PAPER

Solving the Phoneme Conflict in Grapheme-to-Phoneme Conversion Using a Two-Stage Neural Network-Based Approach

Seng KHEANG[†a)], ***Nonmember***, **Kouichi KATSURADA**[†], **Yurie IRIBE**[††], ***and*** **Tsuneo NITTA**[†,†††], ***Members***

SUMMARY To achieve high quality output speech synthesis systems, data-driven grapheme-to-phoneme (G2P) conversion is usually used to generate the phonetic transcription of out-of-vocabulary (OOV) words. To improve the performance of G2P conversion, this paper deals with the problem of conflicting phonemes, where an input grapheme can, in the same context, produce many possible output phonemes at the same time. To this end, we propose a two-stage neural network-based approach that converts the input text to phoneme sequences in the first stage and then predicts each output phoneme in the second stage using the phonemic information obtained. The first-stage neural network is fundamentally implemented as a many-to-many mapping model for automatic conversion of word to phoneme sequences, while the second stage uses a combination of the obtained phoneme sequences to predict the output phoneme corresponding to each input grapheme in a given word. We evaluate the performance of this approach using the American English words-based pronunciation dictionary known as the auto-aligned CMUDict corpus [1]. In terms of phoneme and word accuracy of the OOV words, on comparison with several proposed baseline approaches, the evaluation results show that our proposed approach improves on the previous one-stage neural network-based approach for G2P conversion. The results of comparison with another existing approach indicate that it provides higher phoneme accuracy but lower word accuracy on a general dataset, and slightly higher phoneme and word accuracy on a selection of words consisting of more than one phoneme conflicts.

key words: *two-stage neural network, grapheme-to-phoneme conversion, many-to-many mapping, prediction through phonemic information, phoneme conflict*

1. Introduction

A speech synthesis system usually creates output speech via phonemic information rather than direct representation of textual information. As a result, the quality of the precise conversion of arbitrary text into its corresponding phoneme string has a strong impact on the performance of the whole system. The phonemic transcription of a written word could possibly be generated by consulting a pronunciation dictionary available inside the system for the in-vocabulary words or predicted through a data-driven G2P conversion for the Out-Of-Vocabulary (OOV) words.

Fundamentally, some previous approaches [2], [3], [7] integrated many-to-one mapping techniques between letters and phonemes, in which a phoneme is determined by using a sequence of letters. This approach proved unsatisfactory because there is no strict correspondence between letters and phonemes [8], especially in the case of a less regular spelling language like English. Various many-to-many mapping techniques between letters and phonemes for taking the G2P conversion to the next level have subsequently been proposed. For example, Rama et al. treat the letter-to-phoneme conversion problem as a phrase-based statistical machine translation problem [11]. They removed the one-to-one alignments from one of the most complex American English words-based dictionary (known as the auto-aligned CMUDict corpus [1]) and induced again many-to-many alignments between letters and phonemes using GIZA++ toolkit. Consequently, they reported 91.4% and 63.81% for the average phoneme accuracy and word accuracy, respectively. Based on the same corpus, the letter-to-phoneme conversion by inference of the rewriting rules provided a 74.40% word accuracy measured in terms of word precision averaged on the full dataset (including the training and testing datasets) [12]. The HMM-based approach with context-sensitive observations for G2P conversion [13], proposed in 2010 by Ogbureke et al., showed a strong interest in the use of context information at both graphemic and phonemic levels. Ogbureke et al. also stated that different corpora always provided different performances because they obtained as much as 79.79% word accuracy on the Unilex corpora containing the UK English words, but only a maximum of 57.85% for the above mentioned CMUDict corpus owing to a large number of loan words and some remarkable errors. Conversely, the joint sequence model, proposed in 2008 by Bisani and Ney [14], is one of the most popular approaches in G2P conversion. Recently, the Weighted Finite-State Transducer (WFST)-based G2P conversion [15] achieved a good word accuracy result (~75.5%) on the CMUDict dataset by utilizing a standard joint N-gram model and investigating N-best rescoring with a Recurrent Neural Network Language Model (RNNLM).

However, it appears that the above-mentioned approaches *–regarded as single-stage model-based approaches–* are not really applicable to the problem of conflicting phonemes at the output level of G2P conversion, where an input grapheme* could, in the same context, produce many possible corresponding output phonemes at the

Manuscript received June 28, 2013.
Manuscript revised November 4, 2013.

[†]The authors are with Toyohashi University of Technology, Toyohashi-shi, 441–8580 Japan.
[††]The author is with Aichi Prefectural University, Nagakute-shi, 480–1198 Japan.
[†††]The author is with Waseda University, Tokyo, 169–8050 Japan.
a) E-mail: kheang@vox.cs.tut.ac.jp
DOI: 10.1587/transinf.E97.D.901

*In this paper, a grapheme is strictly equal to a single letter, rather than a spelling unit.

same time. For instance, if the model takes a sequence of seven graphemes as input, the grapheme "A" on sequence "HEMATIC" can produce the phoneme /AE/ when it belongs to the word "SCHEMATIC", and also /AH/ when it is within another word "MATHEMATICIAN". Thus, it is difficult to identify the correct phoneme corresponding to "A" since there is more than one choice. This kind of problem may negatively impact the performance of the G2P conversion model. Consequently, this paper aims to take it into account in order to help to improve the phoneme predicting quality in G2P conversion.

Over the years, several different neural network-based approaches for G2P conversion have been developed; however, recently they have not been very competitive [17]. Most of these approaches were constructed as one-stage models [2], [4], [5], so they were not integrated with the many-to-many mapping technique between graphemes and phonemes. Considering these facts, in this paper, a two-stage neural network-based approach for G2P conversion is reasonably proposed, which enables the use of grapheme and phoneme contexts in a way that is different from that of previous approaches for dealing with the problems outlined above. The first-stage neural network is implemented as a many-to-many mapping model between graphemes and phonemes for the automatic conversion of word to phoneme sequences. Next, the second stage uses a combination of the phoneme sequences obtained as an input pattern to predict the output phoneme corresponding to each input grapheme in a given word. At this stage, it is particularly capable of generating different phonemic patterns from the same input grapheme sequence that appears in different words.

We evaluate our approach against two proposed baselines using the reconstructed version of the auto-aligned CMUDict corpus. The results indicate as much as a 2% improvement in word accuracy measured on the OOV words. Thus, the approach can be regarded as an improvement on the previous one-stage neural network-based G2P conversion.

The remainder of this paper is organized as follows: In Sect. 2, we discuss the ability currently lacking in single-stage neural network-based G2P conversion. We then describe the two-stage neural network-based approach in Sect. 3, and present its experimental results in Sect. 4. We discuss the experimental results by investigating the error analysis in Sect. 5 and then conclude this paper in Sect. 6.

