Improving the performance of Letter-To-Phoneme conversion by using Two-Stage Neural Network

KHEANG SENG †1 KOUICHI KATSURADA †1
YURIE IRIBE †1 TSUNEO NITTA †1,2

Abstract: In Text-To-Speech (TTS), Letter-To-Phoneme (L2P) conversion is one of the most important tasks, which allows converting automatically from arbitrary text into the corresponding phoneme sequence. According to the existing researches, the performance is already quite good for the in-vocabulary words, but not for the Out-Of-Vocabulary (OOV) words. For improving the performance of L2P conversion on OOV words, this paper focuses on two different issues. The first issue concerns the unknown relationship between letter and phoneme, while the second one is related to a specific difficult problem where a letter sequence could correspond to another phoneme sequence in the same context. Therefore, we introduce a L2P conversion based on two-stage neural network approach focusing on both letter and phoneme contexts. The first-stage neural network is implemented as a many-to-many mapping model between letters and phonemes for solving the first issue, while the second-stage neural network aims to deal with the second problem by extending the context information at the phonemic level in order to generate a pattern of phonemes that could be easily recognized by the neural network. As a result, based on the auto aligned CMU corpora [1], it is proved that our proposed approach could provide a high performance in terms of Phoneme Accuracy (PAcc) and Word Accuracy (WAcc) on the OOV words.

Keywords: Letter-To-Phoneme conversion, two-stage neural network, many-to-many mapping between letters and phonemes

1. Introduction

In Text-To-Speech (TTS) system, a L2P module plays an important role during the automatic conversion of texts into phoneme sequences [2]. Moreover, it has also been used by various applications such as Computer Assisted Language Learning (known as CALL system), Spoken Term Detection (known as STD), etc.

Since the traditional approach (i.e. dictionary lookup) had no ability to produce any new phoneme sequences from the OOV words, many beneficial data-driven approaches have been widely used with many interests such as the Decision Tree, Artificial Neural Network (ANN) [3, 4], Hidden Markov Model (HMM) [5, 6, 7], rewriting rules [8], probabilistic and statistical approach [9], etc. Practically, a sequence of letters had been used instead of a single letter, based on a many-to-one mapping algorithm, for predicting a corresponding phoneme [10, 5, 6]. However, it was still impossible to deal with a specific difficult problem where a letter sequence could be mapped to several possible phonemes in the same context. Then, a many-to-many mapping between letters and phonemes has been proposed instead and widely used by many researchers for solving the mentioned problem as well as other unpredicted problems during the converting process [9, 11, 6].

For example, Taraka et al. treated the L2P conversion problem as a phrase-based statistical machine translation problem [9]. They removed the one-to-one alignments from the auto aligned CMU corpora [1] and induced their own alignments using GIZA++ available as a part of the MOSES toolkit. As a result, by using Minimum Error Rate Training (MERT) and A* beam search decoder, they reported 91.4% as the average of PAcc and 63.81% the averaged of WAcc. Based on the same aligned CMU corpora, the L2P conversion by inference from rewriting rules [8] provided 74.40% as WAcc by measuring in terms of word precision averaged on the whole dataset (Training and testing data). On the other hand, the HMM-based approach with context-sensitive observations for grapheme-to-phoneme conversion [6], proposed in 2010, showed a strong interest in the use of context information at both letter and phonemic levels. The authors also mentioned that different corpora always provided different performances since they obtained ~79.79% as WAcc on the Unilex corpora (UK English words), but only ~57.85% as WAcc on the above mentioned CMU corpora (American English words) which contains many loan words and errors.

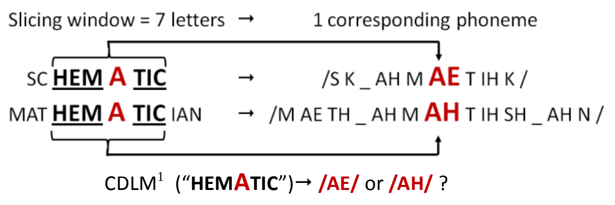

Slicing window = 7 letters →
1 corresponding phoneme
SC HEM A TIC → /S K _ AH M AE T IH K /
MAT HEM A TIC IAN → /M AE TH _ AH M AH T IH SH _ AH N /
CDLM1 ("HEMATIC")→ /AE/ or /AH/ ?

Figure 1: *Conflict output phonemes of the letter 'A' on sequence "HEMATIC", while "HEM" and "TIC" respectively represent the left and right contexts of the letter 'A'.*

According to Figure 1, the conflicts between output phonemes are still found even if the phoneme context is used. For example, if we take 7 letters as input and 5 phonemes as output, the letter sequence "HEMATIC" still has two possible phoneme sequences (e.g. /AH M AE T IH/ and /AH M AH T IH/), so we cannot know which one is the best phoneme sequence given "HEMATIC" as input. Because of such a kind of problem, the machine learning cannot have ability to learn efficiently the pronunciation rules, especially for the case of American English language which consists of many loan words from other languages.

