A Study on Two-Stage-based Architecture for Grapheme-to-Phoneme Conversion

（書記素－音素変換のための 2 ステージ
アーキテクチャに関する研究）

January, 2016

Doctor of Engineering

KHEANG Seng

キン　セン

Toyohashi University of Technology

Abstract

A modern speech synthesis (Text-to-Speech or TTS) system usually generates output speech through phonological information (or phonetic transcription) 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. In general, the phonemic transcription of a written word could be possibly generated by consulting a pronunciation dictionary available inside the system for the in-vocabulary words or predicted through a data-driven Grapheme-to-Phoneme (G2P) conversion for the unknown or out-of-vocabulary (OOV) words. Besides the TTS system, the G2P conversion has also been widely adopted for other systems such as computer-assisted language learning, automatic speech recognition, spoken term detection, spoken document retrieval and speech-to-speech machine translation systems.

Due to the variability in the pronunciation rules, there is no strict correspondence between graphemes and phonemes, especially in American English language. Thus, many G2P approaches using a many-to-many mapping technique between graphemes and phonemes has been proposed. In order to improve the prediction performance of the G2P conversion model, in this thesis, we propose several approaches based on a two-stage architecture. This architecture allows to treat the problems occurred in the conversion using two different steps: graphemes-to-phonemes and phonemes-to-phonemes.

Our first approach is called *"a two-stage neural network-based G2P conversion"* which is designed for dealing with the problem of conflicting phonemes, where an input grapheme could, in the same context, produce many possible output phonemes at the same time. For example, if a neural network 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 "MATHEMATICIAN". Thus, it is difficult to identify the correct phoneme corresponding to 'A' since there is more than one choice. To solve such a problem, our proposed model first converts the input text/word into multiple phoneme substrings and then uses a combination of the obtained phoneme substrings as a new input pattern to predict the output phoneme corresponding to each input grapheme in a given word.

Since the performance of the neural network-based model for G2P conversion is limited, we use an existing weighted finite-state transducer (WFST)-based method implemented in the Phonetisaurus toolkit to implement our second proposed model. Except the acronyms and words with special pronunciations, we have figured out that most of the error words in G2P conversion are caused by the wrong prediction of their own vowel graphemes. Therefore, we design several *grapheme generation rules*, which enable extra

details (or sensitive information) for the vowel and consonant graphemes appearing within a word. These rules are applied to the input text/words at the first-stage of our proposed model. The evaluation results have shown that a G2P model using different rules can produce different output results that allow each rule to tackle different problems which may occur in different contexts during a conversion. This shows that a single approach does not suffice when addressing all the problems encountered by G2P conversion. Considering this fact, a combination of various approaches using different techniques is a reasonable strategy for treating the problems in a flexible manner.

Combining various techniques can both lend flexibility to the conversion and improve its predictive performance. Therefore, in this thesis, we present a *phoneme transition network (PTN)-based architecture for G2P conversion*. First, it converts a target word into multiple phoneme strings using different existing data-driven methods. Then, it aligns the obtained results–the phoneme-sequence hypotheses–using dynamic programming algorithm, combines them into a confusion network (or PTN), and determines the final output phoneme sequence by selecting the best phonemes from all the PTN bins–blocks of phonemes/transitions between two nodes in the PTN. Moreover, in order to extend the feasibility and improve the performance of the proposed PTN-based model to another higher level, we introduce a novel use of right-to-left (reversed) grapheme-phoneme sequences along with grapheme generation rules. Both techniques are helpful not only for minimizing the number of required methods or source models in the proposed architecture but also for increasing the number of phoneme-sequence hypotheses as well as new phoneme candidates, without increasing the number of methods. Therefore, the techniques serve to minimize the risk from combining accurate and inaccurate methods that can readily decrease the performance of phoneme prediction.

Various model combinations have been conducted and tested. Evaluation results using various word-based pronunciation dictionaries or datasets (such as NETtalk, Brulex, CMUDict and CMUDict_noisy) and K-fold cross-validation techniques show that our proposed PTN-based model, when trained using the reversed grapheme-phoneme sequences, often outperforms conventional left-to-right grapheme-phoneme sequences. In addition, the evaluation also demonstrates that the PTN-based method for G2P conversion is more accurate than all the baseline approaches that are tested in terms of both phoneme and word accuracy.

In the future, we plan to create new and effective grapheme generation rules to further improve our proposed approach, enabling a trained model to generate more accurately output phoneme-sequence hypotheses, such that only two models (using conventional and reversed grapheme-phoneme sequences) will be sufficient for our PTN-based G2P conversion. Moreover, the hamming-distance, calculated from the articulatory features of phonemes, shall be used for the dynamic programing algorithm in the PTN generating process. A PTN sequence shall be used instead of a single phoneme sequence to represent an OOV keyword in the spoken term detection as well as other systems.

Contents

List of Figures

List of Tables

Abbreviations

ANN	**A**rtificial **N**eural **N**etwork
AROW	**A**daptive **R**egularization **O**f **W**eight **V**ectors
ASR	**A**utomatic **S**peech **R**ecognition
CALL	**C**omputer-**A**ssisted **L**anguage **L**earning
CMU	**C**arnegie **M**ellon **U**niversity
CN	**C**onfusion **N**etwork
CRF	**C**onditional **R**andom **F**ields
CTC	**C**onnectionist **T**emporal **C**lassification
DP	**D**ynamic **P**rograming
EM	**E**xpectation **M**aximization
FANN	**F**ast **A**rtificial **N**eural **N**etwork
FST	**F**inite **S**tate **T**ransducers
G2P	**G**rapheme-to-**P**honeme
GGR	**G**rapheme **G**eneration **R**ule
HCRF	**H**idden **C**onditional **R**andom **F**ields
HMM	**H**idden **M**arkov **M**odel
IV	**I**n **V**ocabulary
LPC	**L**inear **P**rediction **C**oding
LSTM	**L**ong **S**hort-**T**erm **M**emory
LVCSR	**L**arge-**V**ocabulary **C**ontinuous **S**peech **R**ecognition
ME	**M**aximum **E**ntropy
MIRA	**M**argin **I**nfused **R**elaxed **A**lgorithm
MLN	**M**ulti**L**ayer **N**eural network
MT	**M**achine **T**ranslation
NAROW	**N**arrow **A**daptive **R**egularization **O**f **W**eight **V**ectors

OBC	**O**rthogonal **B**inary **C**odes
OOV	**O**ut **O**f **V**ocabulary
PAcc	**P**honeme **Acc**uracy
PbA	**P**ronunciation-**b**y-**A**nalogy
PER	**P**honeme **E**rror **R**ate
PSOLA	**P**itch **S**ynchronous **O**ver**l**ap-**a**dd
PTN	**P**honeme **T**ransition **N**etwork
RELP	**R**esidual-**E**xcited **L**inear **P**rediction
RNN	**R**ecurrent **N**eural **N**etwork
RNNLM	**R**ecurrent **N**eural **N**etwork **L**anguage **M**odel
ROVER	**R**ecognizer **O**utput **V**oting **E**rror **R**eduction
S2ST	**S**peech-**t**o-**S**peech **T**ranslation
SSMCW	**S**tructured **S**oft-**M**argin **C**onfidence **W**eighted learning
STD	**S**poken **T**erm **D**etection
SVM	**S**upport **V**ector **M**achine
TSNN	**T**wo-**S**tage **N**eural **N**etwork
TTS	**T**ext-**t**o-**S**peech
U-STAR	**U**niversal **S**peech **T**ranslation **A**dvanced **R**esearch
WAcc	**W**ord **Acc**uracy
WER	**W**ord **E**rror **R**ate
WFST	**W**eighted **F**inite-**S**tate **T**ransducer

Chapter 1

Introduction

Contents

Human communication is a fundamentally cooperative enterprise, operating most naturally and smoothly within the context of (1) mutually assumed common conceptual ground, and (2) mutually assumed cooperative communicative motives [1]. A hypothesis is that the first uniquely human forms of communication were pointing and pantomiming. The social-cognitive and social-motivational infrastructure that enabled new forms of communication then acted as a kind of psychological platform on which the various systems of conventional linguistic communication could be built. According to the Wikipedia page (https://en.wikipedia.org/wiki/History_of_communication), human communication was revolutionized with speech approximately 500,000 years ago; symbols were developed about 30,000 years ago [2], and writing about 5,000 years ago.

During the old era and before the birth of computer, human had to read the written text to be understood or to transfer its meaning to another people through speech. The revolution of computers, electronics, media and technologies has completely changed the human life, especially the way of human thinking and communicating. Many impossible things and creatures have been found, analyzed, realized, invented and then innovated by

human. For many years, scientists have dreamed of building machines able to converse with their creators, by endowing them with a measure of "intelligence" or "understanding" together with speech recognition and synthesis capabilities [3]. Even if building a machine that would have human-like intellectual ability in both understanding and talking is impossible, applications generating artificial speech (i.e., speech synthesis) or understanding human speech (i.e., speech recognition) are highly demanded on the market, especially now when the quality of both the synthesized speech and the recognized information is much better than a decade ago. This is because speech technologies have as their principle objective to facilitate the interaction between humans and computers.

In order to make the speech technologies possible, the knowledge of language reading and understanding is definitely required. Learning new languages is difficult and very time-consumming. For example, it is required a lot of time and efforts to be capable to read a text written in a specific language; moreover, the reading and speaking experiences are definitely unavoidable to understand how to pronounce the text or word correctly. For a human being, the reading performance is acceptable as long as it can be understood by other people and especially by the native speakers. Unlikely, this is not enough for the computer applications including the automatic speech recognition and comuputer-assisted language learning softwares, where a perfect quality of text reading is theoretically and highly demanded. Therefore, when the quality of pronuciation is concerned, the phonological knowledge in reading must be taken into account too.

The remainder of this chapter is organized as follows. It first overviews a world of languages and then describes what the texts, phonetics, prosody and speech in Text-to-Speech system in Section 1.1. Next, it presents the objectives of our research in Section 1.2. The advantages of grapheme-to-phoneme conversion and its applications are briefly explained in Section 1.3. Then, it presents our proposed approaches for grapheme-to-phoneme conversion as our contributions in Section 1.4. An organization of the thesis is lastly written in Section 1.5.

1.1 Languages, texts, phonetics, prosody and speech

Now, there are 7.2 billion people on earth. According to the infographic of a world of langauges depicted in Fig.1.1, a survey of 6.3 billion people shows that there are at least 7,102 known languages alive in the world today. Twenty-three of these languages are a mother tongue for more than 50 million people and make up the native tongue of 4.1 billion people around the world. These languages includes: Chinese, Spanish, English,

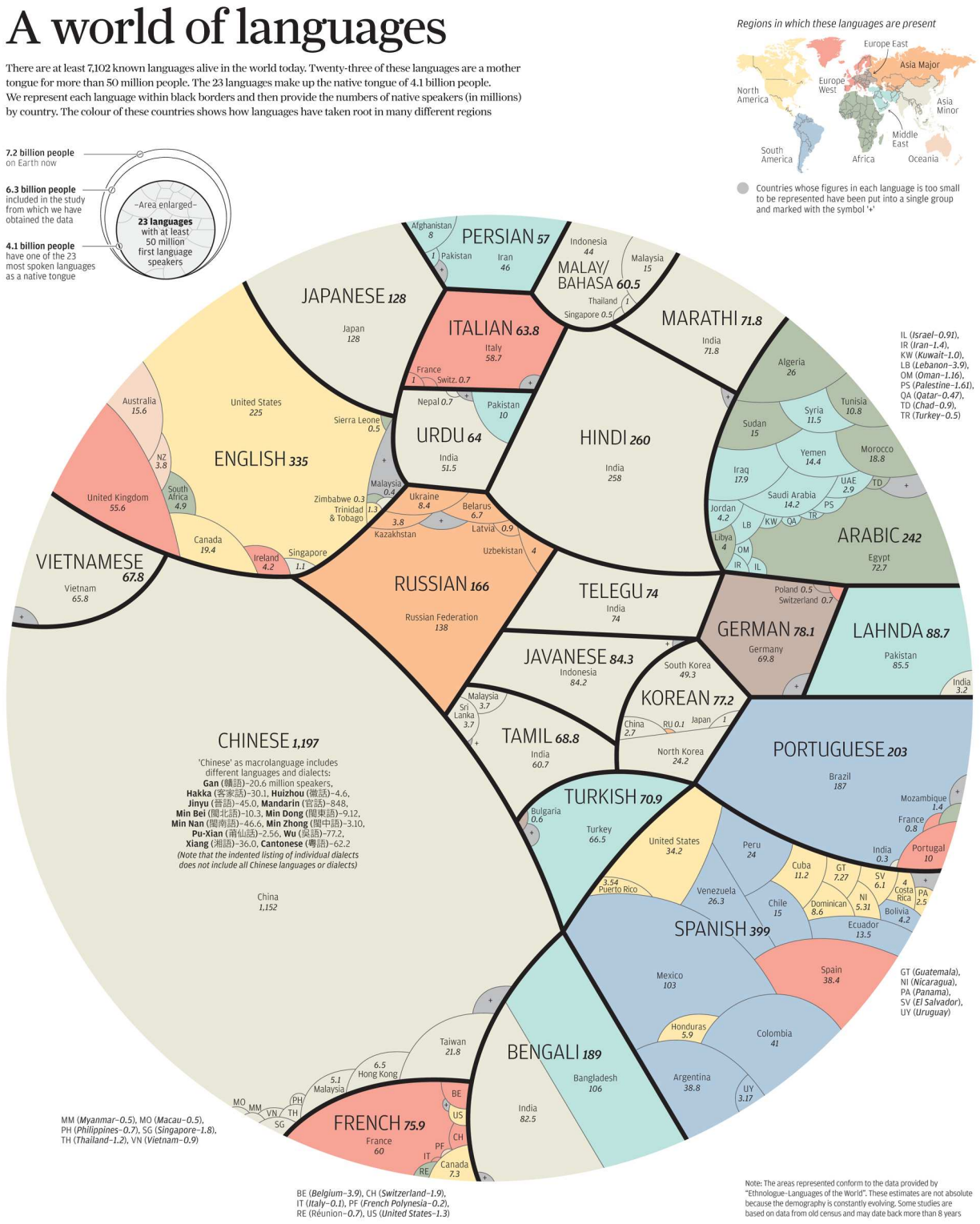


FIGURE 1.1: Infographic: a world of languages, created by the fine linguists at the South China Morning Post (http://www.scmp.com/infographics/article/1810040/infographic-world-languages?comment-sort=recommended)

FIGURE 1.2: Infographic: Number of countries in which each language is spoken

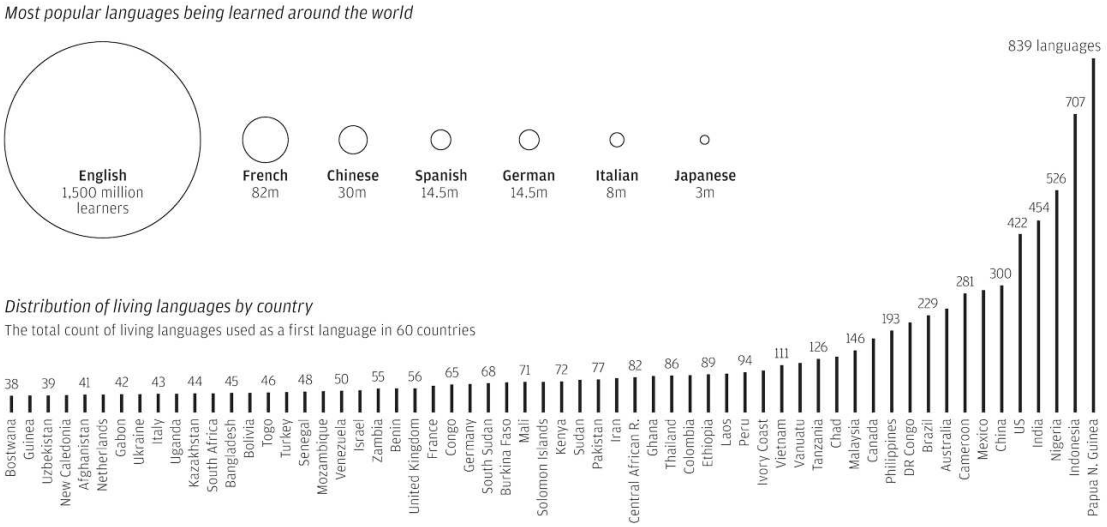


FIGURE 1.3: Infographic: Most popular languages being learned around the world

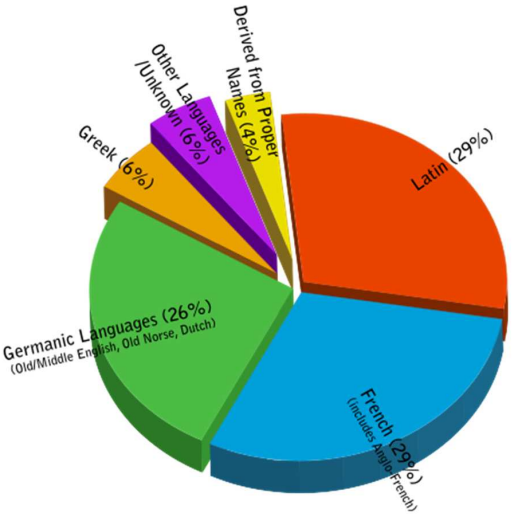


FIGURE 1.4: The most common foreign language influences in English
https://en.wikipedia.org/wiki/Foreign_language_influences_in_English

Hindi, Arabic, Portuguese, Bengali, Russian, Japanese, Lahnda, Javanese, German, Korean, French, Telegu, Marathi, Turkish, Tamil, Vietnamese, Urdu, Italian, Malay/Bahasa and Persian. In Fig.1.1, each language is represented within black borders, and the number of native speakers (in millions) by country is also provided; the colour of these countries shows how languages have taken root in many different regions. The bottom part of Fig.1.1 demonstrates that English is the most popular language being learned and spoken by many countries around the world, even though Chinese has the largest number of native speakers.

Regarding the fact that each language has been used by many people living in different countries for such a long time, the pronunciation rules in some languages have been regularly increased and become unstable day-by-day due to the social impacts. For example, according to Fig.1.4, the pronunciation rules in the modern English language have been gradually changed and become the most complicated among other languages in the world because it has been influenced by many foreign languages, such as, French, Latin, Greek, Germanic, etc.

In general, the orthographic text or word in any language could be perfectly pronounced only by the linguistic experts who have a deep phonological knowledge in that language; the complexity of pronunciation rules is highly language-dependent. To do this, the experts have to learn and memorize all the possible pronunciation rules as well as its using contexts (e.g., the context of speaking). It is natural for human beings that they have a special ability to memorize those rules and then use them to speak the words out through the human speech production and control mechanisms which are very complex to be understood, but it might sound like a joke to make a blind machine have such ability. However, for over the years, the scientists have turned that joke into reality. They created the first computer-based speech synthesis system in the late 1950s, and the first sophisticated Text-To-Speech (TTS) system in 1968 [4]. Since then, the TTS system has attracted many research interests and been chosen as one of the most interesting topics in the fields of speech processing.

A TTS system is a computer-based speech system that is capable of transforming the input textual information into intelligible speech signal (for example, the artificial human spoken voice). As depicted in Fig.1.5, it consists of two fundamental components. The first component (or *front-end*) is reponsible for three main tasks: *text normalization, phonetic transcription* and *prosody generation*, which converts texts to linguistic

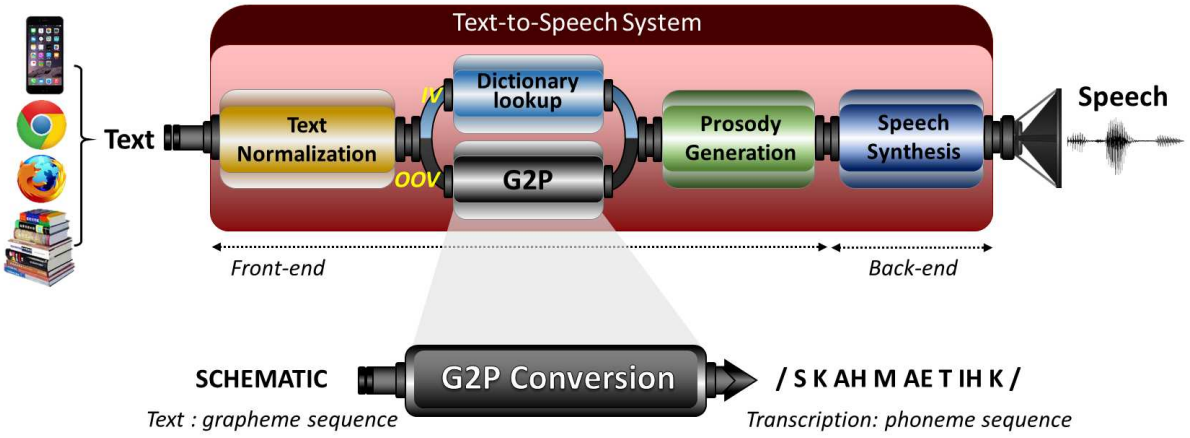


FIGURE 1.5: Architecture of a TTS system. In this figure, the phoneme symbols are based on the CMU phone sets.

specification–a sequence of phonemes[1] annotated with contextual information. The second component (or *back-end*) is known as a *waveform generator* or *speech synthesizer* that uses the obtained linguistic specification to generate a speech waveform as output.

Text normalization system is mainly implemented to convert the raw input text into an appropriate orthographic form that will be used in the phonetic transcription process. It consists of sentence segmentation, tokenizing and normalization of non-standard words [5]. The main problem in the sentence segmentation is the ambiguity of the period and the ambiguity of marking sentence boundaries or abbreviations. Many approaches have been proposed for the period disambiguation, such as the rule-based systems for heuristic period disambiguation operated on local grammars containing abstract contexts for within-sentence periods and sentence boundaries [6], the Mikheev's rule-based segmentation [7], the decision tree classifier in Riley that use context features (including word lengths, capitalization, and word occurrence probabilities on both sides of the period in question)[8], etc. The tokenizing task is simply responsible for spliting the text at white spaces and at punctuation marks that do not belong to abbreviations identified in the preceding process. The normalization of non-standard words is usually a very complex task not only in the TTS system but also other systems, and it also includes several language-dependent problems [9]. This subtask includes number conversion (e.g., sorting number, ranking number, counting number, phone number, ID, etc.), homograph

[1] A phoneme is the smallest unit of sound in speech. Phonemes are used in the spoken language while letters are used in the written language. For example, the word "cat" consists of three phonemes making the sounds /k/ (as in "can"), /a/ (as in "pad"), and /t/ (as in "tusk"). The number of phoneme symbols is language-dependent and differently represented from one standard to another. According to [3], there are only 26 letters in the English alphabet, but around 40 phonemes when using the phone set of the CMU pronunciation dictionary. That's because some letters and letter groups can be read in multiple ways ('a', for example, can be read differently, as in "pad" or "paid"), so instead of one phoneme per letter, there are phonemes for all the different letter sounds. Some languages need more or fewer phonemes than others (typically 20-60).

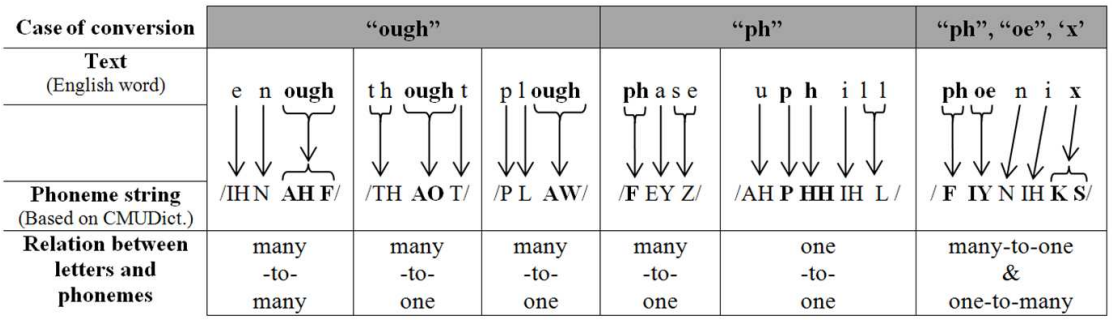


FIGURE 1.6: Different types of relations between letters and phonemes in the English CMU pronouncing dictionary.

disambiguation, appropriate treatment of acronyms (because some have to be spelled, others not), and expansion of abbreviations, emails, URL addresses, dates in different formats, special symbols desinating monetary units, and etc.

According to Fig.1.5, an automatic text-to-phonetic transciption or text-to-phoneme conversion system is responsible for converting the token words of a normalized text–a text under an appropriate orthographic form mentioned in the previous paragraph–into their correponding phonetic or phoneme forms. For example, based on an English word-based pronunciation dicitionary (known as CMUDict), the word "SCHEMATIC" is converted into its correponding phoneme sequence /S K AH M AE T IH K/. This system is very time-consuming to be implemented and also highly language-dependent. For a language with stable pronunciation rules like Spanish, this system is simply implemented using a traditional rules-based approach, such as the context-dependent rewriting rules [10, 11], for example. The problem concerning the automatic pronunciation generation from text has been fairly studied in [9, 12]. However, it is insufficient to use such traditonal approach to deal with a language with deep orthography–in other words, with no obvious letter-to-phoneme correspondence [13, 14]–like English. For instance, Fig.1.6 shows that there has no standand correpondence between the number of letters and phonemes in English.

Since the desired speech output is usually synthesized through phonemic information rather than the direct representation of textual information, the quality of the generated phonemic information has a strong impact on the performance of the whole TTS system, in terms of the degree of understanding and naturalness [15]. Therefore, as shown in Fig.1.5, the phonemic transcription of a written word could possibly be generated by consulting a pronuciation dictionary available inside the system for the in-vocabulary (IV) words or predicted through a data-driven Grapheme-to-Phoneme[2] (G2P) conversion for the unknown or out-of-vocabulary (OOV) words–the words that do not exist in

[2]A grapheme is a single letter or multiple connecting letters that represent the sounds in our speech.

the dictionary. As a result, many data-driven approaches for G2P conversion, such as hidden Markov models (HMMs) [16, 17], support vector machines (SVMs) [18], artificial neural network (ANNs) [15, 19, 20], joint-sequences [21, 22, 23, 24, 25], a weighted finite-state transducer (WFST) [26, 27], conditional random fields (CRF) [28, 29, 30], hidden conditional random fields (HCRF)[31, 32], an adaptive regularization of weight vectors (AROW) [33], a narrow adaptive regularization of weight vectors (NAROW) [34], and structured soft-margin confidence weighted learning (SSMCW) [35], etc., have been proposed. Likewise, the main interest of our research and thesis also focus on the problems concerning G2P conversion.