2. Single-Stage Neural Network-Based G2P Conversion

The G2P conversion model was established for use in predicting the phonemes corresponding to the input text[†], especially the OOV words. It is usually trained using the graphemes-phonemes pairs (*g-p* pairs) extracted from a pronunciation dictionary, a text file containing a large number of words together with their phonetic transcriptions. In this case, each word and its pronunciation in the dictionary must be aligned before being used. Therefore, for each occurrence (i.e., *word→phonemes*) of the auto-aligned CMUDict corpus [1], both grapheme and phoneme sequences have the same length owing to the use of empty grapheme "_" and empty phoneme /_/ notations. For example, the phoneme sequence of the word "CAPAB_ LE" is represented by /K EY P AH B AH L _/.

2.1 Mapping Technique between Graphemes and Phonemes

The context-dependent grapheme model considers the association between graphemes and phonemes as *many-to-one* [2]. Thus, the extracted *g-p* pairs are obtained by passing two different slicing windows [6] through each occurrence of the dictionary; a window is passed through the word grapheme-by-grapheme, while another window, one phoneme in size, is passed through its corresponding phoneme string phoneme-by-phoneme [4]. In this context, several graphemes as input and a single phoneme as output are required. For example, if the word $\mathbf{G} = \mathbf{g_1 g_2} \ldots \mathbf{g_n}$ corresponds to the phoneme sequence $\mathbf{P} = \mathbf{p_1 p_2} \ldots \mathbf{p_n}$, then the extracted pair between the focal grapheme g_i at position i (where $i = 1, \ldots, n$) and its corresponding phoneme p_i is represented as below:

$$\begin{aligned} \underbrace{g_{i-x} + \cdots + g_{i-1} + \ g_i \ + g_{i+1} + \cdots + g_{i+x}} &\rightarrow p_i \\ \Leftrightarrow \quad seq(g_i, x) &\rightarrow p_i \end{aligned} \tag{1}$$

Where $g \in \{\text{“}A\text{”}, \text{“}B\text{”}, \ldots, \text{“}Z\text{”}, \text{empty grapheme “\_”}\}$

$p \in \{/AA/, /AE/, \ldots, \text{empty phoneme } /\_/\}$

Here, + denotes sequence concatenation. The segments $(g_{i-x}+\cdots+g_{i-1})$ and $(g_{i+1}+\cdots+g_{i+x})$ represent left and right contexts of the focal grapheme g_i, respectively, while x indicates the size of each context side. In this equation, an input sequence $seq(g_i, x)$ is constructed by concatenating the focal grapheme g_i with its left and right context information, so the length of this sequence is equal to $(2x + 1)$.

On the other hand, considering the correspondence between graphemes and phonemes as *many-to-many* has also been stated as a beneficial technique in many recent studies because it can cover all possible mappings between graphemes and phonemes (e.g., one-to-one, many-to-one, one-to-many, and many-to-many) [10], [11], [13], [19]. These techniques inspired us to incorporate the context-dependent phoneme model into neural network-based G2P conversion. This results in phoneme p_i in Eq. (1) being definitely replaced by the phoneme sequence $seq(p_i, y)$, where y indicates the size of each context side of p_i. Inversely, Eq. (2) becomes Eq. (1) once the parameter y is set to zero.

$$\begin{aligned} \underbrace{g_{i-x}+\cdots+g_i+\cdots+g_{i+x}} \ &\rightarrow \ \underbrace{p_{i-y}+\cdots+p_i+\cdots+p_{i+y}} \\ \Leftrightarrow \ seq(g_i, x) \ &\rightarrow \ seq(p_i, y) \end{aligned} \tag{2}$$

[†]Here, the input text is just a single word because the pronunciation dictionary being used [1] contains isolated words only.

Table 1 List of the *g-p* pairs extracted from two given words ("SCHEMATIC" and "MATHEMATICIAN") by using a slicing window of seven graphemes (x=3) as input and another window of one (y=0) or five phonemes (y=2) as output.

g-p pair	**Input** +grapheme context (7 graphemes) $seq(g_i, 3)$	**Output** no context (1 ph.) $seq(p_i, 0)$	**Output** +phoneme context (5 phonemes) $seq(p_i, 2)$
P1	_ _ _ ***S*** C H E	***S***	_ _ ***S*** K _
P2	_ _ S ***C*** H E M	***K***	_ S ***K*** _ AH
P3	_ S C ***H*** E M A	_	S K _ AH M
P4	S C H ***E*** M A T	***AH***	K _ ***AH*** M AE
P5	C H E ***M*** A T I	***M***	_ AH ***M*** AE T
P6	H E M ***A*** T I C	***AE***	AH M ***AE*** T IH
P7	E M A ***T*** I C _	***T***	M AE ***T*** IH K
P8	M A T ***I*** C _ _	***IH***	AE T ***IH*** K _
P9	A T I ***C*** _ _ _	***K***	T IH ***K*** _ _
P10	_ _ _ ***M*** A T H	***M***	IH K ***M*** AE TH
P11	_ _ M ***A*** T H E	***AE***	K M ***AE*** TH _
P12	_ M A ***T*** H E M	***TH***	M AE ***TH*** _ AH
P13	M A T ***H*** E M A	_	AE TH _ AH M
P14	A T H ***E*** M A T	***AH***	TH _ ***AH*** M AH
P15	T H E ***M*** A T I	***M***	_ AH ***M*** AH T
P16	H E M ***A*** T I C	***AH***	AH M ***AH*** T IH
P17	E M A ***T*** I C I	***T***	M AH ***T*** IH SH
P18	M A T ***I*** C I A	***IH***	AH T ***IH*** SH _
P19	A T I ***C*** I A N	***SH***	T IH ***SH*** _ AH
P20	T I C ***I*** A N _	_	IH SH _ AH N
P21	I C I ***A*** N _ _	***AH***	SH _ ***AH*** N _
P22	C I A ***N*** _ _ _	***N***	_ AH ***N*** _ _

2.2 Lack of Ability in Phoneme Prediction

When the G2P conversion is treated as a single-stage model, the output phoneme is always predicted directly through the input graphemic information [2], [5]. Table 1 clearly shows that in this case the model lacks the ability to solve the phoneme conflicts at the output level of G2P conversion. For example, it is impossible to distinguish between the conflicted pairs *P6* and *P16* because they have the same input sequence (*e.g., "HEMATIC"*) but different outputs (e.g., */AE/* and */AH/*). Even when the phoneme context gets involved (y>0) in the model or not (y=0), the problem always remains because only one phoneme is obviously produced at the output layer of the model.

In addition, it appears that the grapheme side does not carry enough information or knowledge relating to the phonological interaction [9]. Therefore, the grapheme-based phoneme prediction method implemented in single-stage model-based approaches does not appear to be very effective for improving the G2P conversion performance as long as the conflict at the phonemic level remains unsolved.

3. Two-Stage Neural Network-Based G2P Conversion

In order to deal with the problem discussed in the previous section without affecting the previous many-to-many mapping technique, we employed a two-stage neural network-based approach for G2P conversion.