†1 Toyohashi University of Technology
†2 Waseda University

[1] CDLM (Context-Dependent Letter Model) allows predicting a phoneme by observing an input letter sequence (left context + a target letter + right context) and focusing on a target letter which is located at the middle position of sequence.

In order to improve the performance of L2P conversion on the OOV words, this paper focuses on two different problems including 1) the unknown relationship between letters and phonemes and 2) a specific problem where a letter sequence could correspond to another phoneme or phoneme sequence in the same context. Therefore, this paper proposes a L2P conversion based on two-stage neural network approach focusing on both letter and phoneme contexts, which can deal with the mentioned problems effectively. The first-stage neural network is implemented as a many-to-many mapping model between letters and phonemes for solving the first problem, while the second-stage neural network aims to deal with the second problem.

The reminder of the paper is organized as follows: in the next section, we will present the re-alignment process for preparing the newly training and testing datasets. Then, we will introduce our proposed approach by reporting the experimental results in the following section. At the end, the errors analysis, discussions and conclusions will be presented successively.

2. Data preparation (L-P alignments)

The auto aligned CMU corpora (also called pronunciation dictionary or dataset) [1] have been used by many researchers, even though it contains many wrong letter-phoneme alignments and errors [6]. Obviously, the inconsistency of the aligned corpora could affect badly to the accuracy of the L2P conversion, so various techniques of L2P alignment [9, 12, 13] has been induced by many researchers in order to prepare the consistent and well-aligned corpora. Therefore, we decided to use the same CMU corpora in our experiments and induced our own L2P alignment as well.

2.1 Auto aligned CMU corpora

The aligned CMU corpora are based on the American English language which contains many acronyms and loan words from French, German, Japanese, etc. 34 letters including 7 numeric characters and 40 phonemic units are seen in the corpora. The CMU corpora contain totally ~112,102 words including 838,996 letters and 838,996 phonemes, which is originally divided into 10 folds (e.g. part0, part1 … part 9). Each fold consists of almost the same amount of words, letters as well as phonemes, as shown in Figure 2 and 3.

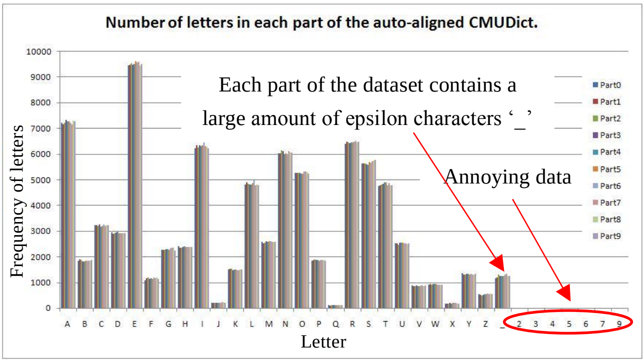


Figure 2: *Amount of letters inside each fold of the original dataset*

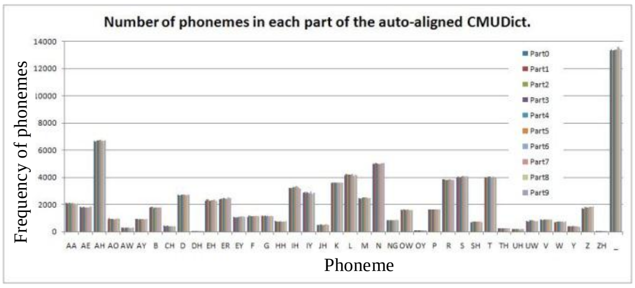


Figure 3: *Amount of phonemes inside each fold of the original dataset*

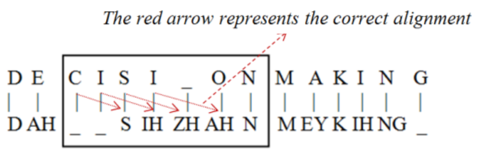

The red arrow represents the correct alignment
D E C I S I _ O N M A K I N G
D AH _ _ S IH ZH AH N M EY K IH NG _

Figure 4: *Wrong mapping between letters and phonemes*

Case of alignment (Subword)	Auto-aligned CMU dictionary.		Newly aligned CMU dictionary *(GIZA++ & Manual Check-up)*	
-LE {-BLE, -TLE, -PLE, …}	C O P **P L E** K AA P **AH L _**		C O P **P _ L E** K AA 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 _**	B I A S **E D** B AY AH S **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**
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 _ _**	

Table 1: *A comparison of the well-alignment between letters and phonemes inside the original dataset (column 2) and the new dataset (column 3)*

