, Ramadhani A, Nuha MU. Comparative Study of Bankruptcy Prediction Models. *TELKOMNIKA*. 2013; 11(3): 591-596.

[16] Yong Z, Yupu Y, Liang Z. Pseudo nearest neighbour rule for pattern classification. *Expert Systems with Applications*: *An International Journal*. 2009; 36(2): 3587-3595.