In this section, we first propose a new phoneme-based method for predicting the output phonemes corresponding to the given words. We then describe the architecture of the proposed approach.

3.1 Prediction Using Phonemic Information

Even though multiple output phonemes can be mapped to the same input grapheme sequence, phoneme prediction in G2P conversion should be done at the phonemic level itself rather than the graphemic level because the grapheme side does not contain enough information relating to the phoneme interactions. From this point of view, we propose a new phoneme prediction method in which the phonemic information is used as input to select the best final output phoneme. Because the G2P conversion model theoretically uses text as input, our proposed method has to be divided into two consecutive steps:

Grapheme sequence ⇒ *Phoneme sequence* ⇒ *Phoneme*

The proposed method first converts the graphemic information into phonemic information without worrying about any conflict at the phonemic level. In this step, each grapheme sequence can produce only one output phoneme sequence at a time. Next, all the related output phoneme sequences are combined and used at the second step of execution to predict the exact output phoneme of the G2P conversion model.

3.2 Architecture of the G2P Conversion Model

On the basis of the new phoneme prediction method presented in the previous section, the proposed G2P conversion model is fundamentally built by putting two different multilayer neural networks in sequence as depicted in Fig. 1. The first neural network is implemented as a many-to-many conversion model to automatically transform each grapheme sequence extracted from a given word into the corresponding phoneme sequence. This facilitates coverage of all possible graphemes-phonemes associations. The second neural network then uses each combination of the obtained phoneme sequences as an input pattern to enable prediction of the final output phoneme corresponding to each input grapheme in the given word. This stage is specially established to take action on the problem of conflicting phonemes, which is impossible to solve in the first stage model.

3.2.1 First-Stage Neural Network

As depicted in Fig. 1, the first-stage neural network is constructed based on the same technique described in Sect. 2, which was implemented to automatically convert a sequence of graphemes (i.e., $seq(g_i, x)$) into another sequence of phonemes (i.e., $seq(p'_i, y)$) that is necessary for helping the second-stage neural network to generate different phonemic patterns out of the same input grapheme sequence appearing in two or more different words.

This model is trained with the *g-p* pairs extracted with

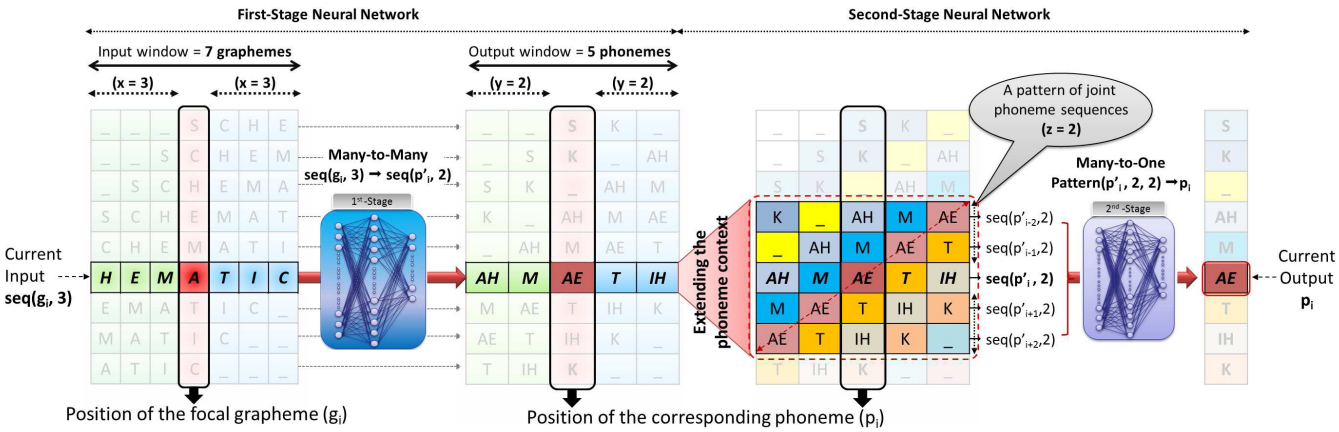


Fig. 1 Architecture of a Two-Stage Neural Network-based approach for G2P conversion, performed on the occurrence "SCHEMATIC" → /S K _ AH M AH T IH K/ while $x = 3$ (seven graphemes), $y = 2$ (five phonemes) and $z = 2$ (five sequences).

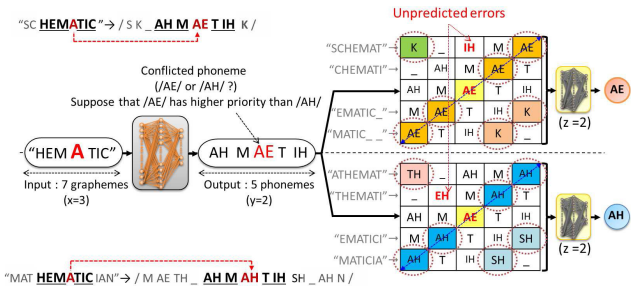


Fig. 2 An example that demonstrates how to solve the phoneme conflicts in G2P conversion using the two-stage neural network-based approach.

respect to Eq. (2) from all the occurrences of the pronunciation dictionary. For example, according to Table 1, if x=3 and y=2 are set, then 22 extracted pairs are obtained from two given words "SCHEMATIC" and "MATHEMATICIAN". After the training process terminates, according to the left part of Fig. 2, some output information (e.g., the phoneme /AH/ or the phoneme sequence /AH M AH T IH/) is lost because of the phoneme conflicts, so the same output phoneme sequence /AH M AE T IH/ is generated from the input of both pairs P6 and P16.

3.2.2 Second-Stage Neural Network

According to Fig. 1, for an input word $G = g_1 g_2 \ldots g_n$ containing n graphemes, a set of n phoneme sequences (e.g., $seq(p'_1, y)$, $seq(p'_2, y)$, …, $seq(p'_n, y)$) are produced after terminating the process at the first-stage neural network. Thus, the desired output phoneme p_i corresponding to the focal grapheme g_i on sequence $seq(g_i, x)$ can be predicted by investigating the information related to p_i (i.e., this refers to p'_i) that can be found at different locations within some of the obtained phoneme sequences; in the case where the current input grapheme sequence $seq(g_i, x)$ outputs the phoneme sequence $seq(p'_i, y)$ at the first-stage neural network, the information concerning p'_i can be found as follows:

- At the central position of the current phoneme sequence $seq(p'_i, y)$.
- Within the right context side of the phoneme sequences preceding $seq(p'_i, y)$. As seen in Fig. 1, those preceding phoneme sequences include $seq(p'_{i-1}, y)$, $seq(p'_{i-2}, y)$, …, $seq(p'_{i-z+1}, y)$ and $seq(p'_{i-z}, y)$.
- Within the left context side of the phoneme sequences succeeding $seq(p'_i, y)$. As can be seen in Fig. 1, those succeeding phoneme sequences include $seq(p'_{i+1}, y)$, $seq(p'_{i+2}, y)$, …, $seq(p'_{i+z-1}, y)$ and $seq(p'_{i+z}, y)$.