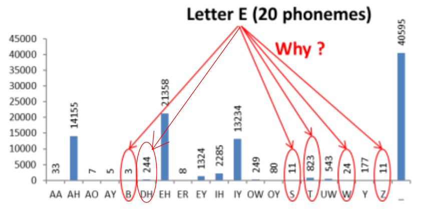


Figure 5: *The wrong mapping between the input letter 'E' and its corresponding phonemes, found inside the auto aligned CMU corpora*

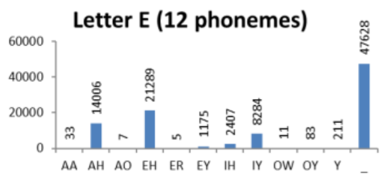


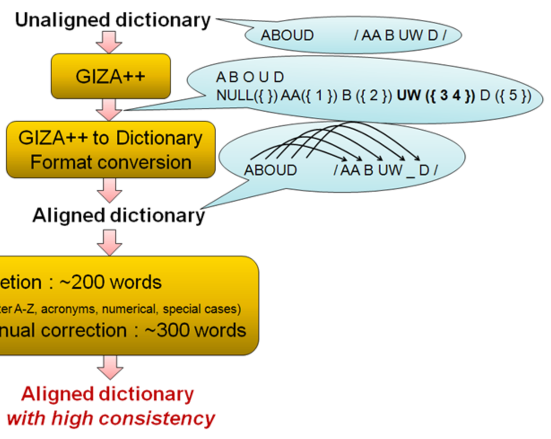
Figure 6: *The mapping between the input letter 'E' and its corresponding phonemes, found inside the newly obtained corpora*

After we conducted the analysis on the auto aligned CMU corpora, we figured out that three main factors cause the inconsistency of pronunciation and reduce the performance of L2P conversion; (1) There are many words containing the wrong mapping pairs as depicted in Figure 4 and Table 1. For example, Figure 5 shows that the letter 'E' which is theoretically used to represent the vowel could map to different 20 phonemes including some incompatible phonemes, such as, 'E'→/B/, 'E'→/DH/, 'E'→/S/, 'E'→/T/, 'E'→/Z/; (2) It also consists of a large amount of unnecessary or noisy data like epsilon letters/phonemes '_' as shown in Figure 4 and Table 1; (3) The corpora contain a small amount of annoying or low frequency data like the numeric characters.

2.2 L2P alignment using GIZA++

Since we found many errors as mentioned in the previous section, we decided to prepare a newly consistent dataset/corpora based on the existing one.

By following the idea of R. Taraka et al. [9], we firstly removed the one-to-one alignments from the original dataset by deleting all the epsilon letters/phonemes. Then, we conducted our own alignment by using GIZA++ open source toolkit[2] as shown in Figure 7. Additionally, a manual check-up has also been included into the process in order to ensure a high consistency and quality of the obtained dataset, which removed most of the abnormal words with low frequency (e.g. JORGE→/*HH* AO R HH EY/), some acronyms (e.g. CNN, MSGR, MRS), and the annoying information from the dataset. In total, about 200 words were deleted while the wrong letter-phoneme mapping pairs within other 300 words were manually corrected.


Unaligned dictionary
ABOUD /AA B UW D/
GIZA++
A B O U D
NULL({ }) AA({ 1 }) B ({ 2 }) UW ({ 3 4 }) D ({ 5 })
GIZA++ to Dictionary Format conversion
ABOUD /AA B UW _ D/
Aligned dictionary
Deletion : ~200 words
(Letter A-Z, acronyms, numerical, special cases)
Manual correction : ~300 words
Aligned dictionary with high consistency

Figrue 7: *Scenario of the L2P alignment*

2.3 Comparison between the original and new dataset

After around 200 words were deleted, the new dataset contains fewer words than the original one, but it is much more reliable than the original one in terms of quality and consistency. The comparison between both datasets was done based on 3 different parameters including (1) the word length, (2) the number of possible phonemes to be mapped by each letter and (3) the low frequency of the epsilon letters/phonemes.

According to our experimental results, Table 1 shows clearly that the words located in the third column (i.e. the new corpora) are always shorter and well-aligned than the one located in the second column (i.e. the original corpora). Moreover, by counting the phonemes that could be possibly mapped from each letter, Figure 8 shows that each letter inside the new dataset corresponds to fewer amounts of phonemes than the one inside the original dataset. For example, according to Figure 5 and 6, the number of phonemes corresponding to the letter 'E' is nicely reduced from 20 to only 12 phonemes.

On the other hand, after the alignment process, a large amount of epsilon letters/phonemes considered as unnecessary or noisy data was removed from our new dataset. In this case, it reduces a lot of noisy samples from the training and testing data. For example, less than 900 units are found in each fold of the new dataset, while more than 1000 units are found in each fold of the original dataset (as seen in Figure 2).