After the token words have been converted to phonemes, the prosody[3] generation module is used to assign the correct picth accent, lexical stress, rhythm, intonation, duration and other related attributes to the phoneme form obtained from the text-to-phoneme (or G2P) conversion module. Long time ago, when the prosody generation for speech synthesis was concerned, the traditional hand-crafted rules was usually used [36]. The poor prosody is a significant factor in limiting speech quality [37], so the researchers have proposed various data-driven based approaches [38, 39, 40, 41] for improving the performance of the automatic prosodic labeling system over the last decade. Recently, the HMM-based approaches for prosody generation have become the most successful technique in the fields of speech synthesis [42, 43].

Finally, the speech synthesis system produces the waveform signal as the final output of the TTS system. It consists of two primary technologies generating synthetic speech waveforms includes: concatenative synthesis and formant synthesis. Each technology has strengths and weaknesses, and the intended uses of a synthesis system will typically determine which approach is used. The scientists have proved that the formant synthesis is better than the concatenative synthesis because it aims to provide good speech quality and intelligibility while reducing storage requirements [3]. For over the years, linear prediction coding (LPC) has been the most popular technique [44, 45, 46]. More recently, because the pitch synchronous overlap-add (PSOLA) synthsis [47, 48] uses pitch (fundamental frequency or F0) that can be easily modified during synthesis, it has become more popular as a way of generating speech output. PSOLA is independent of any particular coding strategy and gives best output when no data reduction is used at all. After Machhi et. al. [49] have studied the effect of different coding methods on the intelligibility of the speech output, they found that residual-excited linear prediction

[3]According to https://en.wikipedia.org/wiki/Prosody_(linguistics), in liguistic, prosody is concerned with those elements of speech that are not individual vowels and consonants but are properties of syllables and larger units of speech. These contribute to such linguistic functions as intonation, tone, stress, and rhythm. Prosody may reflect various features of the speaker or the utterance: the emotional state of the speaker; the form of the utterance (statement, question, or command); the presence of irony or sarcasm; emphasis, contrast, and focus; or other elements of language that may not be encoded by grammar or by choice of vocabulary.

(RELP) provided higher intelligibility than PSOLA for voiced consonants, which were assumed to be more sensitive to coding methods and pitch changes than vowels. This is somewhat against the usual claim that PSOLA gives higher quality than LPC.

1.2 Objectives of the research

A speech synthesis system usually creates output speech using the phonological information rather than textual information. Thus, the quality of the precise conversion of arbitrary text into its corresponding phoneme string has a strong impact on the performance of generated speech output. According to Fig.1.5, the modern TTS systems use the dictionary look-up as the primary method to derive the phoneme transcription of the words. This method does not have any knowledge or ability to derive the pronunciation of any unknown or OOV words. Due to the influence of foreign words and other social impacts, such systems have to deal with new words that have been gradually invented by people from different countries around the world, so the secondary method is needed for a backup strategy. Since the automatic data acquisition methods are the most economic ones in terms of time and effort, the application of machine learning methods to the G2P conversion is the most appropriate choice.

Therefore, the main objectives of our research in this thesis are (1) to analyze the problems occurred in G2P conversion, (2) to understand and compare the state-of-the-art methods to the G2P conversion, and then (3) to find an efficient way to improve the pronunciation of OOV words, in other words, to propose new methods that can improve the performance of phoneme prediction of the G2P conversion model.

In our scope of research, there are some limitation as follows. Only the problems in G2P conversion are counted, the others, including the problems concerning the text normalization, prosody generation, and speech synthesis systems, are not included. Problems concerning the liaison words are not included in this study because the evaluation data or corpus is purely word-based pronunciation dictionary.

1.3 Advantages of G2P conversion

Besides the TTS system, the automatic transcription of unseen words into their corresponding phoneme strings (also called a data-driven G2P conversion) has been widely adopted for other speech systems, as illustrated in Fig. 1.7 and 1.8, which includes automatic speech recognition, computer-assisted language learning, spoken term detection, spoken document retrieval, speech-to-speech translation, etc.

In an automatic speech recognition (ASR) system, both acoustic and language models are two principle components needed to be trained before the decoding process. The acoustic model is trained using a database of recoded speech signals together with their transcript files, while the language model is trained using a word-based pronunciation dictionary (or word lexicon) and the transcript files of the training speech data. The transcription of each speech signal consists of a sequence of words and phonemes, followed by a tag which can be used to associate this word/phoneme sequence with the corresponding speech signal. In order to create an accurate ASR system, a large speech training database that contains many sequences of words spoken by different people and a rich word-based pronunciation dictionary in a specific language are needed. For a system in which a large-vocabulary continuous speech recognition (LVCSR) is concerned, a word lexicon must be reconstructed regularly to accommodate unseen words by using a data-driven approach. A method for dealing with this problem is G2P conversion of such unseen words, which can be used for an expanded lexicon [50]. In addition, the G2P conversion system is also sometimes used to derive the pronunciation of word sequence belonged to some transcript files in which the phonemic information is not available.

Computer-assisted language learning (CALL) is succinctly defined in a seminal work by Levy [51] as *"the search for and study of applications of the computer in language teaching and learning"*. According to the webpage `https://www.llas.ac.uk/resources/gpg/61`, CALL system is often perceived, somewhat narrowly, as an approach to language teaching and learning in which the computer is used as an aid to the presentation, reinforcement and assessment of material to be learned, usually including a substantial interactive element. The simple use of G2P conversion in such system is just for deriving the phonetic transcription of OOV words if needed.

Furthermore, spoken term detection (STD) is the problem of determining whether and where a target word or multi-word phrase has been uttered in a speech recording [52]. Many STD systems are based on large-vocabulary ASR systems, trained on very large amounts of data [53]. In such systems, the STD task becomes the problem of searching a speech recording that has already been recognized and indexed using a large ASR system. Since a word-based searching method is limited to the improvements of search results, a phoneme-based searching method has been practically implemented, in STD system. In this case, the phoneme sequence or pronunciation of keywords corresponding to a search query is used as search input. The pronunciaiton of a keyword can be extracted directly from the word lexicon if found; otherwise, a trained G2P conversion model is usually used to estimate the pronunciations of the remaining OOV keywords [54]. Moreover, the effect of pronunciations on OOV search queries in such system have also been studied in [55, 56, 57].

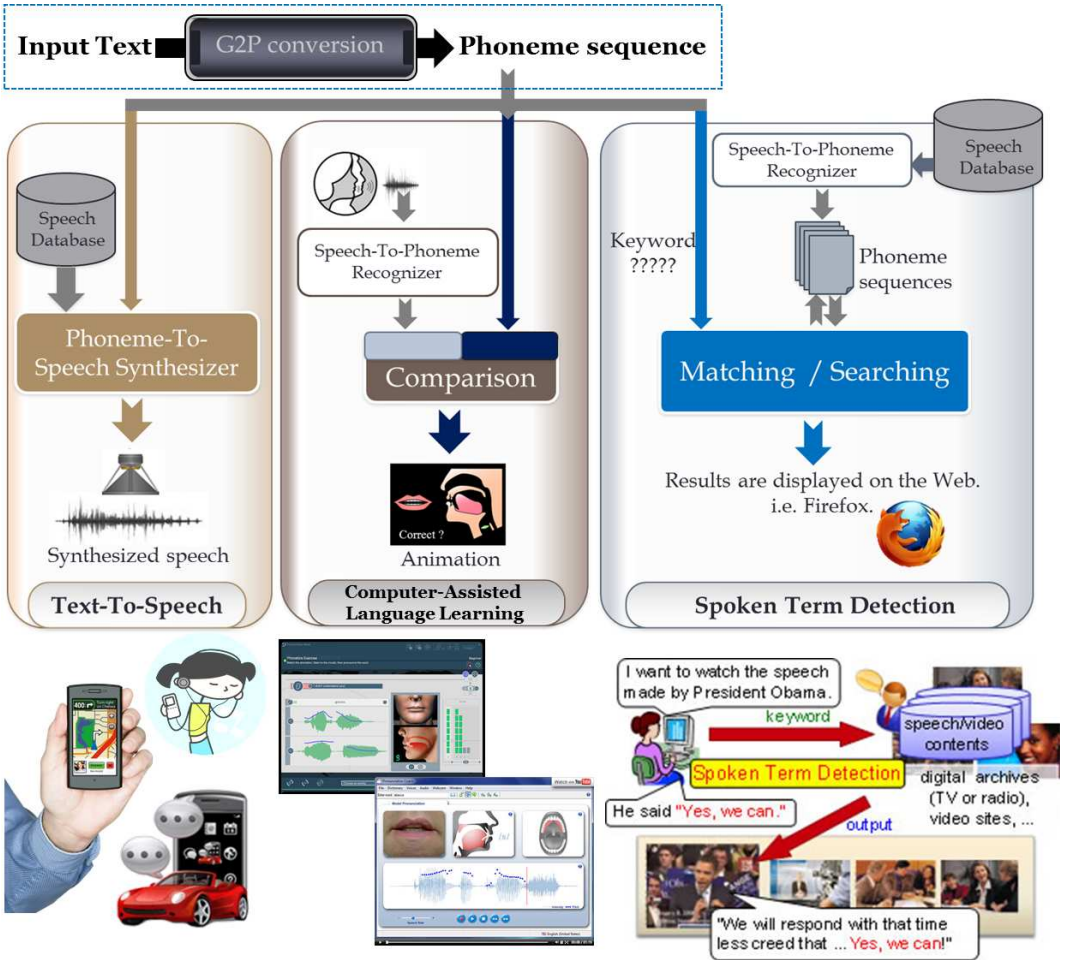


FIGURE 1.7: Various applications using G2P conversion and theirs examples

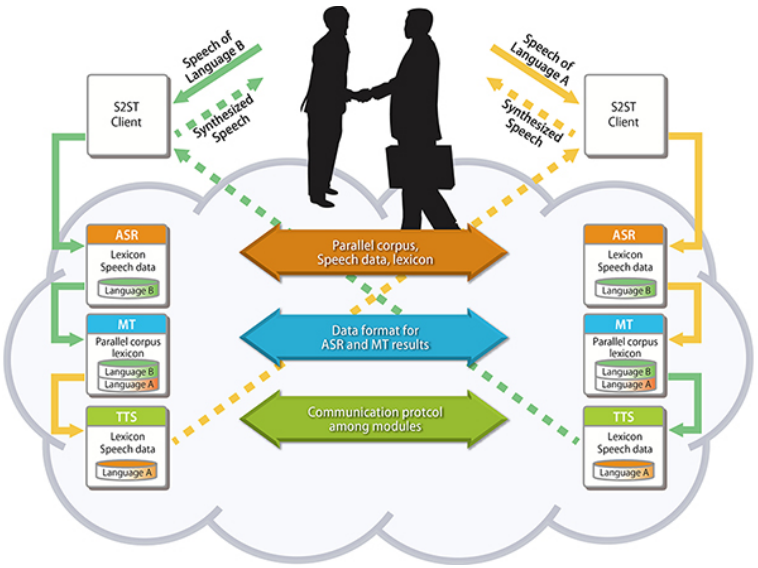


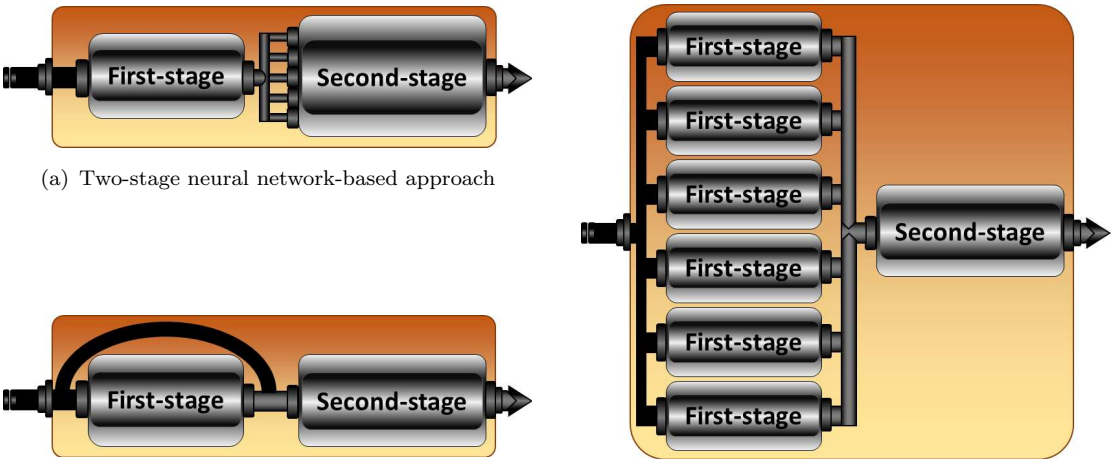
FIGURE 1.8: Architecture of a S2ST system in which a G2P conversion is adopted.
http://www.ustar-consortium.com/standardization.html

On the other hand, modern future speech-to-speech translation (S2ST) technology is an effective mean to break through language barriers between people who do not speak the same language [4]. The goal of this system is to enable real-time, interpersonal communication via natural spoken language for people who do not share a common language; in other words, it aims at translating a speech signal in a source language into another speech signal in a target language. Basically, a S2ST system is composed of three modules: speech recognition from the source language, machine translation (MT) from the source text into the target text, and speech synthesis to the target language. Therefore, G2P conversion plays a quite important role in such a giant system because it will be implemented within all the three modules for dealing with the unknown speech or vocabularies. According to Fig.1.8, in order to establish S2ST systems, the ASR, MT, and TTS systems for both source and target languages must be built by collecting speech and language data such as: audio data, speech transcriptions, pronunciation lexica for each and every word, parallel corpora for translation, and so on. In order to connect these modules for different languages and functions reliably, it is necessary to standardize the communication protocols and data formats between modules, as illustrated in the figure. For instance, the Universal Speech Translation Advanced Research (U-STAR) Consortium was established as an international research collaboration entity to develop a universal network-based speech-to-speech translation system.

1.4 Contributions

In this thesis, our research only focuses on G2P conversion and its improvements. We propose three different approaches based on machine learning techniques for tackling different problems occurred in G2P conversion. These approaches are illustrated in Fig.1.9 and implemented as two-stage architecture-based models in which the input graphemic information is first converted to its preliminary phonemic information, and then all the preliminary information are used as input hypothesis to determine the best final output phonemic information (i.e., Graphemes ⇒ Phonemes ⇒ Phonemes).

We first propose a two-stage neural network-based approach [19, 58] to improve the performance of state-of-the-art single-stage neural network-based approach [15, 59] for G2P conversion. This approach, as depicted in Fig.1.9(a), aims to deal with the problem of conflicting phonemes, where an input grapheme can, in the same context, produce many possible output phonemes at the same time. The two-stage neural network-based G2P conversion model is fundamentally built by putting two different multi-layer neural networks in sequence; based on the context-dependent technique, the first neural network is implemented as a many-to-many conversion model to automatically transform



(a) Two-stage neural network-based approach

(b) Two-stage model using grapheme generation rules

(c) Phoneme transition network-based approach

FIGURE 1.9: Three proposed two-stage-based architectures for G2P conversion.

each grapheme sequence extracted from a given word into the corresponding phoneme sequence; and the second one 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.

Later, because of the limitation of neural network in the improvement of G2P conversion model, we propose another two-stage architecture-based approach [60] using an existing WFST-based G2P conversion framework implemented in the Phonetisaurus toolkit[4]. The differences between this and our firstly proposed approach includes: (1) the use of our newly proposed grapheme generation rules that enables extra details for the vowel and consonant graphemes appearing within a given word, (2) the replacement of neural networks by WFST-based methods, and (3) a new strategy that combines both graphemic and its preliminary phonemic information to be used as hypothesis of the second-stage model. The schema of this approach is illustrated in Fig.1.9(b).

Lastly, we figure out that each existing approach has been designed using specific techniques that address particular challenges faced by G2P conversion. Hence, any single approach will not suffice when addressing all of the problems encountered by G2P conversion. Due to this fact, we propose a phoneme transition network (PTN)-based architecture shown in Fig.1.9(c) in which various approaches/methods are combined to meet the challenges of G2P conversion. Combining various methods can both lend flexibility to the conversion and improve its predictive performance. The proposed method first builds a confusion network using multiple phoneme-sequence hypotheses generated from several G2P source methods, and then determines the best final-output phoneme from

[4] https://code.google.com/p/phonetisaurus/

each block of phonemes in the generated network. Moreover, in order to extend the feasibility and improve the performance of the proposed PTN-based model, we introduce a novel use of right-to-left (reversed) grapheme-phoneme sequences along with grapheme-generation rules. Both techniques are helpful not only for minimizing the number of required methods or source models in the proposed architecture, but also for increasing the number of phoneme-sequence hypotheses as well as new phoneme candidates, without increasing the number of source methods. Therefore, the techniques serve to minimize the risk from combining accurate and inaccurate methods that can readily decrease the performance of phoneme prediction.

1.5 Organization of the thesis

The remainder of this thesis is organized as follows.

First, in Chapter 2, we describe the previous work related to G2P conversion, including G2P alignment methods, traditional solution to G2P conversion, and data-driven solutions to G2P conversion. Moreover, the results of some previous G2P methods evaluated using different corpora are also summarized at the end of this chapter.

In Chapter 3, we explain the lack of ability in phoneme prediction of the state-of-the-art single-stage neural network-based G2P conversion, and then introduce our firstly proposed approach "a two-stage neural network-based G2P conversion" to solve the problem of conflicting phonemes mentioned in the previous section (Section 1.4). The data preparation, evaluation results, and discussions, of this approach are also included.

Then, in Chapter 4, we present the novel two-stage architecture-based approach using WFST-based G2P conversion framework available in the Phonetisaurus toolkit instead of neural networks. In addition, we also introduce a number of grapheme generation rules that enable extra sensitive information for the vowel and consonant graphemes appearing in a given word.

Last but not least, in Chapter 5, we explain the reasons of using multiple-approaches combination to deal with the problems encountered by G2P conversion in a flexible manner. Next, we introduce our accurate PTN-based G2P conversion and our novel use of reversed grapheme-phoneme sequences along with grapheme-generation rules. The evaluation results of both baseline and proposed approaches are nicely written in this chapter, followed by such a good discussion.

Finally, we conclude our thesis in Chapter 6 and also suggest some ideas for the further improvements of G2P conversion and its applications.

Chapter 2

Related Work

Contents

A quality of the precise conversion of arbitrary text into its corresponding phoneme string has a strong impact on the performance of the whole TTS system. Theoretically, the phonemic transcription of each input word is usually assigned by looking-up the built-in pronunciation dictionaries of the system. However, the dictionaries cannot cover the continuously expanding language, especially the language with deep orthography like English, for example. Therefore, an alternative G2P system is necessary to predict the phoneme string corresponding to the unknown or OOV words.

This chapter provides an overview of the state-of-the-art approaches for G2P conversion. We first describe the traditional solutions to automatic phonemization in Section 2.1. Then, the G2P alignments and data-driven machine learning-based solutions for G2P conversion are briefly reviewed in Section 2.2.

2.1 Traditional solutions

In a traditional solution, the challenge of automatic phonemization of words is usually approached by rewrite rules. These rules-based approaches are often used in TTS system as an alternative for dictionary look-up, since they were extensively studied long before computers had gain a center place in the development of the mankind [61]. According to the work of Chomsky and Halle in 1968 [10], the rules are context-dependent and carefully designed by expert linguists, which are very expensive in terms of time-consuming and complexity. These rules are usually represented in the following form:

$$A\ [B]\ C \rightarrow D \tag{2.1}$$

where B represents the target letter substring to be converted, D is the phoneme substring corresponding to B. A and C are the surrounding left- and right-context respectively of B. The B substring is variable in length, which can be appeared as a single letter, one or more graphemes (each corresponding to a single phoneme substring D), a completed word, etc. Furthermore, rules may involve different linguistic characteristics such as: syllable boundaries, part-of-speech tags, stress patterns or etymological origin of a word. For example, inspired by the work of Chomsky and Halle, the automatic rules-based system of Elovitz et. al. [62] created in 1976 contains 329 phonological rules; other typical rule sets are also described in [63, 64].

In order to derive the pronunciation for the input word, the rules designed by experts are applied in the order that they appear in the rule list–usually from the most specific rule to the least specific one. Whenever several rules exist for the same target letter in different contexts, the rule that appears at the top of the list is applied in the first place because of having higher priority than the rule that appears at the bottom or lower part of the list. Theoretically, the words are usually scanned form left to right direction and the rule triggers are linearly searched. Every time a rule match is found, an output phoneme is assigned and then the search window is shifted to the right N characters; in this context, N is the number of characters that were necessary to trigger the rule. If no match is found, the size of the sliding windows is then decremented and the rules are scanned again until a match triggers the rule. The default rules with lowest priority are based on single characters, therefore a match is always found in any case. The larger character clusters are given priority when scanning, therefore, every time the window is shifted after having emitted a phoneme, its size is reset to the maximum value. In a language with deep orthography like English, consonant clusters are usually converted first, as for vowels, the letter-to-phoneme correspondences are rather ambiguous and

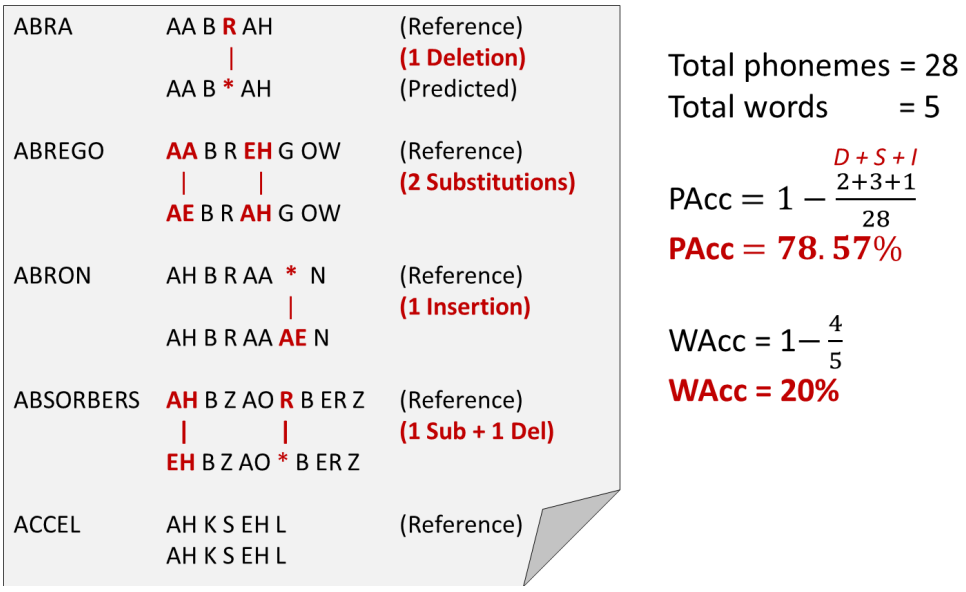


FIGURE 2.1: Example of the PAcc and WAcc calculations.

they account for the main part of the errors. Previously converted consonants can be then used as a part of the contextual information for converting vowels.

In general, the performance of G2P systems are reported in phoneme and word accuracy. The phoneme accuracy (written as PAcc) is calculated using either the Hamming distance or Levenshtein distance between gold-standard outputs–the reference phoneme-sequences–and the predicted sequences to find the number of correct phonemes. The word accuracy (written as WAcc) is calculated by counting the number of fully correct phoneme sequences given testing words. Mathematically, both phoneme and word accuracies can be calculated as follows:

$$PAcc = 1 - PER = 1 - \frac{S_p - D_p - I_p}{N_p} \tag{2.2}$$

$$WAcc = 1 - WER = 1 - \frac{S_w}{N_w} \tag{2.3}$$

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 are usually used in the 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). For instance, after deriving the phoneme sequences corresponding to five input words (e.g., "ABRA", "ABREGO", "ABRON", "ABSORBERS" and "ACCEL"), five different phoneme sequences (e.g., /AA B AH/, /AE B R AH G OW/, /AH B R AA AE N/, /EH B Z AO B ER Z/ and /AH K S EH L/) are respectively generated. Here, the

phoneme symbols are based on the CMU phoneme-set. The evaluation results in Fig.2.1 shows that the values of PAcc and WAcc are equal to 78.57% and 20%, respectively. In G2P conversion, the value of WAcc is more important than PAcc.

According to Damper et al. in 1998 [65], they evaluated the rules proposed by Elovitz et al. in 1976 [62] on a Teacher's Word Book (TWB) dictionary of 16,280 words [66]. As a result, the word accuracy as low as only 25.7% was achieved. This result is very different from the 80-90% word accuracy reported by Elovitz et al. [62], which can be explained by the fact that Damper et al. used a stricter evaluating technique that did not classify pronunciations not containing any severe errors as "good" pronunciations. Also, this later evaluation was performed on TWB dictionary that uses a phone-set of 52 phonemes, while the rewrite rules include only 41 phoneme symbols. Such a discrepancy in phoneme inventories may be one of the main causes of errors [4].

Rule-based systems require hiring an expert linguist and therefore have a high production and maintenance cost, they clearly lack in flexibility and are highly language-dependent. Moreover, they do not take into consideration any kind of statistical measures such as rule probability, frequency counts, etc., that could be helpful in order to improve robustness [3]. In the last two decades data-driven approaches have been widely used to solve the problem of automatic phonemization. They are flexible and mostly language-independent, which makes them a perfect alternative to rule-based approaches.

2.2 Data-driven solutions

Over the years, data-driven based machine learning solutions for G2P conversion have been widely developed to challenge the phonemization of the OOV words, and widely adopted in many modern speech systems as explained in Section 1.3.

Inspired by Chomsky and Halle [10], the context-dependent based technique still plays an important role in both the classification-based approaches [67, 68, 69] and the generative models [22, 70, 71, 72] for G2P conversion. There are two components in both classification and generative systems that allow training from pairs of word-phonemes sample. The first component is to discover hidden relations between graphemes and phonemes, called "alignments", which allows the G2P system to learn what phoneme to generate for each input grapheme and its context. The second component is a learning mechanism to train a model to generate output phonemes given words. There are two paradigms for training the aligned grapheme-phoneme data. G2P conversion can be viewed either as *a multi-class classification problem*, where each sub-phoneme output is drown directly

from the focused grapheme and its context (surrounding graphemes) without considering the phoneme sequence output, or as *a sequence prediction problem*, which takes into account the grapheme sequence input and phoneme sequence output [73]. In the classification-based approaches, each phoneme is predicted independently using a classifier such as a neural network [69], instance-based learning [67], and decision tree [68]. On the other hand, the sequence-based approaches are different from the classification-based approaches because they take previous phoneme decisions into consideration for the current phoneme decision. These approaches includes HMMs [16, 70], joint N-grams models [21, 22, 71], pronuciation by analogy (PbA) [72, 74], constraint satisfaction inference (CSInf) [75], WFST [26, 27], and so on. In addition, the G2P learning is also closely related to *structured learning techniques* including HMMs [76], averaged perception algorithm [77], SVMs for structured outputs [78], CRFs [29, 31, 79] HCRFs[32], and a family of adaptive regularization of weight vectors (AROW) [33, 34, 35, 80].