Here, parameter z indicates the number of preceding or succeeding phoneme sequences. In this paper, the phoneme sequences preceding and succeeding $seq(p'_i, y)$ are called *the neighborhood phoneme sequences of* $seq(p'_i, y)$.

Consequent on these facts, we propose the phoneme context extending technique in which all the related phoneme sequences (i.e., the sequences containing information about p'_i, which include the current phoneme sequence and its neighborhood sequences) are concatenated. This can generate a phonemic pattern with larger context including a strong knowledge related to the phonological interaction between the output phoneme p_i and other phonemes in the conversing word. Since the neural network-based approach is used at the first-stage, it is then used at the second stage because of the coding time reduction and its simple implementation. Therefore, the second-stage neural network determines the final output phoneme via the generated pattern using the following equation:

$$\overbrace{seq(p'_{i-z}, y) + \cdots + seq(p'_i, y) + \cdots + seq(p'_{i+z}, y)}^{\text{preceding sequences} \qquad \text{succeeding sequences}} \rightarrow p_i$$
$$\Leftrightarrow \quad Pattern(p'_i, y, z) \rightarrow p_i \tag{3}$$

Owing to the problem presented in Table 1, it is difficult to distinguish the output between the g-p pair P6 and P16 because they have the same input grapheme sequence (e.g.,

"HEMATIC"). However, the example in Fig. 2 demonstrates that our two-stage neural network-based approach for G2P conversion can provide a good solution to the problem by adding the second-stage neural network model. This facilitates the creation of two different phonemic patterns representing the grapheme "A" in sequence "HEMATIC", which belongs to two different words (e.g., "SCHEMATIC" and "MATHEMATICIAN"). Furthermore, the phonemes along the diagonal positions and those at the top-left, as well as the bottom-right of each pattern, are very important for distinguishing between the output phonemes in cases where they have the same input grapheme sequences.

In practice, some unpredicted errors occurred after the first-stage neural network because it virtually impossible to obtain a perfectly trained neural network to represent a complex system like G2P conversion. Fortunately, as can be seen in Fig. 2, these errors could help to produce some extra patterns for the second-stage sometimes.

4. Evaluation

In this section, we first describe the data preparation process. We then briefly explain the experimental setup, after which we report on the experimental results obtained from various proposed test sets.

4.1 Data Preparation

4.1.1 Auto-Aligned CMUDict Corpus

We chose the American English words-based pronunciation dictionary (known as the auto-aligned CMUDict corpus [1]) to evaluate the performance of our proposed approach against two baseline approaches. This corpus, which contains many acronyms and loan words from different languages such as Japanese, French, and German, has been widely used by researchers [11]–[13]. It was originally created using 34 graphemic symbols (e.g., "A"…"Z", "2"…"7", "9" and empty grapheme "_") and 40 phonemic symbols (e.g., /AA/, /AE/, /SH/, empty phoneme /_/, etc.).

It comprise a total of 112,102 isolated words, including 838,996 graphemes and phonemes, owing to the aligned corpus. Further, it was originally subdivided into 10 folds (e.g., part0, …, part9) each of which contains almost the same number of words, graphemes as well as phonemes [22].

4.1.2 Newly Aligned CMUDict Corpus

Various researchers have stated that the auto-aligned CMUDict corpus has a lower consistency than other corpora and also has errors [10], [13], while others have emphasized that the quality of the pronunciation dictionary could negatively affect the G2P conversion performance [18]. As a result, we reconstructed a version of the auto-aligned CMUDict corpus with higher consistency (i.e., a newly aligned CMUDict corpus) using the GIZA++ toolkit and then used it in our experiments. Because the number of numeric graphemes was too low, all of the words containing numeric graphemes were removed. As a result, it remained only 27 graphemic symbols remained in the new corpus.

Case of alignment	Original CMU Dictionary [1]	Newly aligned CMU Dictionary
-LE {-BLE, -TLE, -PLE, …}	C O P P _ L E K A A P _ AH L _	C O P P _ L E K A A P _ AH L _
-ION	D E C O M P O S I T I O N D IY K AH M P OW Z IH SH _ AH N	D E C O M P O S I T I O N D IY K AH M P OW Z IH SH _ AH N
	D E C I S I _ O N M A K I N G D AH _ _ S IH ZH AH N M EY K IH NG _	D E C I S I O N M A K I N G D AH S IH ZH _ AH N M EY K IH NG _
	A C C L I M _ _ A T I O N AE _ K L AH M EY SH AH _ _ _ N	A C C L I M A T I O N AE K _ L AH M EY SH _ AH N
	D E F A M _ _ A T I O N D EH F AH M EY SH AH _ _ _ N	D E F A M A T I O N D EH F AH M EY SH _ AH N
-ED	C R A M P E D K R AE M P T _	C R A M P E D K R AE M P _ T
	B I A S E D B AY AH S T _	B I A S E D B AY AH S _ T
Others	_ _ A W F U L N E S S AO F AH _ _ _ L N AH _ S	A W F U L N E S S AO _ F AH L N AH S _
	B _ _ I C Y C L E D B AY S IH K _ AH L _ D	B I C Y C _ L E D B AY S IH K AH L _ D
	B O G A _ _ C K I B AH G AA T S K _ IY	B O G A _ C K I B AH G AA T S K IY
	C A _ _ C I Q U E K AH S IY K _ _ _ _	C A C I Q U E K AH S IY K _ _

Fig. 3 Comparison of the alignment between graphemes and phonemes in the auto-aligned CMUDict (column 2) and the newly aligned CMUDict (column 3).

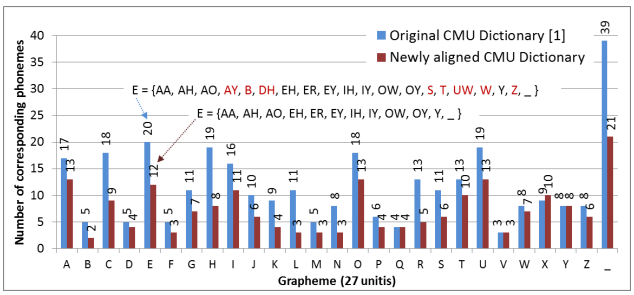


Fig. 4 Consistency measurement based on the number of corresponding phonemes that could be mapped by each grapheme inside the original and new datasets.

The resulting corpus proved more reliable and consistent than the original. Figure 3 shows that the word located in the third column is always shorter and well-aligned than the one located in the second column. In addition, by counting the phonemes that could possibly be mapped from each grapheme, Fig. 4 demonstrates that the grapheme in the newly aligned CMUDict corresponds to fewer numbers of phonemes than the one inside the auto-aligned CMUDict. For example, the number of phonemes that could be mapped by the grapheme "E" is reduced from 20 to only 12.