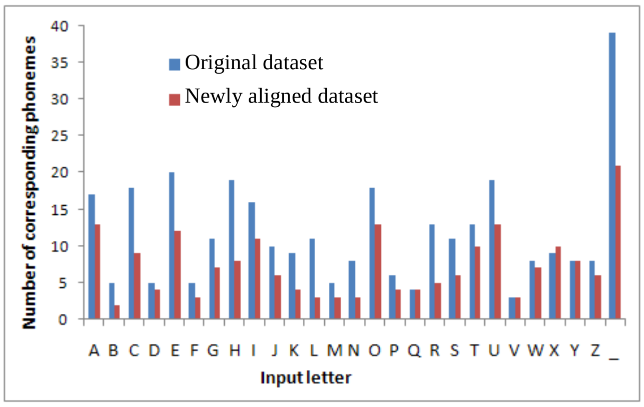


Figure 8: *Consistency measurement based on the number of corresponding phonemes that could be mapped by each letter inside the original and new corpora. This figure shows that most of the wrong mappings between letters and phonemes were corrected in order to maintain the quality of the aligned corpora.*

3. Two-Stage Neural Network approach

To improve the performance of L2P conversion on the OOV words, we have to focus on two difficult problems as described in the first section. In order to deal with these problems, our approach is implemented based on two main ideas as below.

The first idea is based on the fact that the relationship between letters and phonemes inside the corpora is unknown and there has neither standard rule of pronunciation, nor strict correspondence between the number of letters and the number of phonemes [14]. For this reason, many-to-many mapping algorithm seems to be the best way in representing the association between letters and phonemes because it can increase the number of pronunciation rules.

The second idea, so called Phoneme Sequences Pattern-Observation Model (PSPOM) is introduced in this paper for dealing with the problem where the conflict rules of pronunciation are found. This technique keeps extending the context information at the phonemic level in order to generate a pattern of phonemes that could be easily recognized by the machine learning (i.e. neural network).

[2] Source code of GIZA++ is available at *http://code.google.com/p/giza-pp/*

	A sequence of 7 letters				1 desired phoneme		A sequence of 5 phonemes			Word
	Left context	Focusing letter	Right context	→			Left Context	Desired phoneme	Right Context	
Rule 1	_ _ _	S	C H E	→	/S/	→	_ _	S	K _	SCHEMATIC ↓ /S K _ AH M AE T IH K /
Rule 2	_ _ S	C	H E M	→	/K/	→	_ S	K	_ AH	
Rule 3	_ S C	H	E M A	→	/_/	→	S K	_	AH M	
Rule 4	S C H	E	M A T	→	/AH/	→	K _	AH	M AE	
Rule 5	C H E	M	A T I	→	/M/	→	_ AH	M	AE T	
Rule 6	H E M	A	T I C	→	/AE/	→	AH M	AE	T IH	
Rule 7	E M A	T	I C _	→	/T/	→	M AE	T	IH K	
Rule 8	M A T	I	C _ _	→	/IH/	→	AE T	IH	K _	
Rule 9	A T I	C	_ _ _	→	/K/	→	T IH	K	_ _	
Rule 10	_ _ _	M	A T H	→	/M/	→	_ _	M	AE TH	MATHEMATICIAN ↓ /M AE TH _ AH M AH T IH SH _ AH N/
Rule 11	_ _ M	A	T H E	→	/AE/	→	_ M	AE	TH _	
Rule 12	_ M A	T	H E M	→	/TH/	→	M AE	TH	_ AH	
Rule 13	M A T	H	E M A	→	/_/	→	AE TH	_	AH M	
Rule 14	A T H	E	M A T	→	/AH/	→	TH _	AH	M AH	
Rule 15	T H E	M	A T I	→	/M/	→	_ AH	M	AH T	
Rule 16	H E M	A	T I C	→	/AH/	→	AH M	AH	T IH	
Rule 17	E M A	T	I C I	→	/T/	→	M AH	T	IH SH	
Rule 18	M A T	I	C I A	→	/IH/	→	AH T	IH	SH _	
Rule 19	A T I	C	I A N	→	/SH/	→	T IH	SH	_ AH	
Rule 20	T I C	I	A N _	→	/_/	→	IH SH	_	AH N	
Rule 21	I C I	A	N _ _	→	/AH/	→	SH _	AH	N _	
Rule 22	C I A	N	_ _ _	→	/N/	→	_ AH	N	_ _	

Table 2: *List of the pronunciation rules generated by mapping from several letters to one or several phonemes*

In this paper, the implementation of L2P model is based on a two-stage neural network approach. The first-stage neural network is implemented as a many-to-many mapping model between letters and phonemes for overcome the first problem, while the second-stage neural network aims to deal with the second problem by using PSPOM.

In this section, we will describe the functionality of the first and second stage neural networks. Then, we will show how the two-stage neural network approach deals with the problems.

3.1 First-stage neural network

The first stage neural network aims to create a many-to-many mapping model between letters and phonemes for getting over the first problem where the association between letters and phonemes cannot be easily defined.