The remainders of this section are organized as follows. In Section 2.2.1, a brief overview of G2P alignments is described. From Section 2.2.2 until the end of the main Section 2.2 is focused on the data-driven approaches for G2P conversion. In addition, the summary of the G2P results for various datasets found in the literature are listed in Tables 2.4 and 2.5 at the end of this chapter.

2.2.1 Alignments in G2P conversion

In G2P conversion, the training data are generally available in the form of word-phoneme pairs without any explicit information indicating individual grapheme-to-phoneme relationships (as seen in Table 2.1). To simplify the conversion task, almost all automatic G2P methods require the training data to be aligned in advance because it allows to discover the hidden relationships between grapheme(s) in the input word and phoneme(s) in the output phoneme-sequence. In this context, it is possible to say that the alignment is the correspondence between the orthographic and the phonetic forms of the word.

According to the earlier studies such as those described in [69, 81], the grapheme-to-phoneme alignments were manually done, however manual elaboration of alignments is very costly and language-dependent. Then, the use of automatic alignments has become the most preferable because it is the best solution in terms of time and cost. For languages with deep orthography such as English, automatic alignment is a difficult problem mainly due to the influence of many foreign words (or loan words) and the lack of transparency in the writing system of this languages. According to P. Taylor in 2005 [70], the lack of clarity in the English orthography adds complexity to the alignment task since any phoneme can potentially align to a maximum of four letters (e.g., the

TABLE 2.1: Unaligned training data extracted from the CMUdict corpus

Word	**Phoneme sequence**
ABRA	AA B R AH
ABREGO	AA B R EH G OW
ABRON	AH B R AA N
ABSCESS	AE B S EH S
ABSHIRE	AE B SH AY R
ABSORBERS	AH B Z AO R B ER Z
ABSTINENT	AE B S T AH N AH N T
ABUTS	AH B AH T S
ACACIA	AH K EY SH AH
ACADIA	AH K EY D IY AH
ACCEL	AH K S EH L
.....	.....
.....	.....
.....	.....
Word_n	Pronunciation_n

TABLE 2.2: Possible alignment cadidates in one-to-one alignment.

Word (Graphemes) :	P	R	O	N	O	U	N	C	I	N	G
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Alignment 1:	P	R	AH	N	AW	_	N	S	IH	NG	_
Alignment 2:	P	R	AH	N	_	AW	N	S	IH	_	NG
Alignment 3:	_	_	P	R	AH	N	AW	N	S	IH	NG

word "PLOUGH" → /P L AW/). The cases of four-to-one correspondences are not so common but two-to-one are numerous, for example, "ENOUGH" → /IH N AH F/. The cases where one letter aligns to more than one phoneme are less frequent but also deserve special attention. An example is the word "SIX" → /S IH K S/.

2.2.1.1 One-to-one alignment

In general, automatic epsilon-scattering method proposed by Black et al. in 1998 can be used to produce one-to-one alignment between letters and phonemes in the training data for G2P conversion [68]. In this case, each letter in the input word can be aligned to only one phoneme in the output phoneme sequence. For the cases where the number of letters–the word's length–is greater than that of phonemes, a null phoneme symbol representing a silent sound (noted as /_ / in Table 2.2) is introduced into phonetic representations to match the length of grapheme and phoneme strings.

The alignment process starts with the initial probability of mapping a grapheme g to a phoneme ϕ (annotated as $Prob(g, \phi)$) which is calculated based on the mapping counts. The initial probability table first adds the necessary null phoneme symbols into all possible positions in phonemic representations. This process is repeated for every word in the training lexicon. For instance, an example of three of the possible alignment candidates for the word "PRONOUNCING" is given in Table 2.2.

Such probabilistic initialization allows obtaining all possible imperfect alignment candidates (e.g., the third alignment in Table 2.2). The goal of epsilon-scattering algorithm is to maximize the probability that letter g matches phoneme ϕ and, therefore, to choose the best alignment from possible candidates. It is done by applying the Expectation-Maximization (EM), according to Dempster et al. [82]. The EM is associated with joint grapheme-phoneme probabilities. Under certain circumstances, the EM guarantees an increase of the likelihood function at each iteration until convergence to a local maximum. The obtained alignments are not always logical, e.g., the word "THROUGH" in the CMUdict corpus may be in some cases aligned to $/TH \_ R \_ \_ \_ UW/$. This alignment imposes the correspondence between grapheme 'H' and phoneme /UW/, which introduces additional ambiguity to the training data. One way to overcome this obstacle is to build a list of allowables as in Black et al. [68]. It is just a simple table, that does not require any expert knowledge of the language. The allowables table defines for each grapheme a set of phonemes to which they can be aligned. All other alignments are prohibited. Some words with very opaque relationship between letters and phonemes would require adjustments made in the allowables table in order to produce alignments.

Another modern way to find a relationship between letter and phonemes is to use dynamic programming (DP) algorithm. DP based alignment uses a letter-phoneme association matrix A, of the dimension $L \times P$, where L is the size of the letter set and P is the size of the phone set. At the first step, the matrix A is initialized in a naive way with the elements $a^0_{l,p}$ which are incremented each time the letter l and the phoneme p are found in the same word. At the next iteration $a^1_{l,p}$ are incremented if the letter l and the phoneme p are found in the same alignment position.

At this first iteration the nulls are introduced into the dictionary as a consequence of the DP matching where both phonemes and graphemes can be associated with nulls. At the EM step, the matrix A is updated in a way that the word alignment score is maximized. Nulls are not entered as a part of the updated matrix A in order to avoid the tendency to generate unnatural alignments. The role of nulls is restricted to the DP matching phase which can be considered a path-finding problem. DP is not only more efficient than epsilon-scattering method but also allows nulls in both letter and phoneme strings.

TABLE 2.3: An example of an alignment based on graphones

Word (Graphemes):	S	P	EA	K	ING
	↓	↓	↓	↓	↓
Phonemes:	S	P	IY	K	IH+NG

The alignment 'X' to $/K\ S/$ is done automatically while the epsilon scattering method requires an a priori introduction of double phonemes $/K/\ /S/$ to $/KS/$ or $/K+S/$.

2.2.1.2 Many-to-many alignment

Theoretically, finite state transducers and multi-gram models can use many-to-many alignments. In some previous works [22, 71, 83], the authors used G2P alignment as the first step to infer the pronunciations of unknown words. Bisani and Ney [71] baptized the alignment element as "graphone", or a grapheme-phoneme joint multi-gram, which is a pair $q = (g, \phi) \in Q \subseteq (G^* \times \Phi^*)$. Letter sequence and phoneme sequences can be of different length (G and Φ are the grapheme and phoneme sets respectively). An example in Table 2.3 shows that a word of eight graphemes (i.e., G = { 'S', 'P', 'E', 'A', 'K', 'I', 'N', 'G' }) are mapped to a sequence of six phonemes (i.e., Φ = { /S/, /P/, /IY/, /K/, /IH/, /NG/ }). As a result, five pairs/graphones are obtained after the alignment. This means that Q = {('S', /S/), ('P', /P/), ("EA", /IY/), ("K", /K/), ("ING", /IH+NG/)}.

Those graphones that map one phoneme to one letter are called singular graphones (e.g., the pairs ('S', /S/), ('P', /P/) and ('K', /K/)). Graphone alignments can be inferred by using hand-crafted rules, DP search with predefined alignment constraints or costs, or by an iterative estimation of alignment probabilities.

The best sequence of graphones is induced from the dictionary data by searching for the most probable sequence of graphones, first assigning uniform distributions to all possible graphones (within the manually set length constraints) and then applying the EM algorithm. After graphones are aligned joint multi-gram sequence model is applied to automatically derive pronunciations [21, 22].

On the other hand, some of the automatic phonemization methods do not require alignments since the letter-phoneme correspondences are calculated during the training, e.g., Hidden Markov Models (HMM) use Baum-Welsh training algorithm.

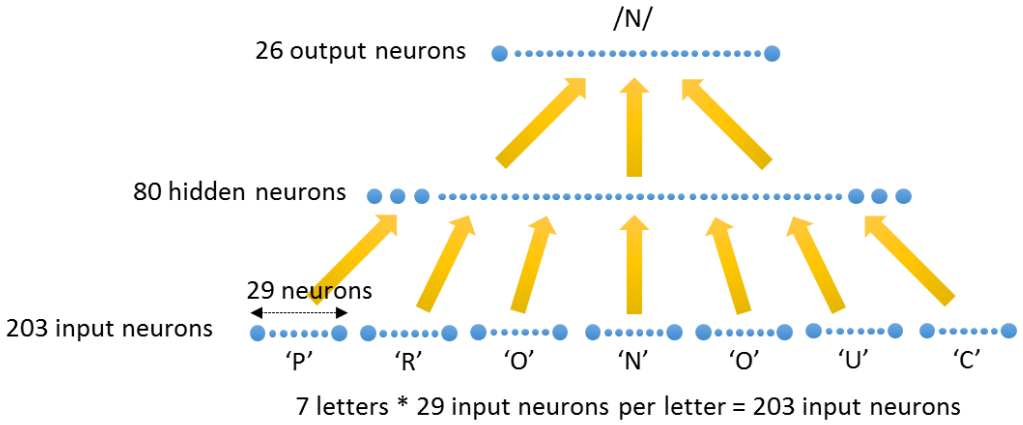


FIGURE 2.2: Architecture of the NETtalk system.

2.2.2 Artificial neural networks (ANNs)

There have been many data-driven based machine learning solutions for solving the problem of mapping arbitrary texts into phoneme strings. One of the first and best-known approaches to automated G2P conversion is the NETtalk system developed by Sejnowski and Rosenberg in 1987 [69]. The G2P problem was considered as a multi-class classification problem. The authors were pioneers in applying artificial neural networks using the back-propagation algorithm to a learning problem.

Fundamentally, NETtalk system was designed as a feed-forward multi-layered perceptron with three layers of units and two layers of weighted connections. It architecturally consisted of an input layer of letter units, a hidden layer, and an output layer of phoneme units. As depicted in Fig.2.2, the input layer received a 7 letter window, where the central letter represented the target/focusing grapheme to be converted, and the other 3 letters to its left and right sides represented the left- and right-contexts, respectively. To train such a system, each letter was encoded using a vector of 29 bits, 1 bit for each of the 26 letters of the English alphabet and 3 additional bits for the punctuation marks and word boundaries. Therefore, the total perceptron units at the input layer was equal to $7 \times 29 = 203$ units. Likewise, each of 54 output phonemes was encoded using a vector of 26 bits, 21 bits for 21 different articulatory features (such as voiced, unvoiced, points of articulation, plosive, nuclei, etc.), and 5 additional bits for representing the stress level and syllable boundaries. According to [84], the 54 phoneme symbols using in NETtalk system includes /a/, /b/, /c/, /d/, /e/, /f/, /g/, /h/, /i/, /k/, /l/, /m/, /n/, /o/, /p/, /r/, /s/, /t/, /u/, /v/, /w/, /x/, /y/, /z/, /A/, /C/, /D/, /E/, /G/, /I/, /J/, /K/, /L/, /M/, /N/, /O/, /Q/, /R/, /S/, /T/, /U/, /W/, /X/, /Y/, /Z/, /@/, /!/, /#/, /*/, / ^ /, /+/, /-/, /_ / and / . /.

As a pre-processing task, NETtalk required the data to be aligned in a one-to-one manner. Sejnowski and Rosenberg [85] manually aligned a 20,012 English word-based corpus created from MerriamWebsters Pocket Dictionary–called NETtalk corpus[5]. As explained in Section 2.2.1.1, when the number of letters exceeded the number of phonemes in a word, the so-called silent phonemes / _ / were inserted. In the opposite case, new double phonemes were invented for adjusting the length of phoneme sequence to the word's length; for example, the phonemes /*k*/ and /*s*/ in the word "axes" were joined in one /æ *k* _ *s*/. Based on various experimental results, the authors reported that a network with 80 hidden units were found to be a good match point between good performance and rather low computational complexity. However, the best results were obtained using 120 hidden units.

The continuous speech and isolated words from the dictionary were used to train and test the system. The continuous speech corpus of 1,024 words featured alternative pronunciations for the same words. In terms of phoneme accuracy, the best results achieved when tested on the continuation of the corpus (439 words) were 78% best guesses and 35% perfect matches. In addition, the system was also trained on a 1,000 word subset from a 20k corpus of most common English words. The number of hidden units varied across the experiments, and the best results on the training corpus were obtained using 120 hidden units. The same number of units was used to test the network on randomized version of 20,012 word-based dictionary. As a result, the best performance was 77% best guesses and 28% perfect matches.

The implemented method was language independent. The system had strong similarities to human learning and memorizing processes, however, it did not come close to modeling human reading acquisition skills yet. Regarding to the different evaluations conducted by Damper et al. [3], a problem concerning the generalization ability of the system had became the biggest issue since that system was never tested on unseen words. Furthermore, the phoneme error rate was not a good enough measure to compare the methods since the quality of synthesized speech could decrease quite quickly even if there was only one erroneous phoneme per word (as explained in Chapter 1). For these reasons, the system performance should be evaluated in terms of word error rate or word accuracy on a set of unseen words.

After then, McCulloch et al. presented an extension of NETtalk system in 1987 [81]. NETSpeak had a few changes in comparison with NETtalk. First of all, the authors claimed that a more concise representation of the input data would help achieving better performance. The number of input units was reduced to 11. The letters were grouped

[5]NETtalk dataset: ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/nettalk.data.gz

into 5 different mixed phonological sets according to the proximity of their manner of articulation with the exception of vowels which were all placed in one set. The remaining 6 bits were to indicate the position of the letter in the group. The output coding uses less phonological and more stress and punctuation features. The number of hidden units throughout all experiments was equal to 77. Another distinctive feature of this approach is that it was tested on a completely unseen set of words, however the authors used a different lexicon which makes the results difficult to compare. The results obtained on 1,200 unseen words by a network trained on 15,080 words from "Teachers Word Book" were equal to 86% of best guesses. The impact of word frequencies on the results was also studied. The words from the dictionary were replicated in appropriate proportions to make a distinction between common and uncommon words. The authors' hypothesis that the system would perform worse on common words due to their rather irregular G2P correspondences was not proved. A hybrid network that combined two separate networks trained on common an uncommon words was also trained and tested [81].

Besides, over the years many different neural networks-based methods for G2P conversion have been proposed for improving the predictive quality of the system when dealing with the unseen words [15, 20, 86].

2.2.3 Hidden Markov Models (HMMs)

Many data-driven techniques that are quite similar to the hand-written context sensitive rules, e.g., neural networks [3, 9, 10, 65], decision tree [8, 67, 68], pronunciation by analogy (PbA) [74, 87, 88], had been proposed to tackle the problem of G2P conversion. To overcome the difficult problem of phoneme prediction, another statistical-based solution using hidden Markov models (HMMs) was differently proposed in the late 20^{th} century [76, 89, 90, 91]. Then, the approach proposed by P. Taylor in 2005 [70] was the one very attractive HMMs-based method for G2P conversion in the early 21^{st} century.

In the HMM, the graphemes are seen as being generated by the phonemes via a noisy process, such that given the grapheme sequence, it is generally non-trivial to determine the phoneme sequence. In this method, the alignment between graphemes and phonemes were not required before the training because it was generated during the model training stage by Baum-Welch training [92] in which the HHMs used the probabilities of the G2P correspondences found in the previous step of the algorithm. Each phoneme is represented by one HMM while letters are the emitted observations. The probability of transitions between models is equal to the probability of the phoneme given the history (previous phoneme). The objective of this method is to find the most probable sequence of hidden models (phonemes) given the observations (letters), using the probability

distributions found during the model training. The standard formulation of HMMs can be written as follows:

$$\hat{\varphi} = argmax_y\{p(\varphi|g)\} = argmax_y\{p(g|\varphi)p(\varphi)\} \tag{2.4}$$

where $p(\varphi)$ is the prior probability of a sequence of phonemes occurring, and $p(g|\varphi)$ is the grapheme-phoneme joint sequence probability. One model trained for each phoneme; the maximum number of letters that a phoneme is able to generate is set to 4, since it is uncommon that more than four letters represent a single sound, at least in English. No looping states are allowed unlike in the general model configuration that serves for speech recognition. In the phoneme domain, certain constraints and patterns determining the sequences of possible phonemes were imposed. This is similar to phonotactic grammar. Phonotactically illegal sequences could cause a severe problem for TTS because the synthesis system will not be able to generate a corresponding waveform. The automatic speech recognition toolkit can be used to train the HMM models and to decode graphemes into phonemes. However, to achieve better results, some pre-processing was needed. Some letters were swapped and words rewritten. This measure was necessary because HMMs cannot model dependencies between observations. However, one of the advantages of the HMM is that they allow to model context-sensitivity in the phoneme domain. This was achieved by cloning the context independent models and applying further runs of Baum-Welch for those tokens of the training data that appeared more than 20 times. The experiments were carried out on Unisyn dictionary of approximately 110K words, most of which are regular English words. There are 42 phonemes in the Unisyn lexicon. The results obtained for a 4-gram model without preprocessing were 39.13% words and 85.12% phonemes correct, preprocessing allowed raising the bar to 49.64% and 87.02% words and phonemes correct correspondingly. Context-sensitive modeling brought the results up to 57.31% words and 90.98% phonemes correct. Stress prediction was included in the experiments. The large portion of errors consisted in schwa-full vowel confusions and stress misplacement.

As an extended work of HMMs-based method, Ogbureke et. al. proposed HMMs with context sensitive for G2P conversion in 2010 [16]. Previously, only phoneme context, which for first-order HMMs includes only the preceding phoneme, was used. In this work, both grapheme and phoneme contexts were modeled. In order to include grapheme context, each observation sequence was transformed increasing at the same time the number of possible observation symbols. No rewrites were necessary. Stress prediction was not considered. The approach combining context-sensitive grapheme, context-dependent phonemes and a 4-grams language model allowed obtaining 57.85% words correct for CMUdict and 79.19% for Unisyn lexicon for British English which is significantly better

in comparison to the results obtained in [70]. This shows that Increasing the number of observations allows obtaining higher accuracy.

In previous studies, only phoneme context and relationship between phonemes and letters were used to independently predict the phoneme corresponding to each input grapheme. According to our firstly proposed approach "Two-stage neural network-based G2P conversion" [19, 58] that will be described in Chapter 3, the benefit from involving both grapheme and phoneme contexts to the phoneme generation model could improve the predictive performance of G2P conversion system to another higher level. Inspired by our proposed two-stage architecture for automatic G2P conversion, coupled hidden Markov models (CHMM)-based method was then proposed by Che et al. in 2012 [17]. In this work, CHMM consists of two HMMs. The first HMM was designed to predict the best graphemic substring segmentation, in which the phoneme was considered as the states and the graphemeic substring represented the observations. On the other hand, another HMM was used to generate the best phonemic string; here, the phoneme represented the observations and the graphemic substring represented the states. All the reasonable graphemic substring segmentations were given before generating phonemes, and then the best combination of phonemic string and graphemic substring segmentation was given by maximizing the joint likelihood of two HMMs. As a result, the authors reported the word accuracy of 74.6% and 94.2% for CMUdict and OALD corpus, respectively.

2.2.4 Joint multigram models

In the alphabetic written system defined by a language, the orthographic form is a conventional representation of a word's pronunciation. A word can be viewed as a stream of graphemes (alphabets or letters), hence the word pronouncing system is highly dependent to the word's length and the internally hidden interactions among the alphabets. In order to model these kinds of dependencies, a joint multigram model, which is a statistical model that allows to learn variable length grapheme and phoneme from the training corpus and later to decode a string of orthographic symbols into a phonetic sequence, has been proposed.

In 1995, Deligne et al. proposed the very first time many-to-many alignments for G2P conversion [83]. Joint sequences of graphemes and phonemes of variable-length were extracted from the training lexicon using the maximum likelihood criterion. The maximum sizes of corresponding sequences were defined before the training. In this study, the algorithm was initialized by computing the relative sequences of all possible many-to-many alignments available from the training lexicon. Then, the authors trained two

different models using EM and Viterbi training algorithms. For the decoding process, it was carried out sequence-by-sequence and not grapheme-by-grapheme as in the majority of G2P classifiers. Different sequence sizes and thresholds (setting a minimum the number of times a consequence had to appear in the training corpus in order not to be discarded) were tested. As a result, the evaluation on a French lexicon BDLEX [93] containing 23,000 words and compounds showed that the best model achieved 64.52% and 95.0% as word and phoneme accuracy, respectively. In addition, thresholding was found very effective in order to improve the performance of the model on unknown words.

Likewise, Bisani and Ney applied a similar joint-multigram approach to align joint sequences of graphemes and phonemes in 2002 [71]. They introduced the term "graphones" (as seen in Table 2.3) to refer to the corresponding graphemic and phonemic chunks of variable length. The pronunciation of the unknown words was also inferred using the standard maximum likelihood training (EM algorithm) as well as Viterbi training. The minimum length of graphones was set to 1 and the maximum to 6 for both graphemic and phonemic domains. However, the best results for Celex lexicon (CELEX) containing 66,278 words were obtained using a 3-gram model. Longer graphones were more difficult to estimate, however the alignments restricted to one-to-one graphones seemed to perform worse than when longer chunks were involved. Thresholding and marginal trimming were used to enhance the models. As a result, The best model achieved 95.02% as phonemes correct rate.

In 2001, Galescu and Allen built a similar 4-gram model although they used a different alignment procedure [94]. Each letter-to-phoneme correspondence was restricted to having at least one grapheme and one phoneme, these correspondences were inferred using the EM algorithm. The performance was evaluated on two English lexica: NETtalk and CMU pronuncing dictionaries. The experiments included stress prediction, however only for latter lexicon. A back-off n-gram model with Witten-Bell discounting was used to train the model. One-to-one manually proofed alignment available for the NETtalk data was also evaluated in the experiments, showing that chunk-based alignments perform slightly better. The results obtained on NETtalk data were 63.93% words and 91.74% phonemes correct. For CMU including the stress markers 62.6% word and 91.0% phoneme accuracies were obtained. When phonemes were predicted disregarding the stress, the corresponding accuracies were 71.5% words and 93.6% phonemes correct. Furthermore, the authors also carried out the reverse task of predicting letters from phonemes using the same models.

On the other hand, in 2003, Chen aligned letters and phonemes using a conditional Maximum Entropy (ME) model with Gaussian priors [22]. Nulls symbols were allowed both in grapheme and phoneme strings, and the letters and phonemes were continuously

realigned during training unlike other previously fixed chunk models [71, 94]. To train a joint maximum entropy, 8-gram both conventional and Viterbi versions of the EM algorithm were used. The results were evaluated on Pronlex lexicon containing 91,216 English words in which the stress markers were not included. Syllable boundaries used as an attempt to enhance the model by preventing the syllable splitting, were found rather ineffective. The results were obtained for three datasets: regular words, proper names and foreign words. For regular words, the accuracies obtained were 72.7% for words and 92.85% for phonemes.

After then, in 2008, Bisani and Ney [21] used a similar model as in their previous work [71] and tested the performance of their system over a variety of English datasets in order to make their results comparable to those reported in literature. Moreover, they studied different model initialization and training schemes, the influence of the held-out set and the effect of different smoothing techniques and the size of graphones on the overall results. The evaluation results showed that the joint multigram models proposed performed better or as good as best performing G2P methods. The results obtained for OALD lexicon were 82.51% words and 96.46% phonemes correct. For NETtalk dictionary (size variable form 15K to 19K) the results ranged between 66.33% to 69.00% for word accuracies and from 91.74% to 92.34% for phoneme accuracy. For CMU dictionary the 75.47% words and 94.22% phonemes correct were obtained. For Pronlex the corresponding accuracies were 72.67% and 93.22% words and phonemes correct. For BEEP dataset[6], 79.92% words correct and 96.46% phonemes were obtained. Since then, joint models have been believed to be beneficial because they can handle the alignment problem intrinsically.

Soon after, in 2009, Jiampojamarn et al. represented the joint n-grams model for G2P conversion as an online discriminative sequence-prediction model [23, 95]. This model used a many-to-many alignment between grapheme and phoneme sequences and a feature vector consisting of n-gram context features, HMM-like transition features, and linear-chain features. For each training iteration, the feature weight vector was updated using the margin infused relaxed algorithm (MIRA) proposed by Crammer and Singer [96]. MIRA modified the current weight vector by finding the smallest changes such that the new weight vector separates the correct and incorrect outputs by a margin of at least the loss for a wrong prediction. This system is known as DirecTL [23]. They conducted experiments on several English and French corpora, including CELEX, Beep, OALD, CMUdict, NETtalk and Brulex, used in [21]. Moreover, the homographs, one-letter words, punctuation, phrase and abbreviations were excluded from the datasets due to the conventions described in [21]. In terms of word accuracy, this system outperforms the previous joint-sequence model [21] on four out of six datasets. The authors reported

[6]Beep corpus: `ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/beep-1.0.tar.gz`

that the word accuracies of CELEX, Beep, OALD, Brulex, CMUdict and NETtalk datasets were 91%, 81,.6%, 89.6%, 94.6%, 72.5% and 68.1% achieved by their systems, while 88.6%, 79.9%, 82.5%, 93.8%, 75.5% and 69.0% achieved by the joint-sequence models, respectively. It seemed their systems achieved slightly lower results than the joint-sequence models on the CMUdict and NETtalk datasets. However, their winning results proved that the MIRA update algorithm in this method was very effective in updating feature weights for distinguishing between correct and incorrect output results.

Last but not least, the updated version of DirecTL was implemented in 2010 and known as DirecTL+ toolkit[7] [24, 25]. In this system, the joint n-gram features were additionally integrated, which allowed the discriminative model to train on information that was present in generative joint n-gram models, and additionally trained on rich source-side context, transition, and linear-chain features. In the experiments, size of the joint n-gram features was set to 6. In terms of word accuracy, the authors reported that the DirecTL+ obtained 89.23%, 76.41%, 85.54%, 73.52% and 95.21% for CELEX, CMUdict, OALD, NETtalk and Brulex datasets, respectively; otherwise, the DirecTL system obtained only 88.54%, 75.41%, 82.43%, 70.18% and 95.03% for the same datasets, respectively. This showed that the additional joint n-gram features was very effective in improving the transliteration performance of the previous discriminative approaches as DirecTL system.