4.2 Experimental Setup

4.2.1 Training and Testing Datasets

We conducted experiments on the newly aligned CMU-

Dict corpus. Nine out of 10 folds (e.g., part0, ..., part8) were combined and then used as a training dataset, while the remainder fold (e.g., part9) was used as a testing dataset. Thus, the training dataset contained a total of 100,713 words or 750,198 graphemes/phonemes, while the testing dataset contained 11,188 OOV words or 83,267 graphemes/phonemes.

To achieve accurate phoneme prediction, we used the Orthogonal Binary Codes (OBC) [4] to encode each symbol, where the length of a vector corresponding to a single symbol was exactly equal to the total number of symbols in the group the symbol belongs to, and therefore each grapheme and phoneme was represented using a vector of 27 elements (or 27 neurons) and 40 elements (or 40 neurons), respectively. For each vector, only one element at a specific index was active or set to one, while the others were set to zero.

- "A" = 100000000000000000000000000
- "B" = 010000000000000000000000000
- "H" = 000000010000000000000000000
- /AA/ = 1000000000000000000000000000000000000000000
- /CH/ = 0000000001000000000000000000000000000000000

4.2.2 Four Different Test Sets

In this research, we proposed and separately utilized four different test sets. First, we created a simple baseline approach (*Baseline1*) and implemented it using only a one-stage neural network. In accordance with Fig. 3, this baseline was built using Eq. (1) or Eq. (2) with $y = 0$.

Next, we proposed an extended interesting baseline approach (*Baseline2*) to help prove that the performance of the G2P conversion model can possibly be improved by just adding the second-stage model. As depicted in Fig. 5, this baseline was designed with respect to the architecture of our two-stage model-based approach, with the exception that the first-stage neural network was replaced by the first baseline approach. This means that once the phoneme context is not involved in the model (when $y = 0$), each output phoneme sequence at the first-stage neural network contained only one phoneme per sequence (i.e., $seq(p'_i, 0) = p'_i$).

We also utilized two other test sets using the same *two-stage neural network-based approach* (written as TSNN in this section to reduce word repetition), but different configurations. For the first configuration (*TSNN_3ph*), we used a sequence of three phonemes (i.e., $y_2 = 1$) as the output of the first-stage neural network, and also three phoneme sequences (i.e., $z_2 = 1$) as the input of the second-stage neural network. We then enlarged the size of the phoneme sequence from three to five phonemes (i.e., $y_1 = 2$) and also the number of sequences from three to five sequences (i.e., $z_1 = 2$) for another configuration (*TSNN_5ph*).

As can be seen in Fig. 1, TSNN_5ph uses a pattern of five joint phoneme sequences obtained from the first-stage neural network to predict the final output phoneme at the second-stage neural network. This means that five input grapheme sequences are involved in the generation of each pattern. This may appear unfair if we compare the performance of TSNN_5ph with that of Baseline1 and Baseline2 using the same input grapheme sequence size. Therefore, the size of the grapheme sequence being used in both baseline approaches must be longer than that being used in TSNN_5ph and depend on the value of z; according to the observation of the five grapheme sequences involved, the bottom part of Fig. 5 shows that each input grapheme sequence used in both baseline approaches must contain four graphemes more than used in TSNN_5ph (i.e., according to Table 2, $x_3 = x_1 + z_1 = x_1 + 2$) and two graphemes more than used in TSNN_3ph (i.e., $x_3 = x_2 + z_2 = x_2 + 1$). Likewise, at the second-stage of TSNN_5ph, only nine exact phonemes are found within each generated pattern of five joint phoneme sequences. Thus, the number of input phonemes at the second-stage of Baseline2 should be equal to nine phonemes (i.e., according to Table 2, $z_3 = z_1 + y_1 = 4$).

According to Eq. (1), for each test set in Table 2, the size of the input grapheme window must be an odd number depending on its context size (e.g., $x_1 = 3 \rightarrow 7$, $x_2 = x_1 + 1 = 4 \rightarrow 8$ and $x_3 = x_1 + 2 = 5 \rightarrow 9$).

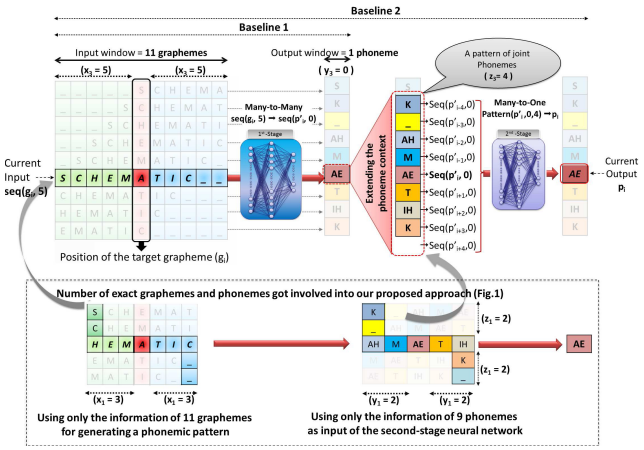


Fig. 5 Architecture of Baseline1 and Baseline2.

Table 2 Configuration of the four proposed test sets.

	First-stage $seq(g_i, x) \rightarrow seq(p'_i, y)$		Second-stage $Pattern(p'_i, y, z) \rightarrow p_i$	
	x (# gr.)	y (# ph.)	y (# ph.)	z (# seq.)
Baseline 1	x_3: 5→9	y_3: 0		
Baseline 2	x_3: 5→9	y_3: 0	y_3: 0	z_3: 4
Two-stage neural network using 3ph.	x_2: 4→8	y_2: 1	y_2: 1	z_2: 1
Two-stage neural network using 5ph.	x_1: 3→7	y_1: 2	y_1: 2	z_1: 2

4.2.3 Configuration of FANN Parameters

We implemented each neural network stage of our proposed model using the functions provided by the FANN (Fast Artificial Neural Network[†]) library. We obtained the best results when each stage was set up as follows:

- Standard neural network with three layers
- Incremental backpropagation algorithm
- Learning rate = 0.8; Momentums = 0.1
- Symmetrical sigmoid activation function[††], where the value of steepness is equal to 0.01
- Number of neurons at the first stage:
  - Input layer $= (2x + 1) * 27$
  - Hidden layer $= (2x + 1) * 27 * 2$
  - Output layer $= (2y + 1) * 40$
- Number of neurons at the second stage:
  - Input layer $= ((2z + 1) * (2y + 1)) * 40$
  - Hidden layer $= ((2z + 1) * (2y + 1)) * 40/2$
  - Output layer = 40
- $(2x+1)$, $(2y+1)$ and $((2z+1) * (2y+1))$ are the sizes of $seq(g_i, x)$, $seq(p_i, y)$ and $Pattern(p'_i, y, z)$, respectively.