This neural network is basically trained by extracting the rules of L2P conversion (i.e. pronunciation rules) from all the elements (i.e. words) inside the training data by using a slicing window [3]. The training process is done by passing the slicing window through each word letter-by-letter or phoneme-by-phoneme from the first until the last position as shown in Table 2. According to the many-to-many mapping algorithm between letters and phoneme, each extracted rules consists of a sequence of M letters (where a target letter is in the middle of sequence) as input and another sequence of N phonemes (where a desired phoneme is in the middle of sequence) as output. For example, according to the rule 6 in Table 2, we can see that the letter sequence “HEMATIC” in which ‘A’ represents the target letter can be automatically converted into the phoneme sequence /AH M **AE** T IH/ in which /AE/ represents the desired phoneme, while fixing the size of input and output slicing windows to 7 letters and 5 phonemes, respectively.

In this stage neural network, the automatic L2P conversion would work perfectly if and only if there had no confliction among the extracted rules during the training period. Otherwise, the performance of L2P conversion will be relatively reduced as long as many conflict rules could be found. Regarding to Table 2, there is totally 22 conversion rules extracted from the word ‘SCHEMATIC’ and ‘MATHEMATICIAN’, in which two rules are conflicted (i.e. rule 6 and rule 16). As a result, the first-stage neural network is so difficult to produce a right phoneme sequence given ‘HEMATIC’ as input. However, such a problem will be easily determined by using the second-stage neural network.

3.2 Second-Stage Neural Network

For the second-stage neural network, it aims to overcome a specific difficult problem where a letter sequence could correspond to another phoneme sequence in the same context by using PSPOM. According to the section 3.1, such a mentioned problem was occurred when some conflict rules are found inside the training data at the first-stage neural network.

The basic idea implemented at the second-stage neural network is to extend the context information at the phonemic level in order to generate two distinct patterns of phonemes that could be easily recognized by the neural network. In this case, the extension of the phoneme context is based on the PSPOM which enables to predict a phoneme by observing on a concatenation among the current sequence of phonemes with its neighborhood sequences. As seen in Figure 9, the second-stage neural network predicts a desired phoneme by observing five connected sequences of five phonemes (i.e. seq-2, seq-1, seq. as the current sequence, seq+1 and seq+2).

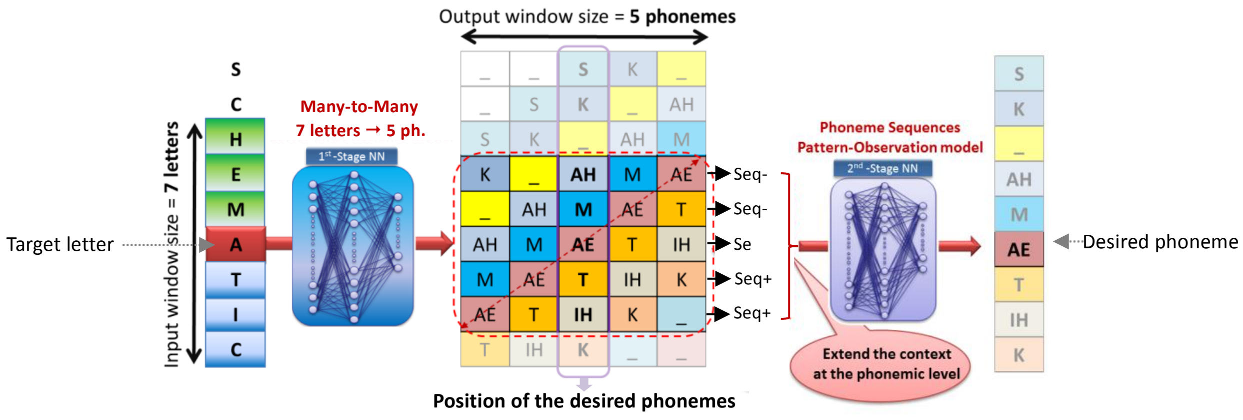

Output window size = 5 phonemes
Many-to-Many
7 letters → 5 ph.
1st-Stage NN
Target letter
Input window size = 7 letters
Seq-
Seq-
Se
Seq+
Seq+
Phoneme Sequences Pattern-Observation model
2nd-Stage NN
Extend the context at the phonemic level
Desired phoneme
Position of the desired phonemes

Figure 9: Architecture of a L2P conversion based on Two-Stage Neural Network focusing on both letter and phoneme contexts