2.2.5 Weighted finite-state transducers (WFST) and others

In 2002, Caseiro et al. built a data-driven-based G2P conversion for European Portuguese by using weighted finite-state transducer (WFST) framework [97]. First, each grapheme sequence and its corresponding phoneme sequence in the training data were aligned using edit distance algorithm. In most case, one-to-one grapheme-phoneme correspondences (singular graphone) were used. However, two-to-one, one-to-two, one-to-three and one-to-four grapheme-phoneme alignments were also used for allowing the direct matching of some special sequences. Then, the n-gram language model was computed based on the aligned training data–joint sequences. Next, the authors implemented G2P conversion model by transforming the n-gram language model into a finite state transducer, and each pair of grapheme-phoneme symbols into a pair of input/output symbols. In decoding phase, a best-path search was needed to be computed through the WFST model in order to find the most likely phoneme string corresponding to an input grapheme sequence. Due to the fact that WFST is flexible in integrating multiple sources of information and other interesting properties, the WFST framework has been utilized throughout, following the approach outlined in [98, 99].

[7] https://code.google.com/p/directl-p/

Over the recent years, another WFST-based method for G2P conversion proposed in 2012 by Novak et al. [26] has been implemented to develop a rapid and high-quality joint-sequence model-based G2P conversion. First, the training words and their phoneme sequences were provided, and these were aligned using an expectation-maximization training procedure based on the many-to-many (m-to-m) aligning technique [21]. In this work, the maximum letter-phoneme correspondence m was set to 2, and both the null grapheme and phoneme symbols were allowed in both sides during the alignments. The obtained joint-sequence corpus was given as an input for n-gram counting (in which the order or length of the n-grams to count was provided), and then a standard joint n-gram model was trained using the MITLM tookit[8] or the OpenGrm NGram library,[9] and smoothed by Kneser-Ney discounting with interpolation. Then, the trained n-gram model was converted to a WFST-based model, which predicted the phoneme sequences of unknown words using the following decoding function:

$$Phseq_{best} = shortestPath(Project_o(W \circ_o M)) \tag{2.5}$$

where "$Phseq_{best}$" refers to the most likely phoneme sequence given the input word "W" under the FSA representation and the n-gram model "M" encoded as FST, "$\circ_o$" refers to the weighted composition, "$Project_o(.)$" is a projection onto the output symbols, and "$shortestPath(.)$" indicates the shortest-path algorithm. This work also investigated N-best re-scoring with a recurrent neural network language model (RNNLM)[100]. In order to train each RNNLM, the aligned corpus of joint grapheme-phoneme sequences was utilized as inputs. The evaluation results showed that the proposed system using RNNLM re-scoring technique could achieve small but consistent improvement over previous approaches, joint-sequence models in Sequitur [21] and DirecTL+ [25] toolkits) by providing 71.14%, 83.52% and 75.56% as word accuracy for NETtalk-19k, OALD and CMUdict, respectively. The extended work of this approaches proposed by the same group of authors in 2013 did not really show any better performance compared to the old one [27].

Besides the approaches mentioned above, the joint sequence models have been differently and successfully used to implement the structured online discriminative learning methods, such as structured AROW [33] and NAROW [34]. Recently, an SSMCW-based method [35] has been proposed for extending multi-class confidence-weighted learning to structured learning, which softens the marginal errors for hypothesis and update parameters using the N-best hypotheses simultaneously and interdependently for robustness against over-fitting. These learning methods are available in Slearp toolkit [33, 34, 35, 80].

[8] https://code.google.com/p/mitlm/
[9] http://www.openfst.org/twiki/bin/view/GRM/NGramLibrary

TABLE 2.4: Summary of results of previous G2P methods for different corpora. Although the methods in this table used the same corpus, they might differently subdivide the training, development and testing datasets; they also might use different K-folds cross-validation. Due to these facts, the results reported here are somehow incomparable.

Dataset	**Author**	**G2P method**	**Toolkit**	**PAcc (%)**	**WAcc (%)**
NETtalk	Sejnowski and Rosenberg (1987)	ANNs	-	78.00	35.00
	Torkkola (1993)	DT	-	90.80	-
	Andersen et al. (1996)	DT	-	89.90	53.00
	Bakiri and Dietterich (1997)	DT	-	-	64.80
	Jiang et al. (1997)	DT	-	91.90	65.80
	Yvon (1996a)	PbA	-	-	65.96
	Damper and Eastmond (1997)	PbA	-	91.20	60.70
	Marchand and Damper (2000)	PbA	-	92.40	65.50
	Galescu and Allen (2001)	Joint N-gram	-	91.74	63.93
	Bisani and Ney (2008)	Joint N-gram	Sequitur	-	69.00
	Jiampojamarn et al. (2009)	MIRA	DirecTL	-	70.18
	Jiampojamarn et al. (2010)	MIRA	DirecTL+	93.30	71.82
	Kubo et al. (2013)	SAROW	Slearp	93.25	71.44
	Kubo et al. (2014)	NAROW	Slearp	93.47	72.03
	Kubo et al. (2014)	SSMCW	Slearp	93.63	72.66
NETtalk 15k	Bisani and Ney (2008)	Joint N-gram	Sequitur	91.74	66.36
	Lehnen et al. (2011)	CRF	-	90.50	60.20
	Novak et al. (2012)	WFST	Phonetisaurus	-	67.77
	Novak et al. (2013)	WFST	Phonetisaurus	91.76	66.41
NETtalk 18k	Bisani and Ney (2008)	Joint N-gram	Sequitur	92.17	68.21
NETtalk 19k	Bisani and Ney (2008)	Joint N-gram	Sequitur	92.34	69.00
	Novak et al. (2012)	WFST	Phonetisaurus	-	71.14
Noisy NETtalk	Bisani and Ney (2008)	Joint N-gram	Sequitur	90.22	65.99
	Jiampojamarn et al. (2010)	MIRA	DirecTL+	89.67	66.48
	Kubo et al. (2013)	SAROW	Slearp	90.21	66.98
CMUdict	Andersen et al. (1996)	DT	-	91.10	57.90
	Jiang et al. (1997)	DT	-	91.80	73.10
	Black et al. (1998b)	DT	-	91.95	57.80
	Pagel et al. (1998)	DT	-	87.84	62.79
	Ogbureke et al. (2010)	HMM	-	-	57.85
	Che et al.(2012)	HMM	-	-	74.60
	Galescu and Allen (2001)	Joint N-gram	-	93.62	71.50
	Bisani and Ney (2008)	Joint N-gram	Sequitur	94.22	75.47
	Jiampojamarn et al. (2009)	MIRA	DirecTL	-	75.41
	Jiampojamarn et al. (2010)	MIRA	DirecTL+	-	76.41
	Novak et al. (2012)	WFST	Phonetisaurus	-	75.56
	Novak et al. (2013)	WFST	Phonetisaurus	94.15	75.58
	Kubo et al. (2013)	SAROW	Slearp	93.85	73.52
	Kubo et al. (2014)	NAROW	Slearp	93.89	73.54
	Kubo et al. (2014)	SSMCW	Slearp	93.91	73.72
Celex	Bisani and Ney (2008)	Joint N-gram	Sequitur	97.50	88.58
	Jiampojamarn et al. (2009)	MIRA	DirecTL	-	88.54
	Jiampojamarn et al. (2010)	MIRA	DirecTL+	-	89.23
	Lehnen et al. (2011)	CRF	-	97.00	85.60
	Lehnen et al. (2013)	HCRF	-	97.50	88.30
	Kubo et al. (2013)	SAROW	Slearp	97.49	88.19
	Kubo et al. (2014)	NAROW	Slearp	97.70	88.83
	Kubo et al. (2014)	SSMCW	Slearp	97.76	89.29
BEEP	Bisani and Ney (2008)	Joint N-gram	Sequitur	96.46	79.92
	Kubo et al. (2013)	SAROW	Slearp	97.81	88.27

TABLE 2.5: Table 2.4 (Continued)

Dataset	Author	G2P method	Toolkit	PAcc (%)	WAcc (%)
Brulex	Bisani and Ney (2008)	Joint N-gram	-	-	93.75
	Jiampojamarn et al. (2009)	MIRA	DirecTL	-	95.03
	Jiampojamarn et al. (2010)	MIRA	DirecTL+	-	95.21
	Kubo et al. (2013)	SAROW	Slearp	98.92	94.41
	Kubo et al. (2014)	NAROW	Slearp	99.01	94.86
	Kubo et al. (2014)	SSMCW	Slearp	99.01	94.89
OALD	Black et al. (1998b)	DT	-	95.80	74.56
	Pagel et al. (1998)	DT	-	93.60	76.66
	Bisani and Ney (2008)	Joint N-gram	Sequitur	96.46	82.51
	Jiampojamarn et al. (2009)	MIRA	DirecTL	-	82.43
	Jiampojamarn et al. (2010)	MIRA	DirecTL+	-	85.54
	Che et al.(2012)	HMM	-	-	94.20
	Novak et al. (2012)	WFST	Phonetisaurus	-	83.52
Teachers word book (TWB)	McCulloch et al. (1987)	ANNs	-	86.00	-
	Damper et al. (1998)	Elovitz rules	-	-	25.70
Unisyn	Taylor (2005)	HMM	-	90.98	57.31
	Ogbureke et al. (2010)	HMM	-	-	79.19
Pronlex	Chen (2003)	Joint N-gram	-	92.85	72.70
	Bisani and Ney (2008)	Joint N-gram	Sequitur	93.22	72.67
Wiktionary	Kubo et al. (2013)	SAROW	Slearp	78.77	39.81

Chapter 3

Two-Stage Neural Network-based G2P Conversion

Contents

3.1 Introduction

Fundamentally, some previous approaches [3, 70, 88] integrated many-to-one mapping techniques between letters and phonemes, in which a phoneme is determined by using a sequence of letters. These approaches proved unsatisfactory because there is no strict correspondence between letters and phonemes [11], 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 [101]. 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[10]) 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) [102]. The HMM-based approach with context-sensitive observations for G2P conversion [16], 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 [21], is one of the most popular approaches in G2P conversion. Recently, the Weighted Finite-State Transducer (WFST)-based G2P conversion [26] 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[11] could, in the same context, produce many possible corresponding output phonemes at the 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".

[10] CMUdict corpus is available in the Pascal Letter-to-Phoneme Conversion Challenge website (http://pascallin.ecs.soton.ac.uk/Challenges/PRONALSYL/Datasets)

[11] In this chapter, a grapheme is strictly equal to a single letter, rather than a spelling unit.

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 chapter 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 [86]. Most of these approaches were constructed as one-stage models [3, 20], so they were not integrated with the many-to-many mapping technique between graphemes and phonemes. Considering these facts, in this chapter, 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.

The remainder of this chapter is organized as follows: In Section 3.2, 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 Section 3.3, and present its experimental results in Section 3.4. We discuss the experimental results by investigating the error analysis in Section 3.5 and then conclude this chapter in Section 3.6.

3.2 Single-stage neural network-based G2P conversion

The G2P conversion model was established for use in predicting the phonemes corresponding to the input text[12], 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, 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 _ /.

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

3.2.1 Mapping technique between graphemes and phonemes

The context-dependent grapheme model considers the association between graphemes and phonemes as *many-to-one* [3]. Thus, the extracted *g-p* pairs are obtained by passing two different slicing windows [103] 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 [20]. In this context, several graphemes as input and a single phoneme as output are required. For example, if the word $G = g_1 g_2 ... g_n$ corresponds to the phoneme sequence $P = p_1 p_2 ... p_n$, then the extracted pair between the focal grapheme g_i at position i (where $i = 1...n$) and its corresponding phoneme p_i is represented as below:

$$\begin{aligned} \underbrace{g_{i-x} + ... + g_{i-1} + \ g_i \ + g_{i+1} + ... + g_{i+x}} &\rightarrow p_i \\ \Leftrightarrow \qquad seq(g_i, x) &\rightarrow p_i \\ \text{Where} \ \ g \in \{\text{`}A\text{'}, \text{`}B\text{'}, ..., \text{`}Z\text{'}, \text{empty grapheme `\_'}\} & \\ p \in \{/AA/, /AE/, ..., \text{empty phoneme } /\_/\} & \end{aligned} \tag{3.1}$$

Here, the sign + denotes sequence concatenation. The segments $(g_{i-x} + ... + g_{i-1})$ and $(g_{i+1} + ... + g_{i+x})$ 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) [16, 23, 31, 101]. 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.(3.1) 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.(3.2) becomes Eq.(3.1) once the parameter y is set to zero.

$$\begin{aligned} \underbrace{g_{i-x} + ... + g_i + ... + g_{i+x}} \ &\rightarrow \ \underbrace{p_{i-y} + ... + p_i + ... + p_{i+y}} \\ \Leftrightarrow \ \ seq(g_i, x) \ &\rightarrow \ seq(p_i, y) \end{aligned} \tag{3.2}$$

3.2.2 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 [3], [20]. Table 3.1

TABLE 3.1: 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)							**Output** no context (1 ph.)	**Output** +phoneme context (5 phonemes)				
	$seq(g_i, 3)$							$seq(p_i, 0)$	$seq(p_i, 2)$				
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***	_	_

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 [104]. 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.

3.3 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.

3.3.1 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:

$$\textit{Grapheme sequence} \Rightarrow \textit{Phoneme sequence} \Rightarrow \textit{Phoneme}$$

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.

3.3.2 Architecture of the G2P conversion model

On the basis of the new phoneme prediction method presented above, the proposed G2P conversion model is fundamentally built by putting two different multi-layer neural networks in sequence as depicted in Fig.3.1. 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

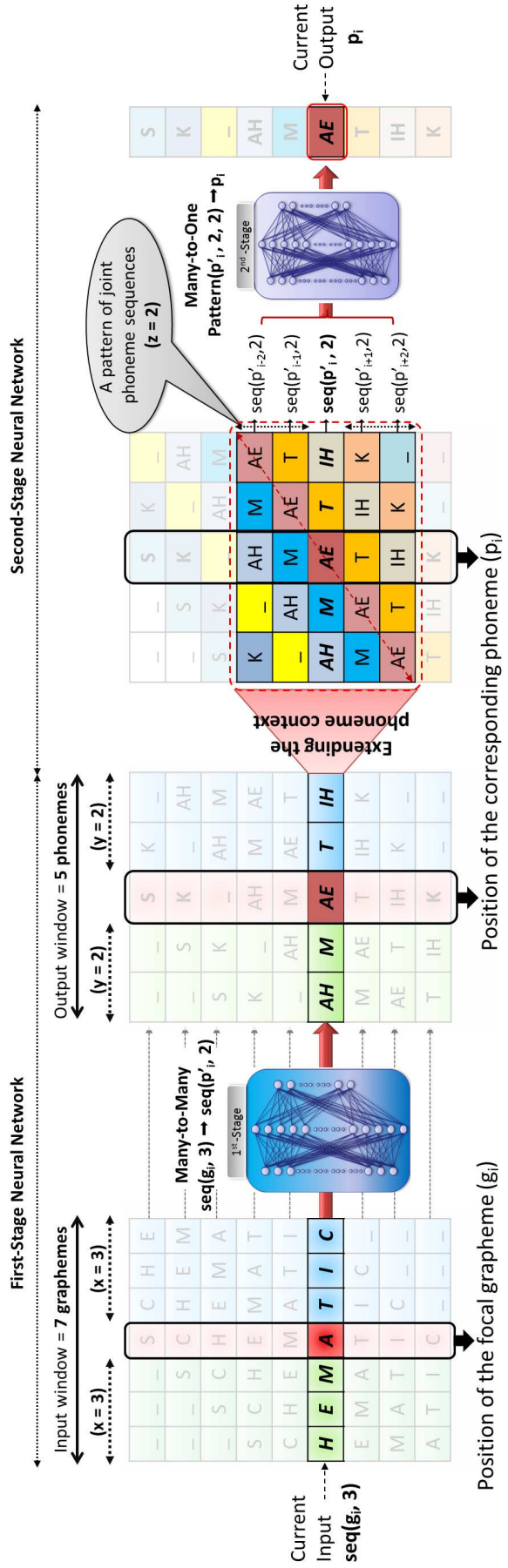


FIGURE 3.1: Architecture of a Two-Stage Neural Network-based approach for G2P conversion, performed on the occurrence "SCHEMATIC" → /S K - AH M AH T IH K/ while $x = 3$ (seven graphemes), $y = 2$ (five phonemes) and $z = 2$ (five sequences).

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.

3.3.2.1 First-stage neural network

As depicted in Fig.3.1, the first-stage neural network is constructed based on the same technique described in Section 3.2, 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 respect to Eq.(3.2) from all the occurrences of the pronunciation dictionary. For example, according to Table 3.1, 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.3.2, 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.

3.3.2.2 Second-stage neural network

According to Fig.3.1, for an input word $G = g_1 g_2 ... g_n$ 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.3.1, 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.3.1, 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 chapter, 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:

$$\begin{aligned} &\overbrace{\underbrace{seq(p'_{i-z}, y) + ...}_{\text{preceding sequences}} + seq(p'_i, y) + \underbrace{... + seq(p'_{i+z}, y)}_{\text{succeeding sequences}}}^{} \rightarrow p_i \\ &\Leftrightarrow \; Pattern(p'_i, y, z) \rightarrow p_i \end{aligned} \tag{3.3}$$

Owing to the problem presented in Table 3.1, 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.3.2 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.3.2, these errors could help to produce some extra patterns for the second-stage sometimes.

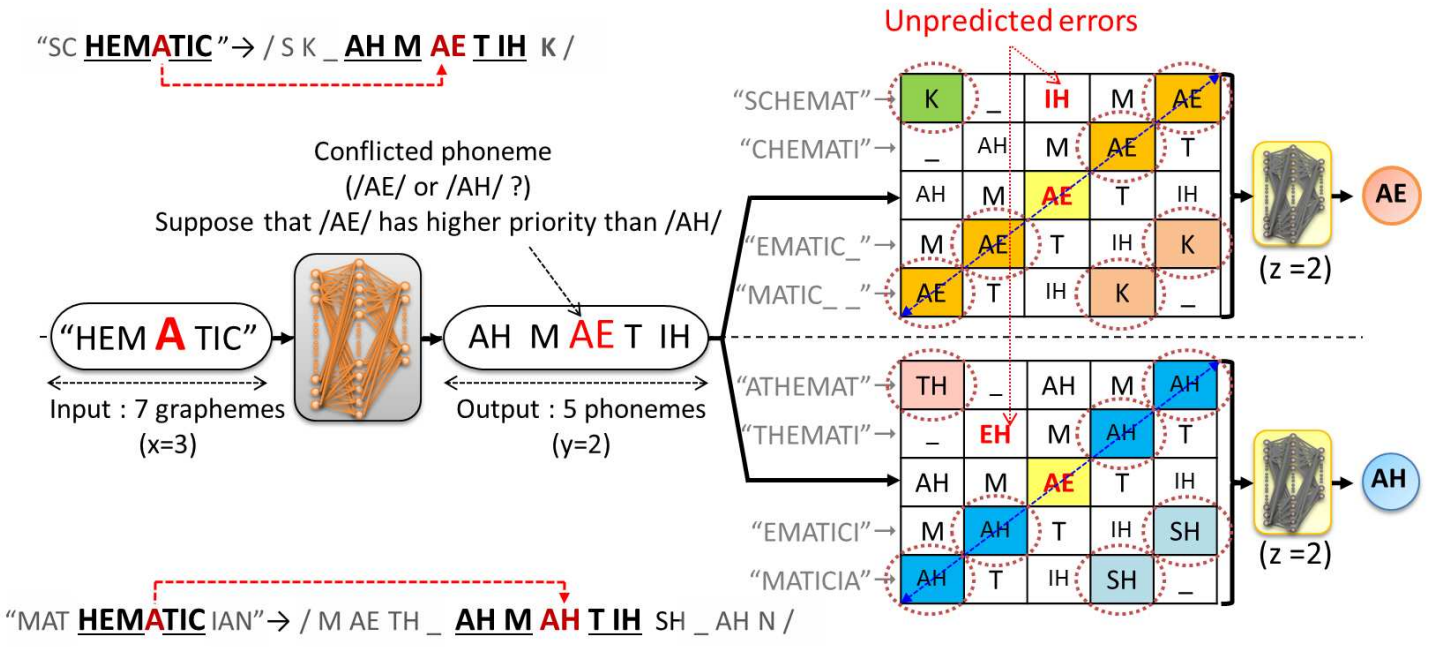


FIGURE 3.2: An example that demonstrates how to solve the phoneme conflicts in G2P conversion using the two-stage neural network-based approach.

3.4 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.

3.4.1 Data preparation

3.4.1.1 Auto-aligned CMUdict corpus

We chose the American English words-based pronunciation dictionary (known as the auto-aligned CMUdict corpus) 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 [16, 101, 102]. 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, part1, ..., part9) each of which contains almost the same number of words, graphemes as well as phonemes [58].

3.4.1.2 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 [16, 23], while others have emphasized that the quality of the pronunciation dictionary could negatively affect the G2P conversion performance [105]. 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.

The resulting corpus proved more reliable and consistent than the original. Fig.3.3 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.3.4 demonstrates that the grapheme in

Case of alignment	Original CMUdict	Newly aligned CMUdict *(After 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 MP O S I **T I O N** D IY K AH M P OW Z IH **SH _ AH N**	D E C O MP 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 CL I **M _ _ A T I O N** AE _ K L AH **M EY SH AH _ _ _ N**	A CCL 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 MA **T I O N** D EH F AH M EY **SH _ AH N**
-ED	CR A MP **E D** K R AE M P **T _**	B I A S **E D** B AY AH S **T _**	CR 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** LN E S S **AO F AH _ _ _** L N AH _ S	**A W F U** LN E S S **AO _ F AH** L N AH S _
	B _ _ I C Y C LE D **B AY S IH K _ AH** L _ D	**B I C Y C _** LE 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 QUE** K AH **S IY K _ _ _ _**	C A **C I QU E** K AH **S IY K _ _**

FIGURE 3.3: Comparison of the alignment between graphemes and phonemes in the auto-aligned CMUdict (column 2) and the newly aligned CMUdict (column 3).

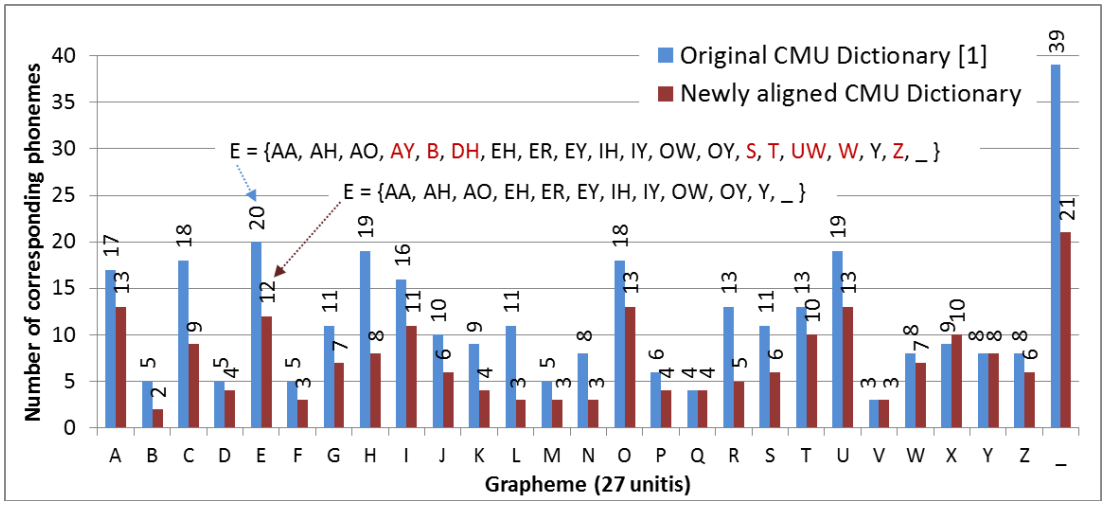


FIGURE 3.4: Consistency measurement based on the number of corresponding phonemes that could be mapped by each grapheme inside the original and new datasets.

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.

3.4.2 Experimental setup

3.4.2.1 Training and testing datasets

We conducted experiments on the newly aligned CMUdict corpus. Nine out of ten 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) [15] 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. According to Tables 3.2 and 3.3, for each vector, only one element at a specific index was active or set to one, while the others were set to zero.

3.4.2.2 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.3.3, this baseline was built using Eq.(3.1) or Eq.(3.2) 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.3.5, 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., $seq(p'_i, 0) = p'_i$).

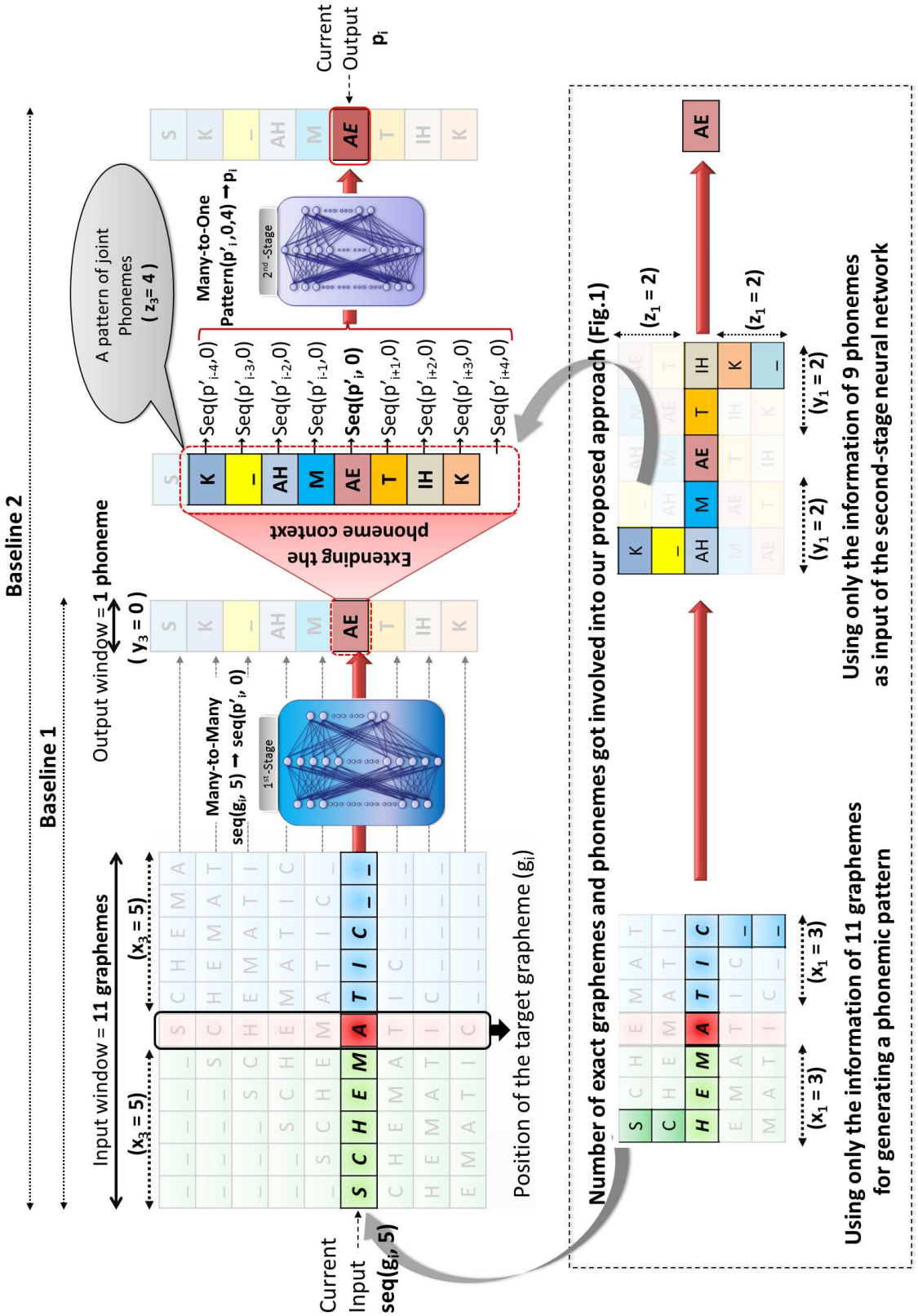


FIGURE 3.5: Architecture of Baseline1 and Baseline2.