4.2.4 Accuracy Measurements

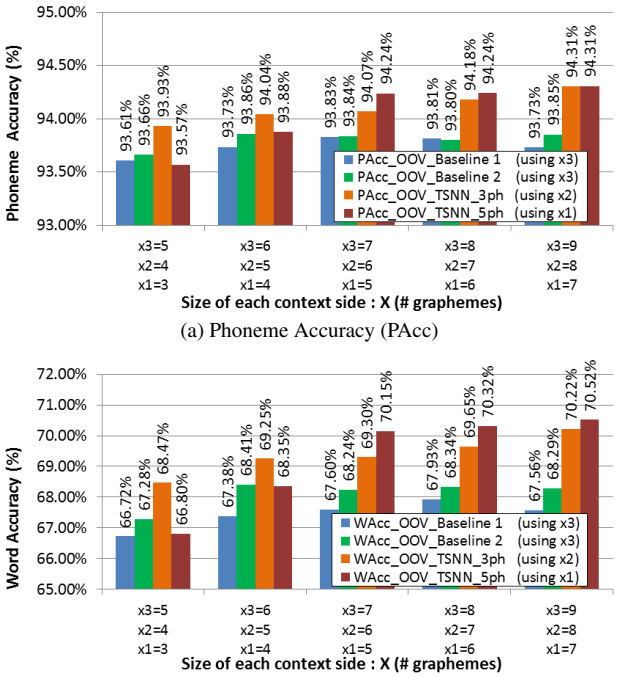
To compare with other approaches introduced in Sect. 1, we evaluated the performance of the model in terms of *phoneme accuracy* (PAcc) and *word accuracy* (WAcc) using the NIST sclite scoring toolkit[†††]. Because the goal of this paper is improvement of the performance of G2P conversion measured on the OOV words, we only report results related to this objective. PAcc and WAcc are calculated as follows:

$$PAcc = 1 - PER = 1 - ((S_p - D_p - I_p)/N_p)$$
$$WAcc = 1 - WER = 1 - (S_w/N_w)$$

where PER and WER are known as Phoneme Error Rate and Word Error Rate, respectively; S_p, D_p, I_p and N_p are the number of phoneme substitutions, phoneme deletions, phoneme insertions, and total phonemes in the reference, respectively. Since only isolated words were used in our experiments, the value of WER was exactly equal to the number of word substitution (S_w) divided by the total number of words in reference (N_w).

4.3 Experimental Results

Each proposed test set used the newly aligned CMUDict corpus to evaluate the model performance. Based on Fig. 6, by investigating various input grapheme sequence sizes (e.g., when $x_1 = 3 \rightarrow 7$, $x_2 = 4 \rightarrow 8$ and $x_3 = 5 \rightarrow 9$), our proposed two-stage neural network-based approach usually provided higher PAcc and WAcc than both baseline approaches.



(a) Phoneme Accuracy (PAcc)

(b) Word Accuracy (WAcc)

Fig. 6 PAcc and WAcc measured on the OOV words.

Table 3 Word Error Rate (WER) of the four proposed test sets, which were evaluated on the OOV words and grouped by the number of erroneous phonemes per word. These reported results were obtained with $x_1 = 7$, $x_2 = 8$ and $x_3 = 9$.

Nb. of errorenous phonemes per word	WER Baseline1 ($x3 = 9$)	WER Baseline2 ($x3 = 9$)	WER TSNN_3ph ($x2 = 8$)	WER TSNN_5ph ($x1 = 7$)
1	21.92%	21.30%	20.25%	19.89%
2	7.55%	7.52%	7.04%	6.99%
3	2.37%	2.33%	1.99%	2.03%
4	0.48%	0.44%	0.40%	0.46%
5	0.10%	0.08%	0.08%	0.11%
6	0.02%	0.04%	0.01%	0.01%

Further, it was also proved that the performance of the G2P conversion given by each test set increased relative to the size of the input grapheme sequence; a nice improvement in WAcc occurred once the number of graphemes started increasing from seven to eleven graphemes (i.e., $x_1, x_2, x_3 = 3 \rightarrow 5$). However, for our proposed approach TSNN_5ph, the best result ($PAcc = 94.31\%$ and $WAcc = 70.52\%$) was reported when the input sequence consisted of 15 graphemes (i.e., $x_1 = 7$). In addition, TSNN_5ph usually outperformed TSNN_3ph when x_1 was greater than four.

In terms of the WER of the OOV words, Table 3 shows that TSNN usually produces less erroneous words than both baseline approaches. Further, the values obtained for PAcc are always higher than 90%, so a small difference in PAcc has a strong impact on the result of WAcc because we had

[†]FANN Library: http://leenissen.dk/fann/wp/

[††]FANN Datatypes: http://leenissen.dk/fann/html/files/fann_data-h.html#fann_activationfunc_enum

[†††]NIST sclite scoring toolkit: http://www.nist.gov/speech/tools/

Table 4 Training time of the four proposed test sets. Both the minimum and maximum durations of each epoch during the training of each neural network stage are described here.

		First-stage		Second-stage	
	Con text	Best epoch	Time/epoch (minutes) [min, max]	Best epoch	Time/epoch (minutes) [min, max]
Baseline1 (1st-stage only) & Baseline2 (both stages)	**x3=5**	49	[15, 16]	10	[13, 14]
	x3=6	41	[20, 22]	13	[13, 14]
	x3=7	46	[26, 27]	10	[13, 14]
	x3=8	51	[21, 22]	16	[13, 14]
	x3=9	48	[38, 42]	8	[13, 14]
TSNN_3ph	**x2=4**	137	[15, 20]	35	[8, 9]
	x2=5	81	[20, 27]	27	[8, 9]
	x2=6	56	[22, 39]	24	[8, 9]
	x2=7	56	[27, 34]	46	[9, 10]
	x2=8	57	[45, 60]	24	[9, 10]
TSNN_5ph	**x1=3**	94	[10, 16]	54	[61, 69]
	x1=4	183	[15, 24]	16	[64, 66]
	x1=5	111	[17, 25]	13	[53, 54]
	x1=6	81	[23, 33]	10	[52, 53]
	x1=7	76	[19, 40]	10	[52, 54]

surmised that most of the erroneous words (more than 19%) contains just one erroneous phoneme.

The training time of each stage model is also reported in Table 4. Because neural networks were used in the experiments, the training time must be calculated and separated epoch-by-epoch (1 epoch = 1 training iteration). From one to another epoch, we observed that the training time usually increases incrementally, so we decided to report two different values of time; specifically, the minimum and maximum training times. The minimum training time is measured around the first epoch, while the maximum training time is measured around the best epoch. In this work, the best epoch refers to a selected epoch where the trained model has the set of weights that will provide the best generalization performance, which is usually found at any epoch that will provide the smallest value of Mean Squared Error (MSE) evaluated on the testing data.