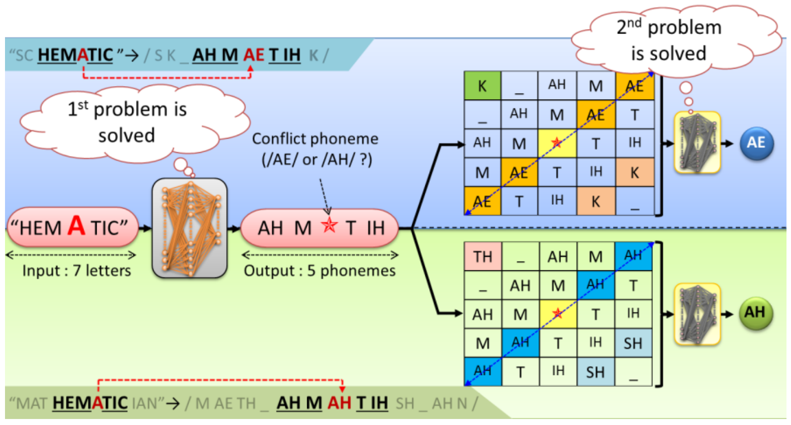

2nd problem is solved
1st problem is solved
Conflict phoneme (/AE/ or /AH/ ?)
"HEM A TIC"
Input : 7 letters
AH M ☆ T IH
Output : 5 phonemes
"SC HEMATIC"→ / S K _ AH M AE T IH K /
"MAT HEMATIC IAN"→ / M AE TH _ AH M AH T IH SH _ AH N /

Figure 10: *How does our approach solve the conflict phonemes?*

Based on the same example in the previous section, the first-stage neural network could not decide a phoneme sequence corresponding to the letter sequence "HEMATIC" since two different rules were found (i.e. rule 6 and 16). However, according to Figure 10, the second-stage neural network is able to produce two different phonemes (i.e. /AE/ and /AH/) from the same letter sequence (i.e. "HEMATIC") by observing on two different patterns of phonemes. As the output window contains five phonemes, so the five connected sequences are then used as input at the second-stage neural network.

This shows that the phoneme context is very useful for our L2P conversion based on two-stage neural network approach.

4. Experimental results

4.1 Datasets

For our experiments, the new corpora or dataset is divided into 10 folds by respecting the original corpora. Then, 9 folds were used as a training dataset (100,713 words or 750,198 samples), while other was used as a testing dataset (11,188 words or 83,267 samples). We employ phoneme accuracy (PAcc) and word accuracy (WAcc) to evaluate the performance of L2P conversion on the OOV words.

As the epsilon letter and epsilon phoneme are also counted, there are totally 27 letters and 40 phonemic units. Each element is encoded by using the numerical vectors such as Orthogonal Binary Codes (OBC) [4]. Thus, each letter is represented by 27 bits (or 27 neurons) while each phoneme is represented by 40 bits (or 40 neurons).

4.2 Configuration of FANN parameters

Each stage neural network has been implemented based on the functions available in FANN library[3]. After conducting some preliminary experiments, we found the best result has been given when the following configurations are used:

- Standard FANN with 3 layers
- Number of neurons at the first-stage neural network:
  - Input layer = M letters * 27 neurons
  - Hidden Layer = M letters * 27 neurons * 2
  - Output layer = N phonemes * 40 neurons
- Number of neurons at the second-stage neural network:
  - Input layer = N sequences * N phonemes * 40 neurons
  - Hidden layer = N sequences * N phonemes * 40 neurons * 2
  - Output layer = 1 phoneme * 40 neurons
- M is the size of letter sequence, while N is the size of phoneme sequence
- Learning rate = 0.8 ; Momentums = 0.1

3 FANN (Fast Artificial Neural Network): http://leenissen.dk/fann/wp/

4.3 Output results

The performance of L2P conversion is usually measured based on the phoneme accuracy (PAcc) and word accuracy (WAcc). Theoretically, PAcc is a measure per phonemic unit, while WAcc is a measure which counts a word as correct if all the phonemes belonging are correct.

During the experiments, our proposed approach used two different-sized neural networks. The first-stage neural network takes a sequence of M letters as input and a sequence of five phonemes as output, while the second-stage neural network takes five sequences of five phonemes as input and only one phoneme as final output. In order to evaluate the performance of our proposed approach, we choose an alternative baseline approach which is implemented based on only a single neural network and similar to the original NETTalk system [2]. The baseline approach takes the same sequence of M letters as input but only 1 phoneme as output. Furthermore, the number of hidden neurons is two times bigger than the number of neurons at the input layers (which is the same configuration to the first-stage neural network).

The goal of this paper is to improve the performance of L2P conversion on the OOV words, so we are not really interested in the results given by the in-vocabulary (Seen) words. According to Figure 11, our proposed approach always provides higher accuracy (i.e. PAcc and WAcc) than the baseline approach.

However, by extending the size of input sequence from 7 letters until 19 letters, we can see that the accuracy (i.e. PAcc and WAcc) on the OOV words given by both approaches (i.e. baseline approach and two-stage neural network approach) is increased relatively to the size of letter sequence. For both approaches, the WAcc on the OOV words increases so nicely when the size of letter sequence increase from 7 to 11 letters. However, in terms of WAcc, the best results are reported when the input sequence contains 19 letters; the L2P conversion based on two-stage neural network approach provides 70.62% on the OOV words and 85.58% on the in-vocabulary words, while the baseline approach provides only 67.56% on the OOV words and 83.04% on the in-vocabulary words.