Grapheme symbol	Encoded value (27 neurons)
'A'	100000000000000000000000000
'B'	010000000000000000000000000
'C'	001000000000000000000000000
'D'	000100000000000000000000000
'E'	000010000000000000000000000
'F'	000001000000000000000000000
'G'	000000100000000000000000000
'H'	000000010000000000000000000
'I'	000000001000000000000000000
'J'	000000000100000000000000000
'K'	000000000010000000000000000
'L'	000000000001000000000000000
'M'	000000000000100000000000000
'N'	000000000000010000000000000
'O'	000000000000001000000000000
'P'	000000000000000100000000000
'Q'	000000000000000010000000000
'R'	000000000000000001000000000
'S'	000000000000000000100000000
'T'	000000000000000000010000000
'U'	000000000000000000001000000
'V'	000000000000000000000100000
'W'	000000000000000000000010000
'X'	000000000000000000000001000
'Y'	000000000000000000000000100
'Z'	000000000000000000000000010
'_'	000000000000000000000000000

TABLE 3.2: List of grapheme symbols and its encoding.

Phoneme symbol	Encoded value (40 neurons)
/AA/	1000000000000000000000000000000000000000
/AE/	0100000000000000000000000000000000000000
/AH/	0010000000000000000000000000000000000000
/AO/	0001000000000000000000000000000000000000
/AW/	0000100000000000000000000000000000000000
/AY/	0000010000000000000000000000000000000000
/B/	0000001000000000000000000000000000000000
/CH/	0000000100000000000000000000000000000000
/D/	0000000010000000000000000000000000000000
/DH/	0000000001000000000000000000000000000000
/EH/	0000000000100000000000000000000000000000
/ER/	0000000000010000000000000000000000000000
/EY/	0000000000001000000000000000000000000000
/F/	0000000000000100000000000000000000000000
/G/	0000000000000010000000000000000000000000
/HH/	0000000000000001000000000000000000000000
/IH/	0000000000000000100000000000000000000000
/IY/	0000000000000000010000000000000000000000
/JH/	0000000000000000001000000000000000000000
/K/	0000000000000000000100000000000000000000
/L/	0000000000000000000010000000000000000000
/M/	0000000000000000000001000000000000000000
/N/	0000000000000000000000100000000000000000
/NG/	0000000000000000000000010000000000000000
/OW/	0000000000000000000000001000000000000000
/OY/	0000000000000000000000000100000000000000
/P/	0000000000000000000000000010000000000000
/R/	0000000000000000000000000001000000000000
/S/	0000000000000000000000000000100000000000
/SH/	0000000000000000000000000000010000000000
/T/	0000000000000000000000000000001000000000
/TH/	0000000000000000000000000000000100000000
/UH/	0000000000000000000000000000000010000000
/UW/	0000000000000000000000000000000001000000
/V/	0000000000000000000000000000000000100000
/W/	0000000000000000000000000000000000010000
/Y/	0000000000000000000000000000000000001000
/Z/	0000000000000000000000000000000000000100
/ZH/	0000000000000000000000000000000000000010
/_ /	0000000000000000000000000000000000000000

TABLE 3.3: List of phoneme symbols and its encoding.

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*).

TABLE 3.4: 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

As can be seen in Fig.3.1, 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.3.5 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 3.4, $x_3 = x_1 + z_1 = x_1 + 2$) 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 3.4, $z_3 = z_1 + y_1 = 4$).

According to Eq.(3.1), for each test set in Table 3.4, 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 $x_3 = x_1 + 2 = 5 \rightarrow 9$).

3.4.2.3 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[13]) library. We obtained the best results when each stage was set up as follows:

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

- Standard neural network with three layers
- Incremental backpropagation algorithm
- Learning rate = 0.8; Momentums = 0.1
- Symmetrical sigmoid activation function[14], where the 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.

3.4.2.4 Accuracy measurements

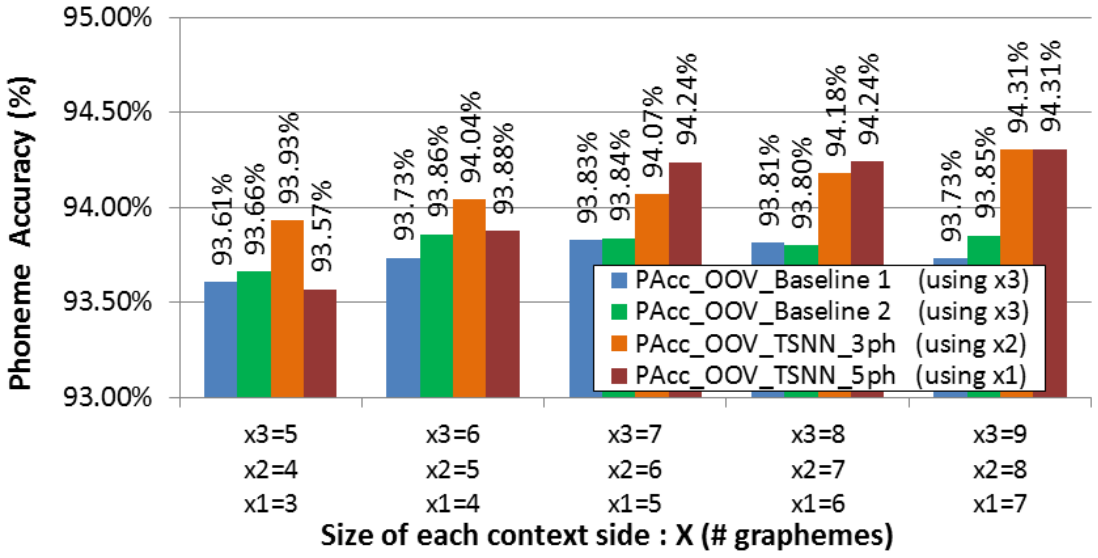
To compare with other approaches introduced at the beginning of this chapter, we evaluated the performance of the model in terms of *phoneme accuracy* (PAcc) and *word accuracy* (WAcc) using the NIST sclite scoring toolkit[15]. Because the goal of this chapter is improvement of the performance of G2P conversion measured on the OOV words, we only report results related to this objective. Both PAcc and WAcc are calculated using Eq.(2.2) and Eq.(2.3) written in Section 2.1 on page 17, respectively.

3.4.3 Experimental results

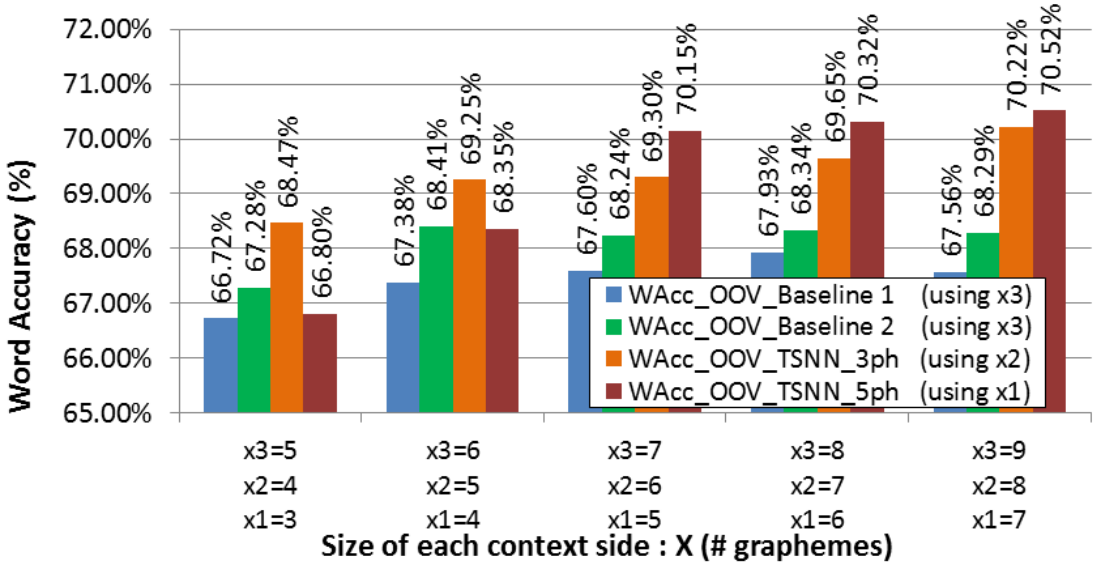
Each proposed test set used the newly aligned CMUdict corpus to evaluate the model performance. Based on Fig.3.6, by investigating various input grapheme sequence sizes (e.g., when $x_1 = 3 \rightarrow 7$, $x_2 = 4 \rightarrow 8$ and $x_3 = 5 \rightarrow 9$), our proposed two-stage neural network-based approach usually provided higher PAcc and WAcc than both baseline approaches.

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

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



(a) Phoneme Accuracy (PAcc)



(b) Word Accuracy (WAcc)

FIGURE 3.6: PAcc and WAcc measured on the OOV words

TABLE 3.5: 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 3.5 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 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 3.6. 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 3.6 shows that the training time per epoch at the first-stage neural

TABLE 3.6: 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	
	Context	**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]

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 3.7: 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 3.8: 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%

3.4.4 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 [26] available in the Phonetisaurus G2P toolkit[16].

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 3.7, 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 Section 3.4.2.1. 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 3.8 show that TSNN_5ph always provides higher phoneme accuracy than WFST, but, unfortunately, lower word accuracy for the first dataset.

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

3.5 Discussion

The experimental results depicted in Fig.3.6 and Table 3.5 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., $x_1 < x_3$).

As explained in Section 3.4.2.2, 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.3.6 and Table 3.5 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 3.8 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.

3.6 Summary

This chapter 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. 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.

Chapter 4

Grapheme Generation Rules for Two-Stage Model-based G2P Conversion

Contents

4.1 Introduction

The two-stage architecture for G2P conversion described in the previous chapter showed the advantage of using phonemic rather than graphemic information to predict the best final output phoneme sequence corresponding to the input word. It also demonstrated that this two-stage model using neural networks is good at identifying single phonemes in a word, but lacks the knowledge for detecting the whole word.

Therefore, in this chapter, we utilize the existing WFST-based approach to implement a novel two-stage architecture-based G2P conversion. This work investigates a new strategy in which we combine both graphemic and phonemic information as the input sequence for the G2P conversion. Moreover, several new grapheme generation rules for transforming each input word into different representations of grapheme sequences are also introduced in this chapter, which enable the addition of extra detail to the vowel and consonant graphemes appearing in a word. In this study, a grapheme could be a single letter or a combination of letters. Most of these rules focusing on the vowel graphemes can achieve a small but consistent improvement on previous approaches.

The remainder of this chapter is organized as follows: in Section 4.2, we introduce several newly invented grapheme generation rules. Then, we describe the novel two-stage model for G2P conversion in Section 4.3 and provide the evaluation results in Section 4.4. The discussion and conclusion are in Section 4.5 and 4.6, respectively.

4.2 New grapheme generation rule (GGR)

The G2P conversion model is usually built as a one-stage architecture for use in predicting phonemes corresponding to input text, especially with OOV words. To improve the model's performance, this research integrated various newly invented grapheme generation rules into the model.

The grapheme side does not carry sufficient information or knowledge relating to the phonological interaction [104]. In order to make the graphemic information more sensitive in the G2P conversion, this work designed new rules with respect to the concept of context-dependent models, particularly for generating different grapheme sequences out of the same input word. Theoretically, for each grapheme of a given word, we concatenate it with the graphemes on its left and right contexts. However, in this study, only the right context information is involved in the rule-making process because we prefer a compact representation for the new grapheme symbols, each of which consists of one or two alphabetical letters (e.g., "A" or "AU").

TABLE 4.1: List of the selected grapheme generation rules.

	Rule (GGR_r)	**Description** (**Word** $W = g_1 g_2 ... g_{m-1} g_m$ ⇒ **Grapheme sequence**) $\hat{G}_r = \bar{g}_1\ \bar{g}_2\ ...\ \bar{g}_{m-1}\ \bar{g}_m$		
	GGR_1	g_i	⇒	g_i (like unigram = default)
		Ex: "OKEECHOBEE"	⇒	O K E E C H O B E E
	GGR_2	g_i	⇒	$g_i g_{i+1}$ (like bigram)
		Ex: "OKEECHOBEE"	⇒	OK KE EE EC CH HO OB BE EE E_
Rules focusing on vowel graphemes	GGR_3	$v_1...v_n$	⇒	$v_1v_2\ v_2v_3\ ...\ v_{n-1}v_n\ v_n$
		Ex: "OKEECHOBEE"	⇒	O K **EE** E C H O B **EE** E
	GGR_4	If (n > 1): $v_1...v_nc_{n+1}$	⇒	$v_1v_2\ v_2v_3\ ...\ v_{n-1}v_n\ v_nc_{n+1}\ c_{n+1}$
		If (n > 1): $v_1...v_n \hookleftarrow$	⇒	$v_1v_2\ v_2v_3\ ...\ v_{n-1}v_n\ v_n$
		If (n = 1): g_i	⇒	g_i
		Ex: "OKEECHOBEE"	⇒	O K **EE EC** C H O B **EE** E
	GGR_5	If (n > 1): $v_1...v_nc_{n+1}$	⇒	$v_1v_2\ v_2v_3\ ...\ v_{n-1}v_n\ v_nc_{n+1}\ c_{n+1}$
		If (n > 1): $v_1...v_n \hookleftarrow$	⇒	$v_1v_2\ v_2v_3\ ...\ v_{n-1}v_n\ v_{n-}$
		If (n = 1): g_i	⇒	g_i
		Ex: "OKEECHOBEE"	⇒	O K **EE EC** C H O B **EE E**_
	GGR_6	If (n > 1): $[c_0]v_1...v_nc_{n+1}$	⇒	$[c_0v_1]\ v_1v_2\ v_2v_3\ ...\ v_{n-1}v_n\ v_nc_{n+1}\ c_{n+1}$
		If (n > 1): $[c_0]v_1...v_n \hookleftarrow$	⇒	$[c_0v_1]\ v_1v_2\ v_2v_3\ ...\ v_{n-1}v_n\ v_{n-}$
		If (n = 1): g_i	⇒	g_i
		Ex: "OKEECHOBEE"	⇒	O **KE EE EC** C H O **BE EE E**_
Rules focusing on consonant graphemes	GGR_7	$c_1...c_n$	⇒	$c_1c_2\ c_2c_3\ ...\ c_{n-1}c_n\ c_n$
		Ex: "APPLICATION"	⇒	A **PP PL** L I C A T I O N
	GGR_8	If (n > 1): $c_1...c_nv_{n+1}$	⇒	$c_1c_2\ c_2c_3\ ...\ c_{n-1}c_n\ c_nv_{n+1}\ v_{n+1}$
		If (n > 1): $c_1...c_n \hookleftarrow$	⇒	$c_1c_2\ c_2c_3\ ...\ c_{n-1}c_n\ c_n$
		If (n = 1): g_i	⇒	g_i
		Ex: "APPLICATION"	⇒	A **PP PL LI** I C A T I O N
	GGR_9	If (n > 1): $c_1...c_nv_{n+1}$	⇒	$c_1c_2\ c_2c_3\ ...\ c_{n-1}c_n\ c_nv_{n+1}\ v_{n+1}$
		If (n > 1): $c_1...c_n \hookleftarrow$	⇒	$c_1c_2\ c_2c_3\ ...\ c_{n-1}c_n\ c_{n-}$
		If (n = 1): g_i	⇒	g_i
		Ex: "APPLICATIONS"	⇒	A **PP PL LI** I C A T I O **NS S**_
	GGR_{10}	If (n > 1): $[v_0]c_1...c_nv_{n+1}$	⇒	$[v_0c_1]\ c_1c_2\ c_2c_3\ ...\ c_{n-1}c_n\ c_nv_{n+1}\ v_{n+1}$
		If (n > 1): $[v_0]c_1...c_n \hookleftarrow$	⇒	$[v_0c_1]\ c_1c_2\ c_2c_3\ ...\ c_{n-1}c_n\ c_{n-}$
		If (n = 1): g_i	⇒	g_i
		Ex: "APPLICATIONS"	⇒	**AP PP PL LI** I C A T I **ON NS S**_
	GGR_{11}	$GGR_3 + GGR_7$		
		Ex: "APPLICATION"	⇒	A **PP PL** L I C A T **IO** O N

Here, $g_i = \{c_i, v_i\}$: grapheme/character at index i;
c_i , v_i : consonant and vowel graphemes at index i;
'$\hookleftarrow$' : end of the word (counted as c_i);
'_' : empty consonant grapheme;
n : number of connecting vowels in a given word;
m : length of the given word.

Because the interaction between vowels in a word has a strong impact on the spelling process, most of the rules written in this chapter were carefully designed to add extra sensitive information to each vowel grapheme appearing in a word. For a few connecting graphemes many rules are possible, but only the rules more related to the vowel graphemes (as listed in Table 4.1) are taken into account. However, in order to compare the impacts of the vowel and the consonant grapheme in the automatic conversion of a word into its phonetic transcription, we also proposed some other rules that mainly focus on the consonant graphemes. As a result, Table 4.1 shows that most of the newly generated grapheme sequences can make the G2P conversion system easily identify not only the pattern of each vowel but also that of each consonant in a given word. In this table, the parameter g_i refers to the grapheme in index i, while c_i and v_i represent the consonant and vowel graphemes in index i, respectively. Moreover, the parameter n represents the number of vowels.

Suppose that an input word $W = g_1 g_2 ... g_m$ consisting of m characters/graphemes is provided as an input. The new grapheme sequence $\hat{G}_r = \bar{g}_1 \ \bar{g}_2 \ ... \ \bar{g}_m$, in which an empty space is used as a separator, can be generated with respect to a rule GGR_r, formulated as follows:

$$\hat{G}_r = GGR_r(g) \tag{4.1}$$

The first rule (GGR_1) represents a unigram model used by most researchers [10, 15, 16, 17, 19, 26, 59, 69, 70, 72, 88, 101, 102], but it appears not to provide sufficient information concerning each vowel or consonant grapheme. The second rule (GGR_2) represents a bigram model, which seems to add too much information to each grapheme because it always combines the consonant grapheme with the vowel grapheme. The other four rules (GGR_3, GGR_4, GGR_5 and GGR_6) are designed specifically for adding the information missing in the first rule. For example, the third rule (GGR_3) can distinguish the separated vowel–the vowel v that appears in the *cvc* pattern– from the vowels at the front part of the connecting vowels, i.e., the vowels $v_1, v_2, ..., v_{n-1}$ of the $v_1...v_n$ pattern. In addition to GGR_3, the other three remaining rules (GGR_4, GGR_5 and GGR_6) are capable of distinguishing between the front vowels $v_1, v_2, ..., v_{n-1}$ and the last vowel Vn of the $v_1...v_n c_{n+1}$ pattern. The use of the empty grapheme "_" in GGR_5 and GGR_6 permits the recognition of the difference between the last vowel v_n of the $c_0 v_1...v_n c_{n+1}$ pattern and that located at the end of word–the vowel v_n of the $c_0 v_1...v_n$ pattern. Moreover, GGR_6 adds more information to the consonant next to the connected vowels (e.g., the graphemes "KE" and "BE"). In addition, the rules GGR_7, GGR_8, GGR_9 and GGR_{10} are proposed for adding extra detail to the consonant graphemes appearing in the given word, which are designed with respect to GGR_3, GGR_4, GGR_5 and GGR_6, respectively. Furthermore, another rule GGR_{11} that combines GGR_3 with GGR_7 was

created to enable the addition of extra detail for both vowel and consonant graphemes appearing within a word.

4.3 Two-stage model for G2P conversion

The architecture of the two-stage model-based approach was first proposed in 2011 to address the problem of phoneme conflicts in G2P conversion [58]. This architecture was basically implemented by connecting two different multilayer neural networks in sequence, which improves the accuracy of the ordinary one-stage neural network-based G2P conversion [15, 59]. However, the evaluation results in the previous chapter (Chapter 3) demonstrated that the two-stage model using the Fast Artificial Neural Network (FANN) Library[17] lacks some knowledge for detecting the whole word, so it provided lower word accuracy but higher phoneme accuracy than the WFST-based G2P conversion availabel in the Phonetisaurus toolkit. Therefore, we used the existing WFST-based approach for G2P conversion [26] to employ a novel two-stage model-based approach.

4.3.1 Prediction using combined grapheme-phoneme (G-P) information

The phoneme prediction method, in which only the phonemic information is used as input to select the best final output phoneme, was first presented in our previous chapter. Its paradigm (*Graphemes* $\Rightarrow$ *Phonemes* $\Rightarrow$ *Phonemes*) shows that this method first converts the input word into phonemic information; then, all the related phonemic information is combined and used to predict the exact output phonemes of the G2P conversion model.

Because only the phonemic information is used in our previous method, we recognized that all of the words producing the same phoneme sequence (or pronunciation) during training in the first-stage are merged together before the second stage. For instance, the words "KOLL," "KOLLE," "CAUL," and "KAHLE" all generated the same phoneme sequence /K AA L/ at the first-stage, so only one sample /K AA L/ $\rightarrow$ /K AA L/ was used at the second stage. Furthermore, some wrong phoneme sequences may be obtained by accident because it is virtually impossible to obtain a perfectly trained first-stage model. Therefore, some training data could be incorrectly merged or ignored by the second-stage model. For example, the word "COALE" wrongly generates /K AA L/ as its output, while its correct phoneme sequence is /K OW L/. Therefore, it

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

is ignored by the second-stage model. Such a problem reduces the number of training data at the second-stage and negatively affects the performance of the model.

In order to address this problem, we propose a new phoneme prediction method in which the input graphemes and output phonemes obtained from the first stage are combined and used as the new input sequence to determine the best final output phoneme sequence corresponding to the input word. Therefore, our newly proposed method also consists of two steps as follows:

- First step : Graphemes → Phonemes
- Second step : Combined G-P pairs → Phonemes

4.3.2 Architecture of the proposed model

On the basis of the new phoneme prediction method presented in the previous section, the novel two-stage G2P conversion architecture is built using two main modules (i.e., first-stage and second-stage models) in sequence.

4.3.3 First-stage model

The first-stage model, implemented based on the original WFST-based G2P conversion presented in [26] and availabel in the Phonetisaurus toolkit[18], is used for the automatic conversion of a word to its corresponding phoneme sequence. As can be seen in Fig.4.1, this model is trained with pairs of words and their phoneme sequences and each input word must first be generated into a grapheme sequence by using any grapheme generation rule from Table 4.1. In this context, each grapheme is represented by a single letter (e.g. "A") or a combination of letters (e.g. "OA"), depending on the rule selected, and they are separated from one another by an empty space. Because it is virtually impossible to acquire a perfectly trained model, some unexpected errors will be produced at this stage.

For example, after training three words with almost the same pronunciation (e.g., "KOLL" → /K AA L/, "KOLLE" → /K AA L/, and "COALE" → /K OW L/), Fig.4.1 demonstrates that the word "COALE" generates "C OA A L E" as its grapheme sequence and then produces /K AA L/ as its output phoneme sequence with one error phoneme /AA/. Supposing that the other two words produce correct phoneme sequences, these three words all output the same phoneme sequence, /K AA L/.

[18]Phonetisaurus toolkit: https://code.google.com/p/phonetisaurus/

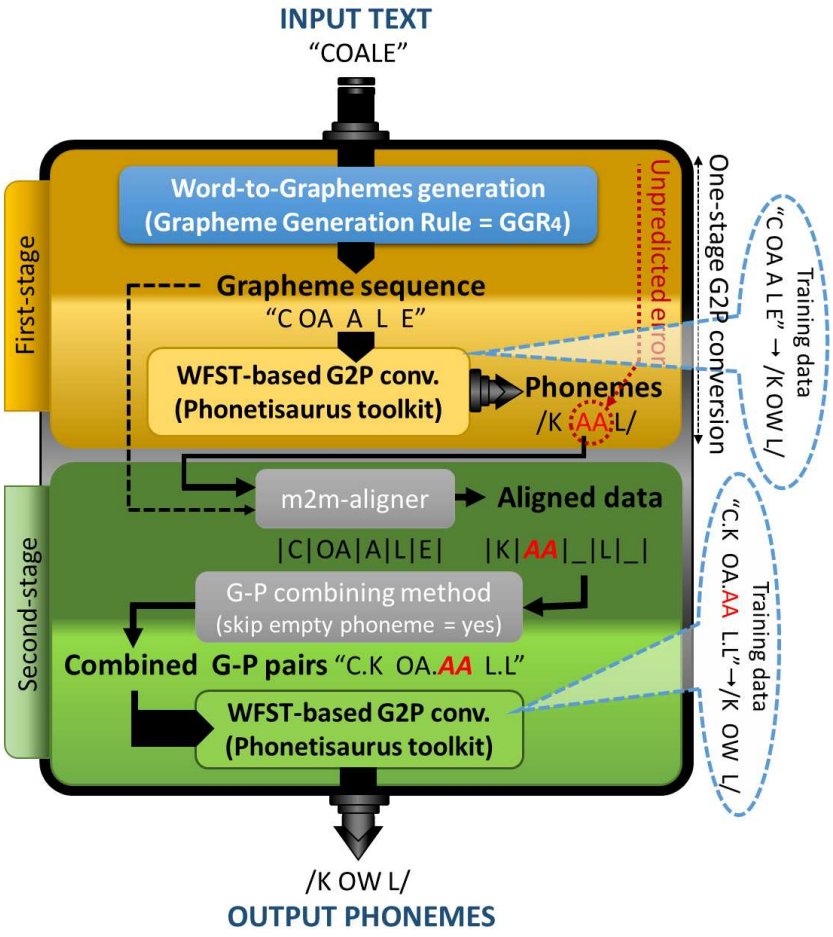


FIGURE 4.1: Architecture of the novel two-stage model-based G2P conversion using grapheme generation rule. In this example, the rule GGR_4 is used for generating the grapheme sequence of the input word "COALE".