Theoretically, the training time of each test set depends exactly on the size of each staged neural network. For example, except for the case of $x_1 = 7$ and $x_3 = 8$, when the value of x is increased, Table 4 shows that the training time per epoch at the first-stage neural network also increased. Otherwise, it does not affect the second-stage at all because it is independent of the value of x. Based on the architecture of the second-stage neural network, both TSNN_3ph and Baseline2 use the same number of neurons and model configurations, but they provide different training times. Hence, we can assume that the training time also depends on the PC performance. Since we trained the model on a shared server (Windows 7 professional 64 bits, Core i7-3930K 3.20 GHz, 32.0 GB) in our laboratory, the training process was sometimes slow or fast depending on the number of user connections and the number of simultaneous training processes launched from the same client PC. Furthermore, the training time per epoch of the second-stage of TSNN_5ph appears too long compared to others because we could not load all the training data at once caused by the memory limitations, so the training dataset had to be decomposed into two or three parts at this stage. For each epoch, those parts were randomly selected one-by-one to be loaded, shuffled, trained, evaluated, and then deleted. As a result, the training time increased proportionately.

Table 5 Example of the words selected consisting of two phoneme conflicts ("R"→ {/ER/, /R/} and "A"→ {/EY/, /AH/}), while $x = 3$ and $y = 2$.

Word			Corresponding phonemes		
	$seq(g_i, 3)$			$seq(p_i, 2)$	
COLL	ABORATE	D	K AH _ L AE	B _ ER EY T	_
EL	ABORATE	S	AH L AE	B _ ER EY T	_ S
EL	ABORATE		AH L AE	B _ R AH T	_
EL	ABORATE	LY	AH L AE	B _ R AH T	_ L IY

Table 6 Accuracy given by TSNN and WFST based on two different datasets.

		TSNN_5ph ($x_1 = 7$)	WFST
Newly aligned CMUDict.	PAcc	94.31%	93.46%
	WAcc	70.52%	73.45%
Words produce the phoneme conflicts	PAcc	93.60%	91.99%
	WAcc	57.50%	57.05%

4.4 Comparing with a Previous Approach

In addition to the evaluation results in the previous section, we also compared our proposed approach to one of the most popular approaches in G2P conversion –the Weighted Finite-Stage Transducer (WFST)-based approach [15] available in the Phonetisaurus G2P toolkit[†].

We compared TSNN and WFST using two different datasets: a general dataset (i.e., the newly aligned CMUDict corpus) and a special dataset (i.e., a small subset of the newly aligned CMUDict corpus) in which only the words consisting of more than one phoneme conflicts, as seen in Table 5, are selected. For the first dataset, the training and testing data were the same as in our previous experiments, which have already been described in Sect. 4.2.1. For the second dataset, we randomly selected 80% and 20% of the total 7,123 extracted words for the training and testing datasets, respectively.

The results in Table 6 show that TSNN_5ph always provides higher phoneme accuracy than WFST, but, unfortunately, lower word accuracy for the first dataset.

5. Discussion

The experimental results depicted in Fig. 6 and Table 3 clearly show that the proposed two-stage neural network-based approach for G2P conversion usually provides the best accuracy on OOV words compared to both baseline approaches, even when it uses a smaller grapheme sequence size than others (i.e., $x_1 < x_3$).

[†]Phonetisaurus toolkit: http://code.google.com/p/phonetisaurus/

As explained in Sect. 4.2.2, at the input layer of the first-stage of the G2P conversion model, the exact number of graphemes and phonemes getting involved in Baseline2 and TSNN_5ph were *quite similar to each other*, but both approaches provided different results; Fig. 6 and Table 3 indicate that TSNN_5ph usually provided a higher performance than Baseline2.

Even when we decreased the value of y from two down to only one (i.e., reduced the size of phoneme sequence from five down to only three phonemes per sequence) in order to have *the same size phonemic pattern (e.g., a pattern of nine phonemes)* at the input layer of the second-stage of both approaches mentioned, our proposed approach (i.e., TSNN_3ph) still outperformed Baseline2. Therefore, it does not matter if the same numbers of phonemes are used or not, the two-stage neural network-based approach for G2P conversion always outperformed both baseline approaches. This can result in the assumption that the grapheme and phoneme contexts are not really effective to fix the problem of conflicting phonemes at the output layer of the G2P conversion model, unless the pattern of joint phoneme sequences is incorporated at the second-stage.

The comparison between the results given by Baseline1 and Baseline2 also demonstrates that the second-stage neural network is very helpful in boosting the accuracy of the G2P conversion model to the next level. Even if the phoneme context information is absent in Baseline2, it is still possible to go beyond the performance attainable by Baseline1, by assigning the value of z to a positive number (e.g., $z = 4$) at the second-stage neural network. Perhaps this technique may also help to improve the performance of existing approaches, such as the joint-sequence model, by creating a hybrid model that integrates the approach into our two-stage model-based G2P conversion.

Further, following the error analysis of the erroneous words, some invisible information was discovered. For example, some extracted graphemes-phonemes pairs in the testing dataset (i.e., OOV words) were never seen during the training process, so the wrong output phonemes were given during the evaluation. In addition, most of the erroneous words containing more than one erroneous phonemes per word were from foreign words such as "SENZAKI" and "AICHI" from Japanese, "BOGDANOWICZ" from Polish, "XIAOGANG" from Chinese, etc.

Conversely, the evaluation results of comparison with another existing approach, the WFST-based approach, in Table 6 demonstrate that our approach provides higher phoneme accuracy but lower word accuracy on the first dataset. However, when the training data contain only words with some phoneme conflicts, our approach yields a better performance than WFST. This shows that the two-stage neural network-based approach is good at identifying the single phoneme in a word by using the grapheme and phoneme contexts differently from previous approaches, especially when a phoneme conflict has occurred. Otherwise, since it does not use any language model-based technique, it lacks knowledge for detecting the whole word compared to the WFST-based approach. Therefore, for the next step of improvement, we have to focus on how to reduce the erroneous words containing only one erroneous phoneme in order to increase the word accuracy.

6. Conclusion and Future Work

This paper has shown that using only one neural network is not enough for solving some complicated problems in G2P conversion. As a result, the two-stage neural network is considered a powerful approach for improving the accuracy of the G2P conversion model. To output the phonemes of the input text, prediction must be based on phonemic rather than graphemic information. Because two different neural networks and OBC encoding algorithm are used, this approach is also counted as an expensive and time-consuming approach, but it can also provide good results while performing on a large and complex corpus such as the auto-aligned CMUDict [1]. In terms of phoneme and word accuracy, the evaluation results show that our proposed approach usually outperforms the baselines and it also can be regarded as an improved version of the single-stage neural network-based approach for G2P conversion.