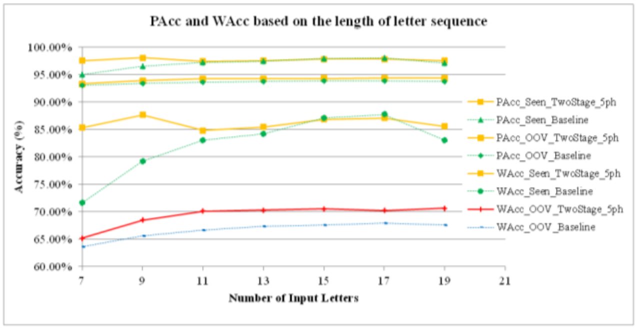


Figure 11: *PAcc and WAcc of the baseline approach and our approach.*
Baseline: 1 ANN (M letters→1 ph.)
*Our approach: 2 ANNs (M letters→5 ph.) & (5ph. * 5 sequences→1 ph.)*
In this research, we are strongly interested in the results of WACC on the OOV words (i.e. the red and blue line)

5. Discussions

Based on the results of our experiments as shown in Figure 11, it is proved that the corpora with high quality are very helpful in creating an accurate L2P model since all PAcc for both approaches (i.e. baseline approach and Two-Stage Neural Network approach) are always higher than 92%.

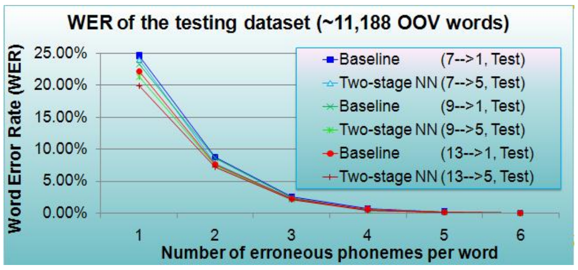


Figure 12: *Word Error Rate (WER) on the OOV words grouped by the number of erroneous phonemes per word*

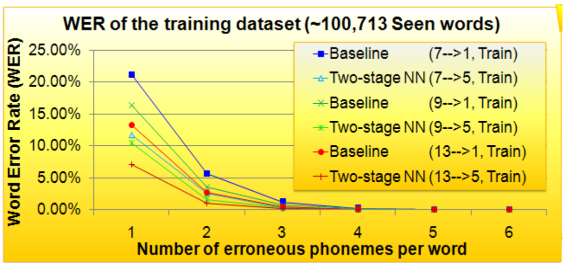
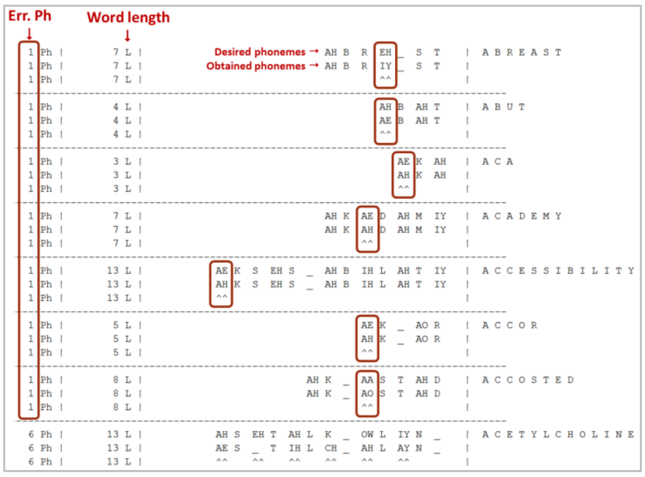


Figure 13: *Word Error Rate (WER) on the in-vocabulary words grouped by the number of erroneous phonemes per word*



```
  1 Ph |      7 L |                     AH B  R  EH _  S  T     | A B R E A S T
  1 Ph |      7 L |                     AH B  R  IY _  S  T     |
  1 Ph |      7 L |                              ^^             |
------------------------------------------------------------------------
  1 Ph |      4 L |                              AH B  AH T     | A B U T
  1 Ph |      4 L |                              AE B  AH T     |
  1 Ph |      4 L |                              ^^             |
------------------------------------------------------------------------
  1 Ph |      3 L |                                 AE K  AH    | A C A
  1 Ph |      3 L |                                 AH K  AH    |
  1 Ph |      3 L |                                 ^^          |
------------------------------------------------------------------------
  1 Ph |      7 L |                     AH K  AE D  AH M  IY    | A C A D E M Y
  1 Ph |      7 L |                     AH K  AH D  AH M  IY    |
  1 Ph |      7 L |                           ^^                |
------------------------------------------------------------------------
  1 Ph |     13 L |   AE K  S  EH S  _  AH B  IH L  AH T  IY    | A C C E S S I B I L I T Y
  1 Ph |     13 L |   AH K  S  EH S  _  AH B  IH L  AH T  IY    |
  1 Ph |     13 L |   ^^                                        |
------------------------------------------------------------------------
  1 Ph |      5 L |                           AE K  _  AO R     | A C C O R
  1 Ph |      5 L |                           AH K  _  AO R     |
  1 Ph |      5 L |                           ^^                |
------------------------------------------------------------------------
  1 Ph |      8 L |                  AH K  _  AA S  T  AH D     | A C C O S T E D
  1 Ph |      8 L |                  AH K  _  AO S  T  AH D     |
  1 Ph |      8 L |                           ^^                |
------------------------------------------------------------------------
  6 Ph |     13 L |   AH S  EH T  AH L  K  _  OW L  IY N  _     | A C E T Y L C H O L I N E
  6 Ph |     13 L |   AE S  _  T  IH L  CH _  AH L  AY N  _     |
  6 Ph |     13 L |   ^^    ^^    ^^    ^^    ^^    ^^          |
```