4.3.4 Second-stage model

The second-stage model is built similarly to the first-stage model, with the exception that it combines both the input grapheme and the output phoneme sequences obtained from the first stage and utilizes that combined sequence as a new input to determine the best final output phoneme sequence corresponding to the original input word. In this chapter, that new input sequence is called *"a sequence of combined G-P pairs"* hereafter. As both the graphemic information and the preliminary phonemic information have already been obtained before the final phoneme prediction, some errors occurring at the output level of the first-stage model can be fixed at the second stage. Therefore, our novel two-stage model for G2P conversion seems to provide a better performance.

According to Fig.4.1, this conversion requires two additional sub-modules for utilizing

the grapheme and phoneme sequences of the first-stage model as input for the second-stage model. The first sub-module is created using the m2m-aligner software[19] for aligning the grapheme and phoneme sequences. The second sub-module automatically transforms the aligned data into a new sequence of combined G-P pairs to be used as input for the second stage; we also implemented a default option to ignore all the G-P pairs in which the grapheme is mapped to an empty phoneme (i.e., /_ /).

For the previous example, three aligned sequences such as "$|K|O|L|L| \rightarrow |K|AA|L|\_|$," "$|K|O|L|L|E| \rightarrow |K|AA|L|\_|\_|$," and "$|C|OA|A|L|E| \rightarrow |K|AA|L|\_|\_|$" are generated after the alignment process. After passing all of them through the second sub-module, three sequences of combined G-P pairs are made, which include two unique sequences "K.K O.AA L.L" and another sequence "C.K OA.AA L.L". Hence, only two new training data (e.g., "K.K O.AA L.L"→ /K AA L/ and "C.K OA.AA L.L" → /K OW L/) are created. Finally, the error phoneme /AA/ can be fixed at the second-stage.

4.4 Evaluation

In this section, we first describe the data preparation. Then, we present different proposed test sets including two baseline approaches and sixteen other approaches. The performance metrics are explained after that, which is followed by the experimental results of all the proposed test sets.

4.4.1 Data preparation

The performance of our proposed approach was evaluated against two baseline approaches. We conducted experiments on the American English words-based pronunciation dictionary (CMUdict corpus) used in our previous paper [19], except that each word and its phoneme sequence used in this study were unaligned (i.e. absence of the empty grapheme '_' and empty phoneme /_ /). Thus, the training dataset contained a total of 100,713 IV words, while the testing dataset contained 11,188 OOV words. Although we used the same CMUdict corpus as [21, 26], the selected words in our datasets were different from those used in [21, 26]. The dataset preparation is fully described in our previous papers [19, 58].

After the data analysis, the grapheme "X" is sometimes mapped to three phonemes /EH K S/ (e.g., "VISX" → /V IH S EH K S/). To this end, we replaced the connected phonemes /K S/ and /K SH/ with two other phonemes /X/ and /XH/, respectively, for words where "X" produces /K S/ and /K SH/.

[19] m2m-aligner: https://code.google.com/p/m2m-aligner/

TABLE 4.2: Configurations of the eighteen proposed test sets.

Proposed test set	Configuration		
	G-P mapping	/K S/ →/X/ /K SH/→/XH/	Grapheme Generation Rule (GGR_r)
Baseline 1	2-2	No	GGR_1
Baseline 1-0	1-2	No	GGR_1
Approach 1	2-2	Yes	GGR_1
Approach 1-0	1-2	Yes	GGR_1
Approach 2	1-2	No	GGR_2
Approach 3	1-2	No	GGR_3
Approach 4	1-2	No	GGR_5
Approach 4-1	1-2	No	GGR_4
Approach 5	1-2	Yes	GGR_5
Approach 5-1	1-2	Yes	GGR_4
Approach 6	1-2	No	GGR_6
Approach 7	1-2	No	GGR_7
Approach 8	1-2	No	GGR_9
Approach 8-1	1-2	No	GGR_8
Approach 9	1-2	Yes	GGR_9
Approach 9-1	1-2	Yes	GGR_8
Approach 10	1-2	No	GGR_{10}
Approach 11	1-2	No	GGR_{11}

4.4.2 Proposed test sets

In this research, we designed and separately utilized eighteen different test sets, as listed in Table 4.2. According to [72], the WFST-based approach proved to outperform other well-established approaches such as Sequitur [21], direcTL+ [25], therefore we chose only the WFST-based approach to represent our baseline approach. As a result, we first propose two baseline approaches (i.e. Baseline1 and Baseline1-0) using GRR_1, which refers to the original WFST-based approach [26].

Next, two similar approaches (Approach1 and Approach1-0) were designed with respect to both baseline approaches, with the exception that they were evaluated using the datasets where the connecting phonemes /K S/ and /K SH/ were manually replaced by /X/ and /XH/, respectively.

In order to show the effect of our proposed grapheme generation rules on the performance of the G2P conversion, especially on the word accuracy of the OOV dataset, we designed the remaining approaches (as listed in Table 4.2) by assigning each of them different rules and configurations.

In the Phonetisaurus toolkit, the relationship between graphemes and phonemes can be many-to-many, but the best results were obtained when it was set to (1-2) or (2-2).

Otherwise, whenever new grapheme generation rules (except for GRR_1) were applied, our results showed that the relationship (1-2) provided the best results. Therefore, in Table 4.2, we show only the approaches where the relationship (1-2) was used.

4.4.3 Experimental results

The approaches listed in Table 4.2 used the CMUdict corpus to evaluate the model's performance. Since the selected words in both training and testing datasets were different from those used in [21, 26], the accuracy of the baseline approaches presented in this chapter was lower than that shown in both previously mentioned papers. In terms of word accuracy (WAcc) of the OOV dataset, Fig.4.2 and Fig.4.3 indicate that most of the approaches using rules related to the vowel graphemes (i.e. Approach3, Approach4, Approach4-1, Approach5 and Approach5-1) provided better performance than those using rules related to the consonant graphemes (i.e. Approach2, Approach6, Approach7, Approach8, Approach8-1, Approach9, Approach9-1, Approach10 and Approach11); they also provided a slightly higher word accuracy than both baseline approaches at the first stage; however, there was no improvement between the one-stage and two-stage architecture. Conversely, in terms of the WAcc of the IV dataset, all approaches provided almost the same results ($98.19\% \sim 98.39\%$) when built as a one-stage model, but they improved when implemented as a two-stage model.

Among the proposed approaches that use rules related to the vowel graphemes, Approach2, Approach6 and Approach11 provided lower word accuracy than the others, even including both baselines, so we excluded both of them from the next analysis process. Appraoch3 was also eliminated, because its word accuracy was lower than that of the other approaches, especially Approach4; moreover, GGR_3 appeared less effective than the rule used in Approach4. The other approaches, such as Approach7, Approach8, Approach8-1, Approach9, Approach9-1, and Approach10, which use rules focusing on the consonant graphemes rather than the vowel graphemes, were also eliminated because they provided much poorer accuracy compared to the other approaches.

Problems in spelling English words mostly occur when a word has many vowels. Therefore, in order to thoroughly analyze the experimental results, the words in both the training and the testing dataset were classified into six different groups (v_1, v_2, v_3, v_4, v_5, and v_6) depending on the total number of vowels found in each word. The group of words without vowels (v_0) was merged with group v_1, while group v_6 included all the remaining groups (v_7, v_8, etc.). The IV and OOV data at the bottom part of Fig.4.4 show that v_2 was the largest group, while v_6 was the smallest.

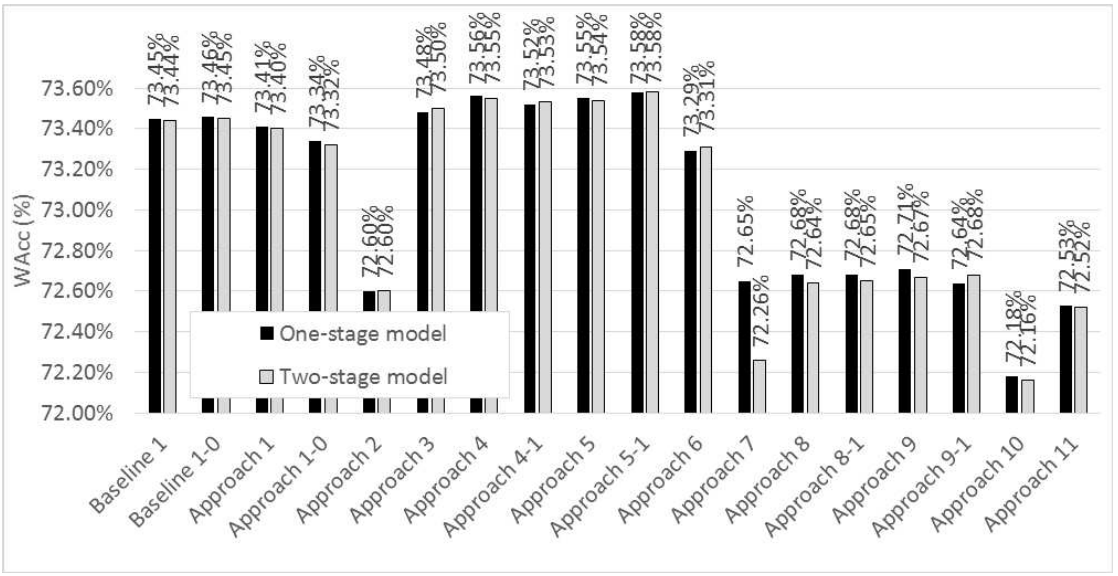


FIGURE 4.2: WAcc of different proposed test sets measured based on OOV words.

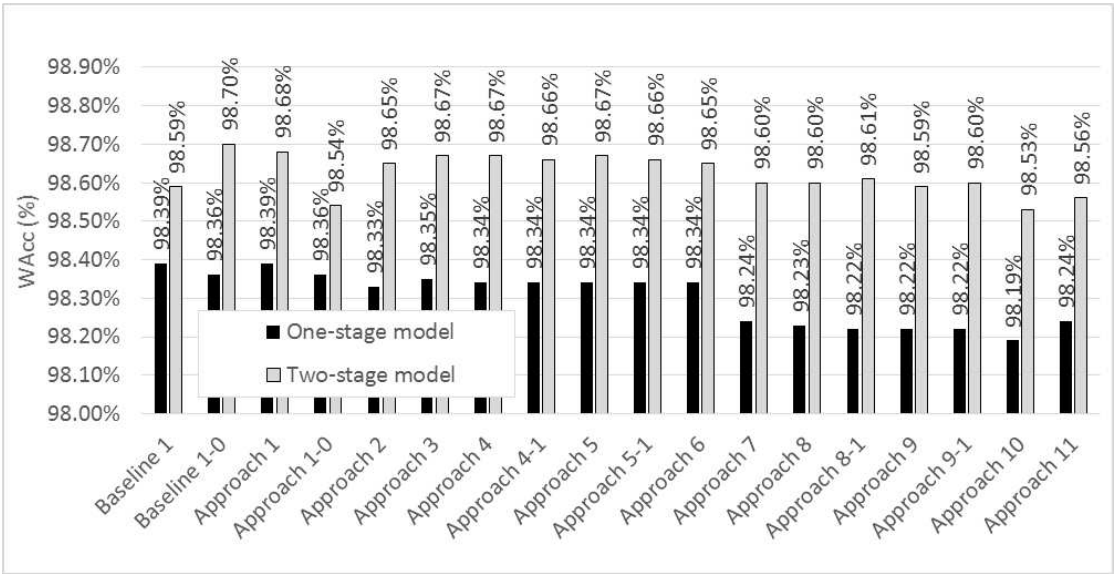


FIGURE 4.3: WAcc of different proposed test sets measured based on IV words.

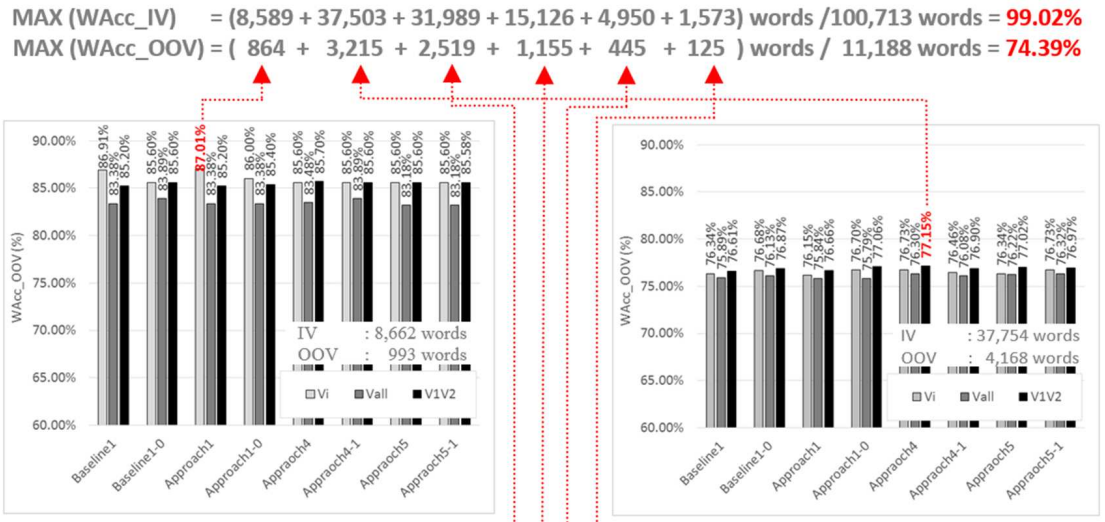


Figure 4.4.1 WAcc of group V_1

Figure 4.4.2 WAcc of group V_2

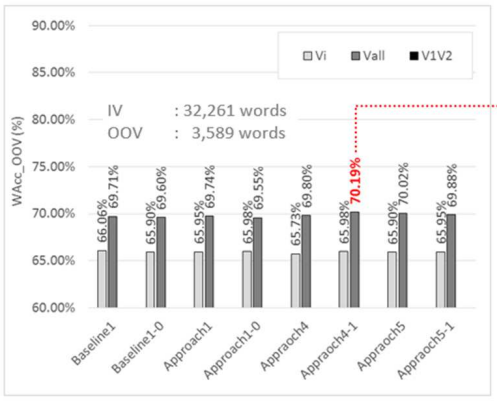


Figure 4.4.3 WAcc of group V_3

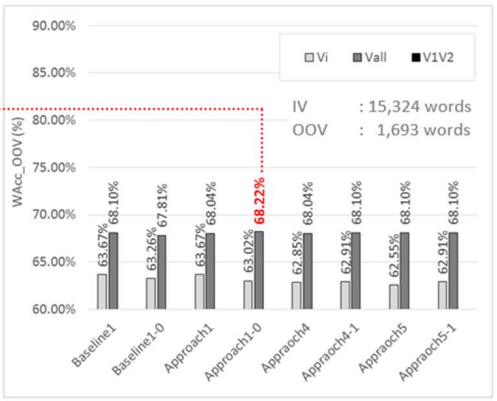


Figure 4.4.4 WAcc of group V_4

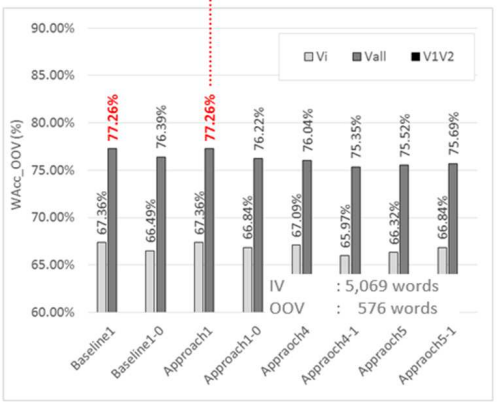


Figure 4.4.5 WAcc of group V_5

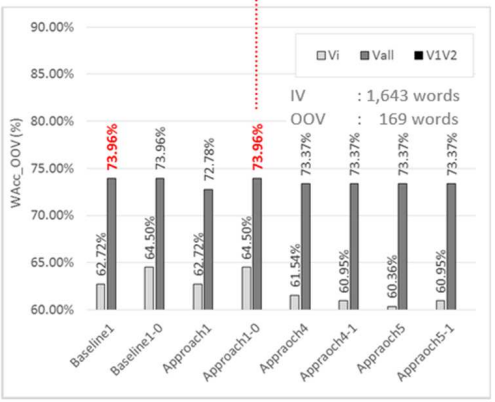


Figure 4.4.6 WAcc of group V_6

FIGURE 4.4: Results of the WAcc obtained from the two-stage model-based G2P conversion and separately measured based on different groups of OOV datasets.

To understand the effects of a different number of vowels in a word and the effects of using different sizes of datasets in the training process, we conducted two different evaluations. First, we trained and evaluated each group of datasets ($v_i = v_1, v_2, ..., v_6$) separately. Second, we used the pre-trained model in Table 4.2 ($v_{all} = v_1 + ... + v_6$) to evaluate each group of datasets (v_i) one by one. The evaluation results given by the different approaches are depicted in Fig.4.4. It shows that the highest values of WAcc for groups v_1 and v_2 (i.e., 87.01% and 76.73%) were obtained using the v_i trained model, while those for the remaining groups were obtained using the Vall trained model. This demonstrates that the pronunciation rules in words with zero, one and two vowels are more consistent than those in words with more vowels. In addition, in the largest group v_2, only 10% of the words consisted of *vvc* syllabels. Based on these facts, we conducted another experiment, where we trained the model using a combined v_1v_2 training dataset (i.e., v_1+v_2) and then evaluated each group v_1 and v_2 separately. As a result, the WAcc of v_2 increased from 76.73% to 77.15%.

We also conducted some experiments in which we kept the G-P pairs with the grapheme mapped to the empty phoneme (e.g., "A._" or "E._" as shown in Fig.4.1), however we did not report those results in this chapter because there was not much difference between the absence and presence of the empty phoneme in the G-P combining method.

4.5 Discussion

The experimental results in Fig.4.2, Fig.4.3 and Fig.4.4 clearly show that our newly proposed rules ($GGR_3, GGR_4,$ and GGR_5) were more effective than the rules representing unigram and bigram models (GGR_1 and GGR_2) since they could help improve the model's performance. However, the results given by Approach6 allow us to assume that the strongest rule, such as in this case GGR_6, does not always lead to the highest performance because it increases the complexity of the training datasets. In addition, the rules that are designed to enable extra detail for the consonant graphemes (i.e., $GGR_7, ..., GGR_{11}$) were not helpful in tackling the problem concerning G2P conversion at all and also degraded the model's performance.

In the one-stage model-based G2P conversion, even though the most effective rules were applied, the WAcc of the IV datasets was very difficult to improve, since it was already very high (for Baseline1, WAcc= 98.39%). However, it could still be improved by adding the second stage. As a result, the two-stage model-based G2P conversion appears to keep almost the same WAcc for the OOV datasets and boosts the WAcc of the IV dataset (i.e., $+0.2\% \sim +0.3\%$ in WAcc which is equal to 200∼300 words difference). Therefore, we believe that our proposed approach also can improve the WAcc of the OOV dataset if we

select the OOV words carefully, as other researchers have done [30, 31, 106]. According to an extra experiment, the newly prepared training and testing datasets (which consist of 100,564 and 11,125 words, respectively) selected only words with grapheme-phoneme pairs appearing at least four times in both datasets. The newly obtained results based on the one-stage architecture prove that our proposed approach using GGR_5 (Approach4) outperformed the baseline approach (Baseline 1-0) ($p < 0.05$), while obtaining 73.89% and 73.54% as WAcc of the OOV dataset using Approach4 and Baseline 1-0, respectively.

Fig.4.4 shows that the highest accuracy for each group of OOV datasets ($v_1 ... v_6$) was obtained using different approaches, which means that it appears to be very difficult to use only one approach to solve all the problems associated with G2P conversion. Therefore, this experiment demonstrates that at least five different approaches are required to reach the maximum value of WAcc related to the OOV datasets. After selecting only the trained models providing a maximum value of WAcc for each group of OOV datasets, we obtained 74.39% and 99.02% as the WAcc of the OOV and IV datasets, respectively. These results show that, if we are able to correctly pick the best output phonemes from several results given by different models, then this combined technique could outperform the baseline approaches (i.e., 0.94% = 108 words difference for the OOV dataset and 0.63% = 634 words difference for the IV datasets).

4.6 Summary

It has been shown in this chapter that using new grapheme generation rules that are designed to enable extra detail for vowel graphemes can improve the performance of G2P conversion. The new phoneme prediction method allows the second-stage model to learn the pronunciation rules more easily than the first-stage model because both the grapheme sequences and the preliminary phoneme sequences have already been identified at the input level. Moreover, we have shown that using a single-stage approach is not sufficient to deal with all the problems associated with G2P conversion, because each approach is designed using different technique to address different challenges and therefore, using various approaches proves very helpful in solving different specific problems.

Chapter 5

Phoneme Transition Network-based G2P Conversion

Contents

5.1 Introduction

Often, there is no strict correspondence between letters and phonemes in spoken words, and this is especially true for an orthographically irregular language like English [11]. Thus, researchers have proposed various data-driven methods using many-to-many mapping techniques between graphemes and phonemes. Methods have been proposed based on hidden Markov models (HMMs) [16, 17], support vector machines (SVMs) [18], artificial neural network (ANNs) [19], joint-sequences [21], margin-infused relaxed algorithms (MIRAs) [23, 24, 25], a weighted finite-state transducer (WFST) [27], conditional random fields (CRF) [28, 29, 30], hidden conditional random fields (HCRF) [31, 32], an adaptive regularization of weight vectors (AROW) [33], a narrow adaptive regularization of weight vectors (NAROW) [34], and structured soft-margin confidence weighted learning (SSMCW) [35]. Most of these methods, and especially SSMCW-based G2P conversion, are implemented in the Slearp toolkit [20] and have demonstrated significantly accurate results. However, each of these methods has been designed using specific techniques that address particular challenges faced by G2P conversion. Therefore, any single approach will not suffice when addressing all of the problems encountered by G2P conversion [60]. Considering this, a combination of various approaches using different methods is a reasonable strategy for treating these problems in a flexible manner. For example, word or phoneme transition network-based methods have been successfully used in various research domains such as automatic speech recognition [107], speech search [57, 108, 109, 110], speech translation [111], and speech summarization [112].

Combining various methods can both lend flexibility to the conversion and improve its predictive performance. Thus, in this chapter we present a Phoneme Transition Network (PTN)-based architecture for G2P conversion. Basically, our proposed PTN-based method first converts a target word into multiple phoneme strings using several data-driven methods. Then, it aligns the obtained results—the phoneme-sequence hypotheses—using a dynamic-programming (DP) algorithm, combining them into a confusion network (hereafter referred to as the "PTN"), and determining the best phoneme from each PTN bin—a block of phonemes/transitions between two nodes in the PTN—to represent the final output. The best phoneme selection in this study is based on a voting strategy according to the frequency and maximum confidence score of the occurrences implemented in the Recognizer Output Voting Error Reduction (ROVER) system [113].

Selecting the set of methods used by the proposed architecture is a crucial task. If accurate methods are combined with inaccurate methods, this can considerably degrade the

[20] Slearp: http://osdn.jp/projects/slearp/

performance of the entire PTN-based G2P conversion model. For example, Schlippe et al. merged five phoneme-sequence hypotheses generated from five different methods to enhance the generation of pronunciation in low-resource scenarios [114]. However, they could not demonstrate any significant improvement using this combined approach without the addition of web-derived pronunciation dictionaries. Even so, this improvement deteriorated as the size of the training data increased, especially for a difficult language like English. On the other hand, according to our previous research [115], when the number of phoneme-sequence hypotheses generated from inaccurate models was more than the number of those generated from accurate models, it was difficult to maintain and improve the performance of the PTN-based model.

In order to mitigate this risk, we selected a minimum number of methods.[21] We also present a novel use for right-to-left (reversed) grapheme-phoneme (g-p) sequences and grapheme generation rules (GGRs) [60]. In this study, both techniques are especially helpful for extending the feasibility and improving the performance of PTN-based G2P conversion, because they increase the number of phoneme-sequence hypotheses without increasing the number of methods used. By reversing the conventional (left-to-right reading direction) g-p sequence, we can provide context information that differs from conventional sequences during the alignment. This allows each single method to train an additional model, thus producing an additional phoneme-sequence hypothesis. In addition, applying various GGRs[22] to the words (that satisfy the rules) in the source corpus will also generate additional grapheme-sequences and more training samples. This increases the size of training data, enabling a single trained model to produce more than one phoneme-sequence hypothesis. Therefore, this chapter proposes two different versions of the PTN-based architecture for G2P conversion. As a result of the reversed g-p sequences, the first architecture uses only three different methods, based on MIRA [25], WFST [27], and SSMCW [35], to train six separated source models in order to generate six phoneme-sequence hypotheses. To reduce the number of methods as well as the number of trained models, we use only a single GGR rule for the second architecture. Consequently, this architecture requires only four models based only on a single method (viz., an SSMCW-based method) to generate the same number of hypotheses.

We evaluated our proposed models against the three baseline methods mentioned in the previous paragraph using multiple datasets and the K-fold cross-validation technique. The results indicate an improvement in both phoneme and word accuracy with respect to OOV words.

[21] In previous research, a method/approach has been used to train a single model only, so the terms "method/approach" and "model" might have a similar meaning. Otherwise, here, we differentiate between them because a single selected method in this chapter can be used to train more than one model.

[22] In English, the interaction between vowels in a word strongly affects its spelling. Thus, GGRs were originally proposed to add extra-sensitive information to each vowel-grapheme appearing in a word.

The remainder of this chapter is organized as follows. In Section 5.2, we describe the three data-driven methods for G2P conversion selected for this study. We then present the PTN-based G2P conversion and its compact version in Sections 5.3 and 5.4, respectively. The evaluation results and discussion are presented in Sections 5.5 and 5.6, respectively. The conclusion is given in Section 5.7.

5.2 Different data-driven methods for G2P conversion

Many data-driven approaches to G2P conversion have been proposed, but the popular joint-sequence or n-gram model-based methods for G2P conversion have been proven to be the most powerful techniques for dealing with OOV words. Because our proposed approach requires the combination of at least three methods, we selected the three most powerful statistical-based methods that differently encode the n-gram model.