In the future, we plan to incorporate a pseudo-phoneme based technique [16] and the graphones based technique [14] into our approach to reduce the conflicting problems between phonemes at the first-stage neural network. A syllable-based approach may also help to improve the performance of the model as well [20], [21]. Moreover, instead of using the FANN library, we will try to implement each stage model using other machine learning toolkits such as the RNNLM toolkit[†] or OpenANN[††]. A hybrid model mentioned in the previous section will be taken into consideration as well, so that we will probably be able to boost the word accuracy of our proposed approach to another higher level.

Acknowledgments

This work is supported by a Grant-in-Aid for Young Scientists (B) 24700167 2012 and for Scientific Research (B) 22300060 2012 from MEXT, Japan, the Strategic Information and Communications R&D Promotion Programme by MIC, Japan, and the Kayamori Foundation of Information Science Advancement.

References

[1] Auto-aligned CMUDict corpus, "Letter-to-phoneme conversion challenge: 10 folds datasets," PASCAL website [Online], http://pascallin.ecs.soton.ac.uk/Challenges/PRONALSYL/Datasets, accessed Nov. 2012.

[2] R. Damper, "Learning about speech from data: Beyond NETTalk," in Data-Driven Techniques in Speech Synthesis, pp.1–25, Kluwer Academic Publishers, 2001.

[†]RNNLM toolkit: http://www.fit.vutbr.cz/~imikolov/rnnlm/

[††]OpenANN: http://openann.github.io/OpenANN-apidoc/

[3] M. Davel and E. Barnard, "Pronunciation prediction with default&refine," Computer Speech and Language, ScienceDirect, ELSEVIER, Jan. 2008.
[4] E.B. Bilcu, "Text-to-phoneme mapping using neural networks," Tampere, Oct. 2000.
[5] E.B. Bilcu and J. Astola, "Neural networks with random letter codes for text-to-phoneme mapping and small training dictionary," 14th EUSIPCO, Florence, Italy, Sept. 2006.
[6] P.C. Bagshaw "Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression," Computer Speech and Language, vol.12, pp.119–142, 1998.
[7] P. Taylor, "Hidden Markov models for grapheme to phoneme conversion," Proc. Interspeech, 2005.
[8] G. Miller, Language and Speech, p.49, W.H. Freeman and Company, San Francisco, 1981.
[9] S. Jiampojamarn, K. Grzegorz, and T. Sherif, "Applying many-to-many alignments and hidden Markov models to letter-to-phoneme conversion," Proc. NAACL HLT, Rochester, New York, April 2007.
[10] S. Jiampojamarn and K. Grzegorz "Online discriminative training for grapheme-to-phoneme conversion," Proc. Interspeech, Brighton, Sept. 2009.
[11] T. Rama, A.K. Singh, and S. Kolachina, "Modeling letter-to-phoneme conversion as a phrase based statistical machine translation problem with minimum error rate training," Proc. NAACL HLT Student Research Workshop and Doctoral Consortium, Colorado, June 2009.
[12] V. Claveau, "Letter-to-phoneme conversion by inference of rewriting rules," Proc. Interspeech, UK, Sept. 2009.
[13] K.U. Ogbureke, C. Peter, and B.C. Julie, "Hidden Markov models with context-sensitive observations for grapheme-to-phoneme conversion," Proc. Interspeech, Japan, 2010.
[14] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," Speech Commun., vol.50, pp.434–451, 2008.
[15] J.R. Novak, P.R. Dixon, N. Minematsu, K. Hirose, C. Hori, and H. Kashioka, "Improving WFST-based G2P conversion with alignment constraints and RNNLM N-best rescoring," Proc. Interspeech, Portland, Oregon, 2012.
[16] M. Davel and E. Barnard, "Extracting pronunciation rules for phonemic variants," ISCA Tutorial and Research Workshop on Multilingual Speech and Language Processing, 2006.
[17] J.A. Bullinaria, "Text-to-phoneme alignment and mapping for speech technology: A neural networks approach," 2011 International Joint Conference on Neural Networks (IJCNN), San Jose, CA, 2011.
[18] S. Jiampojamarn and G. Kondrak, "Letter-phoneme alignment: An exploration," Proc. 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, July 2010.
[19] L. Patrick, H. Stefan, G.V. Andrei, and N. Hermann, "Hidden conditional random fields with M-to-N alignments for grapheme-to-phoneme conversion," Proc. Interspeech, Portland, 2012.
[20] M. Libossek and F. Schiel, "Syllable-based text-to-phoneme conversion for German," Proc. ICSLP, pp.283–286, 2000.
[21] S. Bartlett, G. Kondrak, and C. Cherry, "Automatic syllabification with structured SVMs for letter-to-phoneme conversion," Proc. ACL HLT, Association for Computational Linguistics, Ohio, USA, pp.568–576, June 2008.
[22] K. Seng, Y. Iribe, and T. Nitta, "Letter-to-phoneme conversion based on two-stage neural network focusing on letter and phoneme contexts," Proc. Interspeech, Florence, Italy, pp.1885–1888, 2011.
[23] K. Seng, K. Katsurada, Y. Iribe, and T. Nitta, "Improving the performance of letter-to-phoneme conversion by using two-stage neural network," IPSJ SIG Technical Report, Information Processing Society of Japan, 2012.

Seng Kheang received a B. Eng. degree in Computer Science from Institute of Technology of Cambodia, Cambodia. In 2011, he obtained an M. Eng. degree from Toyohashi University of Technology (TUT), Japan. Since October 2012, he has been a doctoral student at the Department of Computer Science and Engineering, Toyohashi University of Technology. His research interests include text-to-phoneme conversion for speech synthesis, natural language processing, and spoken term detection.

Kouichi Katsurada received a Ph. D. degree from Osaka University in 2000. He has been with Toyohashi University of Technology since 2000. He is currently an Associate Professor at the Center for International Relations and Department of Computer Science and Engineering, Toyohashi University of Technology. His research interests include multimodal interaction, facial image processing, and spoken term detection. He is a member of IEEE, ISCA, IPSJ, JSAI, and ASJ.

Yurie Iribe received M. S. and Ph. D. degrees from the Graduate School of Human Informatics of Nagoya University, Japan. She worked as an Associate Professor at the Information and Media Center, Toyohashi University of Technology, until 2013. She is currently an Associate Professor in the Information Science Department, Aichi Prefectural University, Japan. Her recent research interests include education support and speech recognition. She is a member of ISCA, IPSJ, JSAI, and ASJ.

Tsuneo Nitta received Ph. D. degree from Tohoku University, Japan. He worked at the R&D Center and Multimedia Eng. Lab. Of Toshiba Corp. from 1970 to 1998. He subsequently joined Toyohashi University of Technology as a Professor in the Graduate School of Engineering. In 2012, he became a TUT Professor Emeritus and a visiting Professor both at TUT and Waseda University. His current research interests include speech recognition, speech synthesis, and multimodal interaction. He is an IPSJ Fellow and a member of IEEE, JSAI, and ASJ.