Figure 14: *Example of some erroneous words*

Due to the fact that the phoneme accuracy is over than 90%, a small different value of PAcc has a strong impact on the result of WAcc because we find out that many erroneous words contains just one erroneous phoneme, according to Figure 12, 13 and 14.

On the other hand, after investigating with the erroneous words, some hidden information has been figured out. For example, most of the words containing more than 3 erroneous phonemes are from the loan words such as "SENZAKI", "AICHI", "BOGDANOWICZ", "XIAOGANG", "THOREAU", "BIJUR", etc.

These are the reasons why current researchers pay a lot more attention to the corpora before conducting their experiments. In order to improve the WAcc on the OOV words, some researchers prepared the testing dataset by selecting only the features seen at least once in the training dataset [12]. This shows that the data preparation is very crucial for our future experiments as well and we need more time to manage the training and testing datasets.

6. Conclusions and Future work

This paper presented a two-stage neural network focusing on both letter and phoneme context information for dealing with two specific difficult problems in L2P conversion. Even though this approach is considered as an expensive and time-consuming approach, it could provide higher phoneme accuracy and word accuracy on both in-vocabulary and OOV words compared to other previous approaches that used the same auto aligned CMU corpora. Furthermore, the new observation model called Phoneme Sequences Pattern-Observation Model has played an important role at the second-stage neural network approach since it allows extending the context information at the phonemic level in order to generate different patterns of phonemes from a duplicated sequence of input letters.

In the future, instead of using a memory consuming method like OBC encoding algorithm, another encoding algorithm will be needed for reducing the number of neurons as well as the size of our L2P model. Furthermore, instead of using a single letter, a syllable-based approach may help to improve the word accuracy as well. For the further research, our approach should include another module (technique or method) after the second-stage neural network in order to reduce some erroneous words which contain just one erroneous phoneme.

7. References

[1] "PASCAL (Pattern Analysis, Statistical Modeling and Computational Learning)," [Online]. Available: http://pascallin.ecs.soton.ac.uk/Challenges/PRONALSYL/Datasets/. [Accessed 10 November 2012].

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

[3] A. Fabio and E. J. Mark, "Phoneme Recognition with Staged Neural Networks," in *IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)*, Como, Italy, July, 2000.

[4] E. B. Bilcu, "Text-To-Phoneme Mapping Using Neural Networks," Tampere, 22 October 2008.

[5] P. Taylor, "Hidden Markov Models for Grapheme to Phoneme Conversion," in *Interspeech*, 2005.

[6] K. U. Ogbureke, C. Peter and B. C. Julie , "Hidden Markov Models with Context-Sensitive Observations for Grapheme-to-Phoneme Conversion," in *Interspeech*, Japan, 2010.

[7] S. Jiampojamarn, K. Grzegorz and T. Sherif, "Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion," in *Proceedings of NAACL HLT*, Rochester, New York, April 2007.

[8] V. Claveau, "Letter-To-Phoneme conversion by inference of rewriting rules," in *Interspeech*, Brighton, UK, September 2009.

[9] 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," in *Proceedings of the NAACL HLT Student Research Workshop and Doctoral Consortium*, Colorado, June 2009.

[10] Y. Marchand and I. D. Robert , "A multi-strategy approach to improving Pronunciation by Analogy," *Computational Linguistics*, pp. 192-199, 2000.

[11] I. Stoianov and J. Nerbonne , "Connectionist Grapheme to Phoneme Conversion_Exploring Distributed Representations," 1999.

[12] L. Patrick, H. Stefan, G. Vlad-Andrei and N. Hermann, "Hidden Conditional Random Fields with M-to-N alignments for Grapheme-to-Phoneme Conversion," in *Interspeech*, Portland, Oregon, 2012.

[13] J. Sittichai and K. Grzegorz, "Letter-Phoneme Alignment: An Exploration," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden, July 2010.

[14] G. Miller, ""Lanauge and Speech"," p. 49, 1981.