5.2.1 MIRA-based method for G2P conversion (DirecTL+)

The best-known joint n-gram model-based method for G2P conversion was first proposed in 2008 by Bisani and Ney [21], and it was implemented as a generative system available in the Sequitur toolkit.[23] In this system, the model is trained using the expectation-maximization algorithm, and the phoneme sequence corresponding to a given word $\varphi(g)$ is predicted through a Bayes' decision rule as follows:

$$\varphi(g) = argmax_{\varphi'} P(g, \varphi') \tag{5.1}$$

Here, g represents a given grapheme sequence, where φ' is the most likely pronunciation of the grapheme sequence g.

Soon after, Jiampojamarn et al. represented the joint n-grams model for G2P conversion as an online discriminative sequence-prediction model, which used a many-to-many alignment between grapheme and phoneme sequences and a feature vector consisting of n-grams context features, HMM-like transition features, and linear-chain features [23]. For each training iteration, the feature weight vector was updated using the MIRA algorithm; this system is called DirecTL. The updated version of DirecTL is called the DirecTL+ toolkit,[24] implemented in 2010, in which the joint n-gram features were integrated [24].

[23] http://www-i6.informatik.rwth-aachen.de/web/Software/g2p.html
[24] https://code.google.com/p/directl-p/

5.2.2 Rapid WFST-based G2P conversion (Phonetisaurus)

A WFST-based method for G2P conversion proposed by Novak et al. [27] has been implemented to develop a rapid and high-quality joint-sequence model-based G2P conversion. First, the training words and their phoneme sequences are provided, and these are aligned using an expectation-maximization training procedure based on the many-to-many aligning technique [21]. The joint-sequence corpus is given as an input for n-gram counting (in which the order or length of the n-grams to count is provided), and then a standard joint n-gram model is trained using the MITLM tookit[25] or the OpenGrm NGram library,[26] and smoothed by Kneser-Ney discounting with interpolation. Then, the trained n-gram model is converted to a WFST-based model, which predicts the phoneme sequences of unknown words using the following decoding function:

$$Phseq_{best} = shortestPath(Project_o(W \circ_o M)) \tag{5.2}$$

where "$Phseq_{best}$" refers to the most likely phoneme sequence given the input word "W" under the FSA representation and the n-gram model "M" encoded as FST, "$\circ_o$" refers to the weighted composition, "$Project_o(.)$" is a projection onto the output symbols, and "$shortestPath(.)$" indicates the shortest-path algorithm.

5.2.3 SSMCW-based G2P conversion (Slearp)

Structured online discriminative learning methods, such as structured AROW [33] and NAROW [34], have been successful at improving performance in G2P conversion. Recently, an SSMCW-based method [35] has been proposed for extending multi-class confidence-weighted learning to structured learning, which softens the marginal errors for hypothesis and update parameters using the N-best hypotheses simultaneously and interdependently for robustness against over-fitting.

The general formulation of a G2P conversion model using a structured learning method is as follows:

$$\hat{y} = argmax_y \omega^T \Phi(x, y) \tag{5.3}$$

where the parameters x and y represent a given grapheme sequence and its corresponding phoneme sequence, respectively, ω indicates the weight vector for the classifier, and $\Phi(x, y)$ is a feature vector that consists of the frequencies of joint n-gram features on x and y. The predicted phoneme sequence $\hat{y}$ is obtained using a dynamic-programming

[25] https://code.google.com/p/mitlm/
[26] http://www.openfst.org/twiki/bin/view/GRM/NGramLibrary

algorithm. For a detailed discussion of how the parameters in Eq.(5.3) are determined, please refer to [35].

5.3 PTN-based architecture for G2P conversion

In this section, we first introduce a novel use of reversed g-p sequences and explain how PTN sequences are generated from multiple phoneme sequences. Then, we describe how to determine the best output phoneme sequence from the PTN sequence using voting techniques.

5.3.1 Reversed g-p sequences

To predict a phoneme sequence corresponding to an input grapheme sequence, most existing approaches use an n-gram model to calculate the likelihood probability that a phoneme (sequence) accurately corresponds to a particular grapheme (sequence) [16, 21, 24, 27, 33, 34, 35]. This means that only the context from left to right is seen by the model. Thus, the trained model can only learn or cover the relationship between graphemes and phonemes in a single direction.

According to [116], Sutskever et al. reversed the order of input words in all source sentences, but not in the target sentences, and this was done in order to train a machine-translation model using a multi-layered Long Short-Term Memory Recurrent Neural Network (LSTM-RNN). This cross-mapping technique is possible owing to Connectionist Temporal Classification (CTC) [117], which allows the RNNs to be trained without requiring any prior alignment between the source and target sequences. Sutskever et al. demonstrated that this reversed-word model (slightly) outperformed models based on conventional word sequences.

However, this cross-mapping technique is inadequate for statistical-based methods where a prior alignment between input and output sequences is required [16, 21, 24, 27, 33, 35].[27] Therefore, in this chapter we introduce a new way to use the reversing technique for G2P conversion, such that it avoids alignment problems. Rather than reversing only the input grapheme sequence, we reverse both the input grapheme and the output phoneme sequences, as demonstrated in the following example:

- Conventional g-p sequences: "LURIE"→/L UH R IY/
- Reversed g-p sequences: "EIRUL"→/IY R UH L/

[27] We also conducted tests for G2P conversion, but the results were completely unsuitable, because the grapheme in a left-to-right direction must be aligned to the phoneme in the reversed direction.

5.3.2 PTN generation using multiple phoneme sequences

Over the last few years, it has proven considerably difficult to improve the performance of a G2P conversion model for OOV words, because each method or approach is uniquely designed using different techniques to address particular challenges. It is seemingly impossible to utilize any single method to deal comprehensively with the host of problems encountered by G2P conversion [60]. Therefore, we designed a PTN-based architecture for G2P conversion that allows many different methods to be applied together, in order to deal broadly with the various problems.

The number of methods used by the PTN-based G2P conversion model, as well as the methods themselves, must be carefully selected, owing to the risk of combining accurate methods with inaccurate ones such that the performance of the entire model is degraded. In order to minimize this risk, only a few accurate methods should be used. By contrast, combining only a minimum number of phoneme-sequence hypotheses will not improve the PTN-based G2P conversion [115].

Therefore, in this study, we propose a novel PTN-based architecture using the three most accurate methods for G2P conversion: the SSMCW-based method (available in the Slearp toolkit), the WFST-based method (available in the Phonetisaurus toolkit), and the MIRA-based method (available in the DirecTL+ toolkit). As depicted in Fig.5.1, by using the conventional g-p sequences as training data, we can generate three phoneme-sequence hypotheses from three source models: Slearp, Phonetisaurus (Phon.), and DirecTL+. Furthermore, the reversed g-p sequences allow these three methods to produce three additional models: Slearp.reverse, Phonetisaurus.reverse (Phon.reverse), and DirecTL+.reverse. In total, six phoneme-sequence hypotheses are generated from six models implemented using only three methods.

The ROVER system [113] allows us to align these phoneme sequences using a DP algorithm, and to merge them together in a single confusion network (or PTN), as shown in Fig.5.1. In this context, given the presence of insertion or deletion problems during the alignment, a null phoneme /@/ is used by the PTN to represent a null transition.

5.3.3 Determining the best output phoneme

Theoretically, many phoneme sequences can be generated from a PTN, but only a single sequence is needed to represent the best output of the model. In order to determine the best output sequence, we adopted a voting strategy, according to the frequency and maximum confidence score of the occurrences. This voting scheme is provided in the ROVER system [113]. The phoneme-selection function for each PTN bin is based on

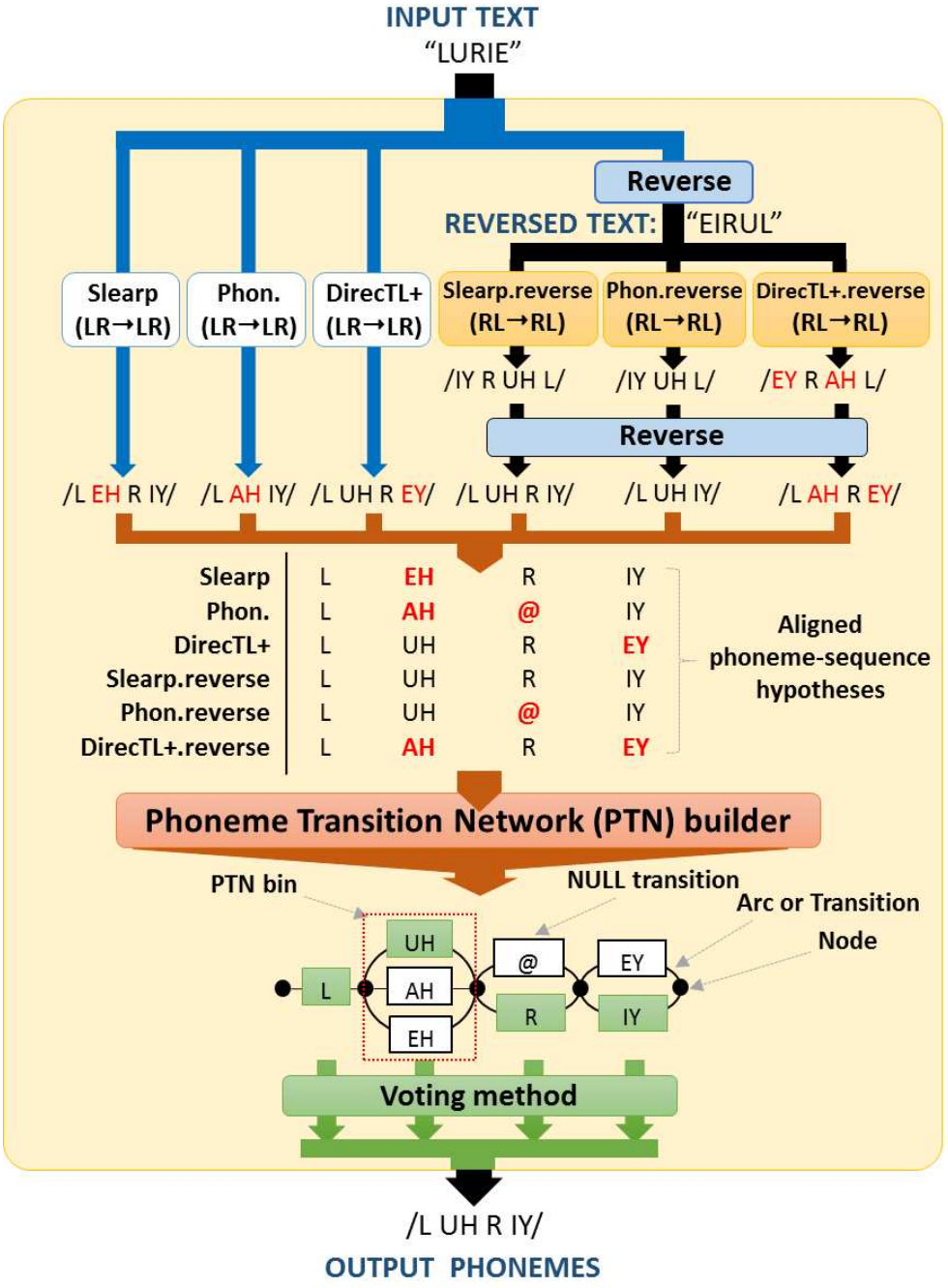


FIGURE 5.1: Architecture for the first proposed PTN-based G2P conversion using six models based on three different methods. (LR→LR) and (RL→RL) represent the models trained using the conventional and reversed g-p sequences, respectively.

TABLE 5.1: Examples of GGR_r rules.

GGR_0	g_i	$\Rightarrow$	g_i	(like unigram = default)
	Ex: "OKEECHOBEE"	$\Rightarrow$	O K E E C H O B E E	
GGR_1	g_i	$\Rightarrow$	$g_i g_{i+1}$	(like bigram)
	Ex: "OKEECHOBEE"	$\Rightarrow$	OK KE EE EC CH HO OB BE EE E_	
GGR_2	If $(n > 1)$ $v_1 ... v_n c_{n+1}$	$\Rightarrow$	$v_1 v_2$ $v_2 v_3$ ... $v_{n-1} v_n$ v_n c_{n+1}	
	If $(n = 1)$ g_i	$\Rightarrow$	g_i	
	Ex: "OKEECHOBEE"	$\Rightarrow$	O K **EE** E C H O B **EE** E	

Here, $g_i = \{c_i, v_i\}$: grapheme at index i;
c_i , v_i : consonant and vowel graphemes at index i;
n : number of connecting vowels in a given word.

the following scoring formula:

$$score(ph) = \alpha(N(ph,i)/n) + (1-\alpha)C(ph,i) \tag{5.4}$$

$$C(ph,i) = MAX(conf_1(ph,i), conf_2(ph,i), ..., conf_n(ph,i)) \tag{5.5}$$

where $N(ph,i)$ is the number of occurrences of the phoneme ph in the i^{th} PTN bin, and n denotes the number of phoneme-sequence hypotheses. Furthermore, $C(ph,i)$ represents the confidence score for the phoneme ph in the i^{th} PTN bin, where $conf_1(ph,i), ..., conf_n(ph,i)$ is the set of confidence scores for ph in the i^{th} PTN bin that correspond to the various sequence hypotheses. The real value $\alpha = [0...1]$ represents a trade-off between the phoneme frequency and the confidence score in Eq.(5.5).

5.4 Reducing the number of required source models

Even if the reversed g-p sequences can make a complementary model that can generate an additional phoneme-sequence hypothesis for each source method, the risk from combining different methods nevertheless remains. Hence, we introduce a novel use of grapheme generation rules (GGRs) [60] to minimize this risk. This allows us to use only a single method for implementing a PTN-based G2P conversion model.

5.4.1 Grapheme generation rules (GGRs)

Textual information does not supply a sufficient amount of information relating to the phonological interaction [104]. In orthographically complex languages such as English, the interaction between vowels in a word significantly affects the spelling. Hence, a technique for generating new grapheme sequences from the same input text (known as

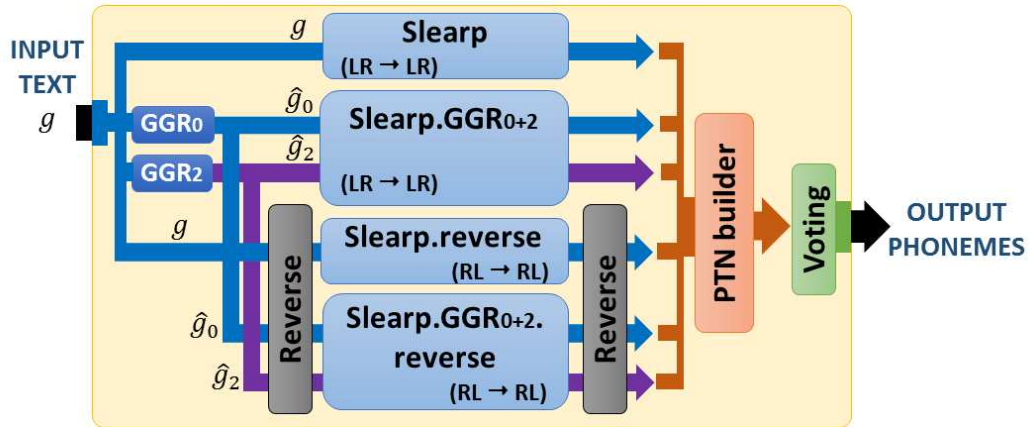


FIGURE 5.2: Architecture for the second proposed PTN-based G2P conversion using four models based on only a single method from the Slearp toolkit.

GGRs) has been proposed for adding extra-sensitive information to each vowel-grapheme appearing in a word [60]. Suppose that a grapheme sequence $g = g_1 g_2 ... g_n$ is provided as an input. The new grapheme sequence $\hat{g}_r = \bar{g}_1 \sqcup \bar{g}_2 \sqcup ... \sqcup \bar{g}_n$, in which an empty space is used as a separator, can be generated with respect to a rule GGR_r, formulated as follows:

$$\hat{g}_r = GGR_r(g) \tag{5.6}$$

A list of few rules, which is selected from [60] and designed to tackle the connecting vowels in the English language, is provided in Table 5.1. In this study, we selected only the rules GGR_0 and GGR_2 for our first-time experiments because we wanted to investigate the difference between the baseline rule GGR_0 and the best rule GGR_2 from [60] when used within the PTN-based G2P conversion. The rule GGR_0 is equivalent to the conventional grapheme sequence (where the space is ignored), but GGR_2 can distinguish the separated vowel v in the cvc pattern from the connecting vowels $v_1, v_2, ..., v_{n-1}$ in the $v_1 v_2 ... v_n$ pattern.

5.4.2 PTN-based G2P conversion using only one method

According to Fig.5.1, after using the reversed g-p sequences, only three different methods are required for generating six phoneme-sequence hypotheses used in the PTN-based G2P conversion. However, the source/trained models remain the same six models. Hence, the integration of GGRs into the source models is especially helpful.

Rather than using only the original word-pronunciation pairs from the source corpus, we applied several GGRs to all the words, in order to generate additional g-p sequences. These were then added to the dataset; the redundant g-p sequences were omitted. According to the example in Table 5.2, we suppose that a source dataset

TABLE 5.2: Example of a newly generated dataset when various GGR_r rules are applied. Here, the g-p sequences in the source dataset are selected from the CMU-Dict_noisy corpus.

	Grapheme sequence	→	Phoneme sequence	GGR_r
Source (S)	NEWLY	→	N UW L IY	
	CREATIVE	→	K R IY EY T IH V	
	IDEA	→	AY D IY AH	
GGR$_0$(S) ($\hat{S}_0$)	N E W L Y	→	N UW L IY	GGR_0
	C R E A T I V E	→	K R IY EY T IH V	GGR_0
	I D E A	→	AY D IY AH	GGR_0
GGR$_2$(S) ($\hat{S}_2$)	N E W L Y	→	N UW L IY	GGR_2
	C R **EA** A T I V E	→	K R IY EY T IH V	GGR_2
	I D **EA** A	→	AY D IY AH	GGR_2
$\hat{S}_0 \cup \hat{S}_2$ ($\hat{S}$)	N E W L Y	→	N UW L IY	GGR_0 or GGR_2
	C R E A T I V E	→	K R IY EY T IH V	GGR_0
	C R **EA** A T I V E	→	K R IY EY T IH V	GGR_2 (+)
	I D E A	→	AY D IY AH	GGR_0
	I D **EA** A	→	AY D IY AH	GGR_2 (+)

(i.e., $S = \{(g,p)_1, (g,p)_2, ..., (g,p)_N\} = \bigcup_{k=1}^{N}(g,p)_k$) consists of N pairs of g-p sequences. Then, a set of R rules is applied, and the newly generated dataset $\hat{S}$ is formulated as follows:

$$\hat{S} = \bigcup_{r=1}^{R} \hat{S}_r = \bigcup_{r=1}^{R} GGR_r(S) = \bigcup_{r=1}^{R}\bigcup_{k=1}^{N}(GGR_r(g),p)_k = \bigcup_{r=1}^{R}\bigcup_{k=1}^{N}(\hat{g}_r,p)_k \tag{5.7}$$

As a result, for each input word (refers to the conventional grapheme sequence g), Fig.5.2 shows that it is possible to generate more than one phoneme sequence from a trained model in which the newly generated dataset $\hat{S}$ is used (e.g., Slearp.GGR$_{0+2}$), given the different representations of its grapheme sequence (e.g., the generated grapheme sequences $\hat{g}_0 = GGR_0(g)$ and $\hat{g}_2 = GGR_2(g)$ seen in Table 5.2). By using both reversed g-p sequences and various GGRs, the number of generated hypotheses Nb_{hyps} can be calculated using the following formula:

$$Nb_{hyps} = \begin{cases} 2 * Nb_{GGRs}, & \text{if the reversed g-p sequences are used} \\ Nb_{GGRs}, & \text{otherwise} \end{cases} \tag{5.8}$$

where Nb_{GGRs} indicates the number of applied rules.

The novel use of GGRs in G2P conversion allows us to use only one method to train one or several models combined at the PTN level. In this study, we compared the performance among the models using GGRs with those using conventional and reversed g-p

TABLE 5.3: Datasets or corpora used in the experiments.

Dataset	Vocabulary size (words)			
	Train	Dev.	Test	K-fold
NETalk (English)	17,508	1,000	1,500	10
Brulex (French)	23,955	1,000	2,500	10
CMUdict (English)	95,286	6,000	11,000	8
CMUdict_noisy (English)	107,438	5,939	11,998	1
CMUdict_noisy_GGR$_{0+2}$ ($\hat{S}$)	*130,533*	*7,787*	*15,372*	1

sequences. Therefore, as seen in Fig.5.2, the second proposed PTN-based architecture for G2P conversion combines six hypotheses generated from four models implemented using only a single method (i.e., the most accurate SSMCW-based method for G2P conversion available in the Slearp toolkit). The Slearp and Slearp.reverse models are trained using the original dataset S, and thus producing only two phoneme-sequence hypotheses. The Slearp.GGR$_{0+2}$ and Slearp.GGR$_{0+2}$.reverse models are trained using the newly generated dataset $\hat{S}$, and thus possibly generating four phoneme-sequence hypotheses. Although the input grapheme sequences g and $\hat{g}_0$ are equivalent, two different phoneme-sequence hypotheses might be produced owing to the different source models.

5.5 Evaluation

In this section, we describe the data-preparation process and the experimental setup. Subsequently, we report the experimental results.

5.5.1 Data preparation

The performance of our two proposed approaches was evaluated relative the baseline models discussed in Section 5.2. We conducted experiments using four different pronunciation dictionaries (three in English and one in French), as listed in Table 5.3. The NETtalk, Brulex and CMUdict datasets were obtained from the Pascal Letter-to-Phoneme Conversion Challenge website[9]. A noisy CMUdict dataset (CMUdict_noisy) containing words with multiple pronunciations (i.e., heteronyms) is available in the Phonetisaurus package. In this study, we used the NETtalk corpus to tune the parameters for each method and the ROVER system.

We subdivided each corpus into training, development, and testing datasets. The NETtalk, Brulex, and CMUdict datasets each originally consisted of ten separated folds.

[9]http://pascallin.ecs.soton.ac.uk /Challenges/PRONALSYL/Datasets

Thus, for each trial of cross-validation, one fold was used as the testing data, some data in another fold was randomly selected for the development data, and the eight remaining folds, along with the leftover data from the fold used for the development data, were extracted and combined for use as training data. By contrast, the source of the CMUdict_noisy dataset originally consisted of only two parts (training and testing datasets). Thus, development data was randomly extracted from the training dataset. In order to conduct a fair evaluation, when the same word appeared multiple times with different phoneme sequences in the development or testing dataset, we retained only a single pair.

Owing to the fact that the GGRs in this paper were designed exclusively for English words and the CMUdict_noisy corpus were used in many previous studies [21, 24, 27], we used only this corpus to evaluate our second PTN-based G2P conversion (see Section 5.4). Eq.(5.7) was applied to increase the size of the training, development, and testing datasets, after GGR_0 and GGR_2 were applied, the details for which are provided in Tables 5.2 and 5.3. Here, GGR_0 was used to convert the format of the original grapheme sequence by adding a space between two connected graphemes.

5.5.2 Experimental setup

5.5.2.1 Proposed test sets

In our experiments, we employed the three original models using the conventional g-p sequences as baseline models—viz., Slearp (*1_Slearp* in Tables 5.4 and 5.5), Phonetisaurus (*2_Phon.*), and DIRECTL+ (*3_DIRECTL+*), presented in Section 5.2.

To see the advantages from using the reversed g-p sequences for G2P conversion, we proposed three additional models (*4_Slearp.reverse, 5_Phon.reverse and 6_DIRECTL+.reverse*) in which the reversed g-p sequences were used in place of the conventional sequences.

As listed in Tables 5.4 and 5.5, in order to compare the performance between G2P conversion based on a single model with G2P conversion based on multiple models, we proposed three PTN-based G2P conversion models. In this case, all six separated models mentioned in the previous paragraph (labeled 1, 2, 3, 4, 5 and 6 in the PTN notation) were considered baseline models. For three-model combinations, we proposed PTN(1+2+3) and PTN(4+5+6) for comparing the performance between the PTN-based model with only the conventional g-p sequences and the one with only reversed g-p sequences. PTN(1+...+6) was proposed both to evaluate the performance of the PTN-based model with all six baseline models and also to observe the effect and risk from combining accurate and inaccurate source models.

On the other hand, in the evaluation of our second PTN-based architecture (see Section 5.4), we implemented four baseline models (viz., *1_Slearp, 2_Slearp.GGR$_{0+2}$, 3_Slearp.reverse*, and *4_Slearp.GGR$_{0+2}$.reverse*), as seen in Fig.5.2 and Table 5.6. The first and third models were trained using the training and development datasets from the original corpus, CMUdict_noisy, whereas the second and fourth models were trained using datasets from the newly generated corpus CMUdict_noisy_GGR$_{0+2}$. For each input word, two different representations of a grapheme sequence can be encoded using GGR$_0$ and GGR$_2$. Thus, two phoneme-sequence hypotheses must be generated from each of the models using GGRs (i.e., 2_Slearp.GGR$_{0+2}$ or 4_Slearp.GGR$_{0+2}$.reverse). In our evaluation, we considered these hypotheses as belonging to two separated models. The evaluation results from all the baseline models (A, B, C, D, E and F) in Table 5.6 were obtained using the same test data—i.e., the same input words—but with different graphemic representations. In order to compare the performance between G2P conversion based on a single model with G2P conversed based on a compact PTN, we proposed the same three PTN models with respect to the evaluation of our first architecture.

5.5.2.2 Experiment configurations

According to the results of the preliminary experiments using the NETtalk corpus, the necessary parameters for the three selected methods for G2P conversion were tuned as follows:

- In the DIRECTL+ toolkit, the size of the n-gram context features and joint n-gram features was set to 7 and 3, respectively. Data alignment was based on the mpaligner software [118], and the association between graphemes and phonemes was set to 2-3.
- In the Phonetisaurus toolkit, the number of discounting (*bins*) and the maximum length of n-grams to count (*order*) were set to 3 and 8, respectively.
- In the Slearp toolkit, the size of the n-gram context and chain features was set to 5, while the joint n-gram feature size was set to 7. Pre-alignment was also based on the mpaligner software with m-m association.
- For both Slearp and DIRECTL+, the minimum number of iterations before ending the training process and the maximum number of iterations after a degradation in the performance of the development data were both set to 10. The best iteration was selected based on both phoneme and word accuracy, and this was measured with the development dataset.

TABLE 5.4: Phoneme (PAcc) and word accuracy (WAcc) for all baseline and PTN-based G2P conversion models, using NETtalk corpus. The italicized text indicates the highest accuracy among the baseline models. The text is bold where a PTN provided a better result than all the baselines, and the background is gray when the PTN(1+...+6) outperformed both PTN(1+2+3) and PTN(4+5+6).

	NETtalk	
	PAcc	WAcc
1_Slearp	93.66%	70.99%
2_Phon.	92.89%	68.56%
3_DirecTL+	93.75%	71.31%
4_Slearp.reverse	*93.79%*	*71.93%*
5_Phon.reverse	93.07%	69.15%
6_DirecTL+.reverse	93.65%	70.89%
PTN(1+2+3)	**94.10%**	**72.73%**
PTN(4+5+6)	**94.16%**	**73.14%**
PTN(1+2+3+4+5+6)	**94.23%**	**73.45%**

TABLE 5.5: Phoneme (PAcc) and word accuracy (WAcc) for all baseline and PTN-based G2P conversion models, using other remaining corpora.

	Brulex		CMUdict		CMUdict_noisy	
	PAcc	WAcc	PAcc	WAcc	PAcc	WAcc
1_Slearp	*99.15%*	*95.65%*	93.60%	73.12%	93.83%	73.55%
2_Phon.	98.95%	94.52%	93.25%	72.39%	93.48%	72.71%
3_DirecTL+	98.20%	92.54%	92.61%	70.91%	92.37%	70.11%
4_Slearp.reverse	99.14%	*95.55%*	*93.74%*	*73.91%*	*93.84%*	*73.96%*
5_Phon.reverse	98.93%	94.43%	93.30%	72.53%	93.54%	73.10%
6_DirecTL+.reverse	98.20%	92.55%	92.19%	69.88%	91.91%	68.92%
PTN(1+2+3)	**99.20%**	**95.89%**	93.11%	73.87%	**94.28%**	**75.20%**
PTN(4+5+6)	**99.20%**	**95.82%**	**94.06%**	**74.96%**	**94.23%**	**75.25%**
PTN(1+2+3+4+5+6)	**99.22%**	**95.98%**	**94.13%**	**75.17%**	**94.28%**	**75.30%**

TABLE 5.6: Performance of the compact PTN-based G2P conversion using only the Slearp toolkit, GGRs, and reversed g-p sequences. The bold text and gray background in this table are used in the same manner as Tables 5.4 and 5.5.

	Phoneme-sequence hyp.	CMUdict_Noisy	
Trained model	Model name for evaluation	PAcc	WAcc
1_Slearp	**A-Slearp**	93.83%	73.55%
2_Slearp.GGR_{0+2}	**B-Slearp.GGR$_0$**	93.93%	74.16%
	C-Slearp.GGR$_2$	93.96%	74.21%
3_Slearp.reverse	**D-Slearp.reverse**	93.84%	73.96%
4_Slearp.GGR_{0+2}. reverse	**E-Slearp.GGR$_0$.reverse**	94.08%	74.99%
	F-Slearp.GGR$_2$.reverse	*94.08%*	*75.08%*
	compactPTN(A+B+C)	93.97%	74.28%
	compactPTN(D+E+F)	**94.11%**	**75.09%**
	compactPTN(A+B+C+D+E+F)	**94.29%**	**75.56%**

In order to improve the performance of the most accurate source models for PTN-based G2P conversion, a set of confidence scores in Eq.(5.5) should be assigned according to the ranking of the models in terms of their accuracy. If $\{a, b, c, d, e, f\}$ is a set of scores for our six baseline models, sorted according to accuracy, then each phoneme of the sequence hypothesis generated from the model with the highest accuracy was assigned the highest score, a, and each one from the model with the lowest accuracy was assigned the lowest score, f. Based on our experiments, for both PTN(1+...+6) and compactPTN(A+...+F), the best results were obtained when the values of a, b, c, d, e and f were assigned to 1.0, 0.7, 0.6, 0.5, 0.4 and 0.2, respectively; for the ROVER system, the value of α and the confidence score of NULL phoneme /@/ (noted as $Nconf$) in Eq.(5.4) and Eq.(5.5) should be equal to 0.7 and 0.8, respectively. On the other hand, we used only a set of three scores $\{a, b, c\}$ for PTN(1+2+3), PTN(4+5+6), compactPTN(A+B+C) and compactPTN(D+E+F); in this case, the best results were obtained when the values of a, b and c were assigned to 1.0, 0.7, and 0.6, respectively.

To conduct our experiments, we simultaneously executed multiple programs on a shared server (CentOS 6.6, Intel(R) 12-Core(TM) i7-4930K CPU 3.40 GHz, RAM 64 GB, HDD 630 GB) in our laboratory.

5.5.2.3 Performance metrics

We evaluated the models' performance in terms of phoneme accuracy (PAcc) and word accuracy (WAcc), using the NIST SCLITE scoring toolkit.[10] In this chapter, we report only the results concerning the OOV words in the testing dataset. We also measured the statistical significance (i.e., p-values) using McNemar's test.

5.5.3 Experimental results

All of the evaluation results for the baseline models and the G2P conversion models based on our first (Fig.5.1) and second (Fig.5.2) PTN-based architectures are described hereafter.

According to Tables 5.4 and 5.5, and with the exception of the NETtalk corpus, Slearp generally performed best among the three baselines (i.e., 1_Slearp, 2_Phon. and 3_DIRECTL+) in which the conventional g-p sequences were used. For instance, in terms of the WAcc, Slearp achieved 95.65%, 73.12%, and 73.55% for the Brulex, CMUdict and CMUdict_noisy corpora, respectively.

[10] http://www.itl.nist.gov/iad/mig/tools/

TABLE 5.7: Percentage of input words where one model ($Model_A$) provides the correct phoneme-sequence hypotheses while another model ($Model_B$) provides an incorrect-sequence hypotheses. The results in this table are based on Fig.5.1 and Tables 5.4 and 5.5. When comparing the result of two models trained using the same method, the result in bold font indicates the model with higher percentage of correct phoneme-sequence hypotheses. For example, in the result for the NETtalk corpus, one cell [$Model_A$(Slearp.reverse), $Model_B$(Slearp)] has a higher percentage than its comparative cell [$Model_A$(Slearp), $Model_B$(Slearp.reverse)].

$Model_A$ = correct and $Model_B$ = incorrect		$Model_B$						
		1_Slearp	2_Phon.	3_DirecTL+	4_Slearp .reverse	5_Phon .reverse	6_DirecTL+ .reverse	
$Model_A$	1_Slearp	0	**9.03%**	6.75%	4.25%	**8.66%**	**6.99%**	NETtalk
	2_Phon.	6.55%	0	7.75%	6.37%	2.69%	8.00%	
	3_DirecTL+	**7.05%**	**10.52%**	0	6.65%	**10.11%**	**3.02%**	
	4_Slearp.reverse	**5.20%**	**9.79%**	**7.31%**	0	**9.46%**	**7.62%**	
	5_Phon.reverse	6.70%	**3.21%**	7.85%	6.55%	0	8.03%	
	6_DirecTL+.reverse	6.88%	**10.37%**	2.62%	6.57%	**9.89%**	0	
$Model_A$	1_Slearp	0	**2.68%**	**4.35%**	**0.73%**	**2.67%**	**4.41%**	Brulex
	2_Phon.	1.52%	0	**4.53%**	1.52%	**0.84%**	**14.54%**	
	3_DirecTL+	1.33%	2.68%	0	1.42%	2.71%	**0.56%**	
	4_Slearp.reverse	0.66%	**2.61%**	**4.36%**	0	**2.61%**	**4.42%**	
	5_Phon.reverse	1.45%	0.78%	**4.50%**	1.46%	0	**4.53%**	
	6_DirecTL+.reverse	1.35%	2.64%	0.51%	1.43%	2.69%	0	
$Model_A$	1_Slearp	0	**6.74%**	**8.84%**	4.35%	**6.60%**	**8.91%**	CMUdict
	2_Phon.	6.00%	0	**9.23%**	5.57%	2.09%	**9.30%**	
	3_DirecTL+	5.64%	6.77%	0	5.03%	6.64%	**2.38%**	
	4_Slearp.reverse	**5.14%**	**7.10%**	**9.03%**	0	**6.95%**	**9.12%**	
	5_Phon.reverse	6.01%	**2.23%**	**9.24%**	5.56%	0	**9.31%**	
	6_DirecTL+.reverse	5.66%	6.79%	2.33%	5.09%	6.66%	0	
$Model_A$	1_Slearp	0	**6.83%**	**9.89%**	4.32%	**6.54%**	**10.07%**	CMUdict_noisy
	2_Phon.	5.99%	0	**10.57%**	5.67%	2.53%	**10.54%**	
	3_DirecTL+	5.38%	6.88%	0	5.23%	6.76%	**2.26%**	
	4_Slearp.reverse	**4.73%**	**6.92%**	**10.17%**	0	**6.86%**	**10.34%**	
	5_Phon.reverse	6.10%	**2.93%**	**10.84%**	6.00%	0	**10.84%**	
	6_DirecTL+.reverse	5.44%	6.75%	2.15%	5.29%	6.66%	0	

Surprisingly, when using reversed g-p sequences rather than conventional sequences, there was a slight improvement ($0.4 \sim 1\%$ for 4_Slearp.reverse and $0.2 \sim 0.5\%$ for 5_Phon.reverse), with the exception of the DIRECTL+ models (i.e., 6_DIRECTL+.reverse) and the Brulex corpus.

When the three models based on the selected methods (viz., SSMCW-, WFST- and MIRA-based methods) were combined, the evaluation results in Tables 5.4 and 5.5 further reveal that our first proposed PTN-based architecture can improve the performance of G2P conversion. PTN(4+5+6), the model with reversed g-p sequences, typically outperformed PTN(1+2+3), the same model but with conventional g-p sequences. Owing to the fact that reversed g-p sequences allow each single model to train an additional and superior model, the number of models and phoneme-sequence hypotheses for PTN-based G2P conversion doubles. Thus, the entire model performance improves. For example,

TABLE 5.8: Percentage of words measured from the OOV dataset for different corpora. This measurement is needed to analyze the correctness and incorrectness between the input sequence hypotheses and the output sequence of the PTN. Here, the results belong to PTN(1+...+6) and compactPTN(A+...+F). *The second-row results in bold font are misjudged words. "Could be correct" refers to the result obtained on condition that the voting method could perfectly select the best phoneme-sequence from the generated PTN.*

A set of conditions			Percentage of words measured from the OOV dataset (%)				
Phoneme-sequence hypotheses (1, 2,..., 6) or (A, B, ..., F) as inputs		Output sequence of the PTN	PTN (1+...+6)				compactPTN (A+...+F)
Status	No. of sequences	Status	NETtalk	Brulex	CMUdict	CMUdict_noisy	CMUdict_noisy
(In)Correct	Some	Correct	19.15%	7.84%	17.83%	18.85%	9.70%
(In)Correct	**Some**	**Incorrect**	**10.76%**	**1.99%**	**8.94%**	**9.42%**	**6.44%**
Correct	All	Correct	53.01%	87.33%	57.32%	56.43%	65.86%
Correct	All	Incorrect	0%	0%	0%	0%	0%
Incorrect	All	Correct	0.01%	0%	0.03%	0.06%	0%
Incorrect	All	Incorrect	17.08%	2.84%	15.88%	15.25%	18.01%
(In)Correct	Some	*Could be correct*	29.91%	9.83%	26.77%	28.27%	16.14%
Incorrect	All	*Could be correct*	1.14%	0.08%	1.23%	1.00%	0.88%

PTN(1+...+6) improved the WAcc of the best baseline models for NETtalk, Brulex, CMUdict, and CMUdict_noisy from 71.93% to 73.45%, 95.65% to 95.98%, 73.91% to 75.17% and 73.96% to 75.30%, respectively.

As explained in Section 5.4, the compact PTN-based architecture for G2P conversion has been proposed in order to minimize the risk from combining inaccurate and accurate methods. Because the size of the training data increases after using GGRs, and despite using the same representation of the grapheme sequence, the results from both (B-Slearp.GGR$_0$ versus A-Slearp) and (E-Slearp.GGR$_0$.reverse versus D-Slearp.reverse) in Table 5.6 demonstrate another method for increasing the performance of the baseline models other than the use of the reversed g-p sequence. By applying both techniques—GGRs and reversed g-p sequences—it is sufficient to use only the most accurate method (e.g., the SSMCW-based method in the Slearp toolkit) when implementing as many models as needed. After merging the hypotheses generated from all of those models with respect to the second proposed architecture (in Fig.5.2), the results from the compactPTN(A+...+F), evaluated using the CMUdict_noisy corpus, show even more improvement in terms of the PAcc and WAcc.

5.6 Discussion

The results in Tables 5.4 and 5.5 demonstrate that there are two ways to improve the performance of each separated model, namely GGRs and reversed g-p sequences.

The previous evaluation results in Tables 5.4 and 5.5 show that models using reversed g-p sequences generally outperformed those using conventional g-p sequences. After analyzing the data, we believe that some conventional sequences and their corresponding reversed g-p sequences were aligned differently owing to differing representations. Hence, we can assume that using the reversed g-p sequences provides better-aligned data for G2P conversion.

In order to appreciate the quality and helpfulness of the phoneme-sequence hypotheses involved in generating the PTN, we conducted an analysis of the sequences, inspired by McNemar's test theory. By calculating the percentage of words for which their corresponding phoneme sequences could be correctly established by one model (noted as $Model_A$) but not another (noted as $Model_B$), we can observe that the comparing results between any two different models in Table 5.7 are bigger than zero percentage for all the corpora. This means that when one model generates an incorrect phoneme sequence, other models can generate the correct sequence. In addition, by comparing two models, especially models using the same method but with a different representation of the grapheme sequence (i.e., the conventional and reversed g-p sequences), we can assume that one model (or an accurate model) will not provide all of the correct results that were provided by another model (or an inaccurate model). This is because it is still likely that one model will generate the correct phoneme-sequence hypothesis, even when another cannot. For instance, a comparison between the Slearp.reverse and Slearp models using the NETtalk dataset shows that 5.20% of the words correctly phoneticized with the Slearp.reverse model were incorrectly phoneticized by Slearp, but only 4.25% the other way around (i.e., correctly phoneticized by Slearp, but not by Slearp.reverse). This evidence strongly reinforce the point that combining multiple models for G2P conversion is more effective than using any single model.

On the other hand, we used the eight conditions in Table 5.8 to analyze the relations in terms of correctness and incorrectness between the phoneme-sequence hypotheses generated from various source models and the output of the PTN-based model. These eight conditions are as follows:

- Some hypotheses are correct → Output sequence of PTN is correct
- Some hypotheses are correct → Output sequence of PTN is incorrect
- All hypotheses are correct → Output sequence of PTN is correct
- All hypotheses are correct → Output sequence of PTN is incorrect
- All hypotheses are incorrect → Output sequence of PTN is correct
- All hypotheses are incorrect → Output sequence of PTN is incorrect
- Some hypotheses are correct → Output of PTN is "Could be correct"
- All hypotheses are incorrect → Output of PTN is "Could be correct"

Results based on the second condition (i.e. the second row in Table 5.8) indicate that 10.76%, 1.99%, 8.94%, and 9.42% of the OOV words in NETtalk, Brulex, CMUdict, and CMUdict_noisy, respectively, were misjudged when using the first proposed PTN-based architecture. Moreover, the misjudged results from CMUdict_noisy were reduced to 6.44% when using the second proposed architecture. This shows that the proposed architectures can nevertheless improve the model performance when selecting a better technique for determining the best phoneme sequence from the PTN sequence.

Even when all of the phoneme sequence hypotheses are incorrect, the PTN-based G2P conversion is still able to select the best phoneme candidate from each sequence (e.g., 0.01% for NETtalk, 0.03% for CMUdict, and 0.06% for CMUdict_noisy). The example in Table 5.9 demonstrates that the PTN-based model can produce a correct output phoneme sequence for the word "BERENDS" even when all of the generated sequence hypotheses are incorrect.

By supposing that the voting method could perfectly select the best output phoneme sequence from the generated PTN, the last row of Table 5.8 shows that the previous results could be improved to 1.14%, 0.08%, 1.23%, and 1.00% for NETtalk, Brulex, CMUdict CMUdict_noisy, respectively; in addition, if we also counted the cases that at least one correct phoneme-sequence hypothesis is used in the PTN generation, then both Tables 5.4, 5.5 and 5.8 show that the performance of the PTN-based G2P conversion would be highly improved from 73.45% to 84.06% ($1.14\% + 29.91\% + 53.01\%$) for NETtalk, from 95.98% to 97.24% ($0.08\% + 9.83\% + 87.33\%$) for Brulex, from 75.17% to 85.32% ($1.23\% + 26.77\% + 57.32\%$) for CMUdict, and from 75.30% to 85.70% ($1\% + 28.27\% + 56.43\%$) for CMUdict_noisy. These large improvements give us hope for the future challenge, which means that the voting method in our proposed PTN-based architectures for G2P conversion need to be improved.

The evaluation results for the compact version of the proposed PTN-based G2P conversion in Table 5.6 demonstrate that the novel use of reversed g-p sequences and GGRs improves PTN-based G2P conversions, even when only a single method is used. By comparing the evaluation results provided by the PTN-based architecture and its compact version, the results using the CMUdict_noisy corpus in Table 5.8 show that 18.85% of the correct words while using the first architecture, but only 9.70% of correct words while using the second architecture, has to take risk in the voting process. Thus, our compact PTN-based G2P conversion effectively minimizes risk in the voting process from combining inaccurate models with accurate ones. Furthermore, many different PTN-based architectures will be proposed to address challenges to G2P conversion in the future.

TABLE 5.9: Example showing how a PTN-based G2P conversion can establish a correct output phoneme sequence even when all of the sequence hypotheses are incorrect.

Reference:	"BERENDS" → /B	EH	R	EH	N	D	Z/	
1_Slearp:	B	EH	R	AH	N	D	Z	Aligned sequence hypotheses
2_Phon.:	B	EH	R	EH	N		Z	
3_DirecTL+:	B		ER	EH	N	D	Z	
4_Slearp.reverse:	B	EH	R	AH	N	D	Z	
5_Phon.reverse:	B	EH	R	EH	N		Z	
6_DirecTL+.reverse:	B	EH	R	EH	N		Z	
PTN sequence:	B	$\left\{\begin{matrix} EH \\ @ \end{matrix}\right\}$	$\left\{\begin{matrix} R \\ ER \end{matrix}\right\}$	$\left\{\begin{matrix} EH \\ AH \end{matrix}\right\}$	N	$\left\{\begin{matrix} D \\ @ \end{matrix}\right\}$	Z	
	↓	↓	↓	↓	↓	↓	↓	
Voting(Output):	B	EH	R	EH	N	D	Z	

5.7 Summary

In this chapter, we showed that the proposed PTN-based G2P conversion is a novel and effective method for improving the quality of phoneme prediction for OOV words. The proposal combines different approaches to phoneme prediction in order to address the various problems encountered by G2P conversion. It also provides significant and consistently improved results compared to models based on a single approach. The novel use of reversed g-p sequences and GGRs in this chapter can make complementary models that allow to generate new hypotheses so that ensemble of them has considerable gain for the PTN-based G2P conversion model, and it also minimizes the risk associated with combining accurate and inaccurate models. Moreover, we demonstrated that the representation of both graphemic and phonemic information plays an important role in improving model performance.

Chapter 6

Conclusions

This thesis is dedicated to exploring way of improving the predictive quality of G2P conversion model which is widely adopted in various systems related to speech technologies. In order to improve its performance in terms of correctness and reliability, we have proposed three different approaches each of which is implemented with respect to the paradigm (*Graphemes* $\Rightarrow$ *Phonemes* $\Rightarrow$ *Phonemes*) in our proposed two-stage architecture-based approach for G2P conversion.

In Chapter 3 of the thesis, it provided a study on our firstly proposed application of G2P conversion. The analysis on the start-of-the-art single-stage neural network-based approach has shown that using only one neural network is not enough for solving some complicated problems encountered by G2P conversion. As a result, our first approach based on multi-layered neural networks has been proposed and called as "A two-stage neural network-based approach focusing on both grapheme and phoneme contexts". In this chapter, our research aimed to improve the conversion performance by dealing with two specific issues includes: (1) a many-to-many relation between letters and phonemes in the conversion and (2) a problem of conflicting phonemes at the output level. To predict the final output phonemes corresponding to the input text, this approach focused on the phoneme rather than grapheme patterns. 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. In terms of phoneme and word accuracy, the evaluation results showed 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. For its further improvements, an integration of a pseudo-phoneme- [87] and graphones-based techniques [21] into our approach seems to reduce the conflicting problems between phonemes at the

first-stage neural network. Instead of using OBC encoding algorithm to represent the input and output information of the G2P conversion model, another encoding algorithm should be token into consideration for reducing the number of neurons as well as size of the neural network model. A syllable-based approach may also help to improve the model performance as well [106], [18]. Moreover, instead of using the FANN library, we should consider to implement each stage model using other machine learning libraries or toolkits such as the RNNLM toolkit[28], OpenANN[29], or RNNLib[30] toolkits.

In Chapter 4, we utilized the exiting WFST-based method available in the Phonetisaurus toolkit in place of multi-layered neural network to implement our secondly proposed two-stage architecture-based G2P conversion. It has been shown in this chapter that using new grapheme generation rules that were designed to enable extra detail for vowel graphemes could improve the performance of G2P conversion in terms of phoneme and word accuracy. The new phoneme-prediction method allowed the second-stage model to learn the pronunciation rules more easily than the first-stage model because both the grapheme sequences and the preliminary phoneme-sequences had already been identified at the input level. The evaluation results measured on the CMUDict corpus at the end of this chapter also demonstrated that a multiple-models combination could become a very helpful and flexible strategy to design a highly accurate architecture for tackling simultaneously different problems encountered by G2P conversion. For the further challenge, more effective rules to reduce the complexity of pronunciation in both training and testing datasets should be designed because they can potentially boost the word accuracy of our proposed approach to a higher level.

The global vocabulary is in continuous expansion like the universe itself, so we need approaches capable to embrace these changes. And last, but definitely not the least is the multilingual aspect of the rapidly developing global village. A non-stop development of speech technologies created a need for a highly intelligible, adaptable and multilingual G2P conversion system that can deal accurately with the OOV words. Therefore, in Chapter 5, we showed that our thirdly proposed PTN-based G2P conversion is a novel and effective method for improving both the quality and the flexibility of phoneme prediction for OOV words. The proposal combines different approaches to phoneme prediction, in order to address the various problems encountered by G2P conversion. It also provides significant and consistently improved results compared to models based on a single approach, based on the evaluation results using various pronunciation dictionaries such as CMUdict, CMUdict_noisy, NETtalk and Brulex. Especially, the novel use of reversed g-p sequences and GGRs in this chapter not only reduces the number of models

[28]RNNLM http://www.fit.vutbr.cz/${\sim}$imikolov/rnnlm/
[29]OpenANN: http://openann.github.io/OpenANN-apidoc/
[30]RNNLib: http://sourceforge.net/p/rnnl/wiki/Home/

and training time for the PTN-based G2P conversion model, it also minimizes the risk associated with combining accurate and inaccurate models. Furthermore, the evaluation results of the PTN-based G2P conversion using GGRs also demonstrated that the representation of both graphemic and phonemic information plays an important role in improving the conversion performance.

In future work, we plan to create new and effective GGRs to further improve our proposed approach, enabling a trained model to generate more accurately output phoneme-sequence hypotheses, such that only two models (using conventional and reversed g-p sequences) will be sufficient for our PTN-based G2P conversion. Moreover, the hamming distance, calculated from the articulatory features of phonemes [119], shall be used for the DP alignment process in the ROVER system. Inspired by the LSTM-RNNs-based method for G2P conversion [120], we shall attempt to challenge our approach at the voting level with the use of a finite state transducer and a joint n-gram model, rather than relying on the simplistic voting method available in the ROVER system. We also think that the suffix information (as seen in Table C.1 in Appendix C) will be also useful for the improvements in the future On the other hand, we also expect that the PTN-based sequence of the input text will be very useful for dealing with the OOV search keywords in the spoken term detection system and other systems as well.

Appendix A

List of Publications

A.1 List of Articles

- S. Kheang, K. Katsurada, Y. Iribe, and T. Nitta, "Using reversed sequences and grapheme generation rules to extend the feasibility of a phoneme transition network-based grapheme-to-phoneme conversion," *IEICE Transactions on Information and Systems*, vol. E99-D, no. 4, April 2016.
- S. Kheang, K. Katsurada, Y. Iribe, and T. Nitta, "Solving the phoneme conflict in grapheme-to-phoneme conversion using a two-stage neural network-based approach," *IEICE Transactions on Information and Systems*, vol. E97-D, no. 4, pp. 901–910, April 2014.
- S. Kheang, K. Katsurada, Y. Iribe, and T. Nitta, "New grapheme generation rules for two-stage model-based grapheme-to-phoneme conversion," *Journal of ICT Research and Applications*, vol. 8, no. 2, pp. 157–174, 2014.

A.2 List of Conference Papers

- S. Kheang, K. Katsurada, Y. Iribe, and T. Nitta, "Model Prioritization Voting Schemes for Phoneme Transition Network-based Grapheme-to-Phoneme Conversion," in *Proc. of the International Conference on Computer and Information Science and Technology (CIST'15)*, (Ottawa, Canada), May 2015.
- S. Kheang, K. Katsurada, Y. Iribe, and T. Nitta, "Novel Two-Stage Model for Grapheme-to-Phoneme Conversion using New Grapheme Generation Rules," in *Proc. of ICAICTA*, (Bandung, Indonesia), pp. 110–115, August 2014.
- S. Kheang, K. Katsurada, Y. Iribe, and T. Nitta, "Improving the performance of letter-to-phoneme conversion by using two-stage neural network," in *IPSJ SIG Technical Report, Information Processing Society of Japan*, (Tokyo, Japan), 2012.

- S. Kheang, Y. Iribe, and T. Nitta, "Letter-to-phoneme conversion based on two-stage neural network focusing on letter and phoneme contexts," in *Proc. of Interspeech*, (Florence, Italy), 2011.

- K. Katsurada, G. Ishihara, S. Kheang, Y. Iribe, and T. Nitta, "Utilizing Confusion Network in the STD with Suffix Array and Its Evaluation on the NTCIR-11 SpokenQuery & Doc SQ-STD Task," in *Proc. of the NTCIR-11 Workshop Meeting*, (Tokyo, Japan), 2014.
- K. Katsurada, S. Miura, S. Kheang, Y. Iribe, and T. Nitta, "Acceleration of spoken term detection using a suffix array by assigning optimal threshold values to sub-keywords," in *Proc. of Interspeech*, (Lyon, France), pp. 11–14, 2013.
- K. Katsurada, K. Katsuura, S. Kheang, Y. Iribe, and T. Nitta, "Using multiple speech recognition results to enhance STD with suffix array on the NTCIR-10 SpokenDoc-2 task," in *Proc. of the NTCIR-10 Workshop Meeting*, (Japan), pp. 588–591, 2013.

Appendix B

Detailed Figures

FIGURE B.1: The results of 10-folds cross-validation of the **Slearp** models using conventional and reversed g-p sequences and evaluated using **NETtalk** dataset.

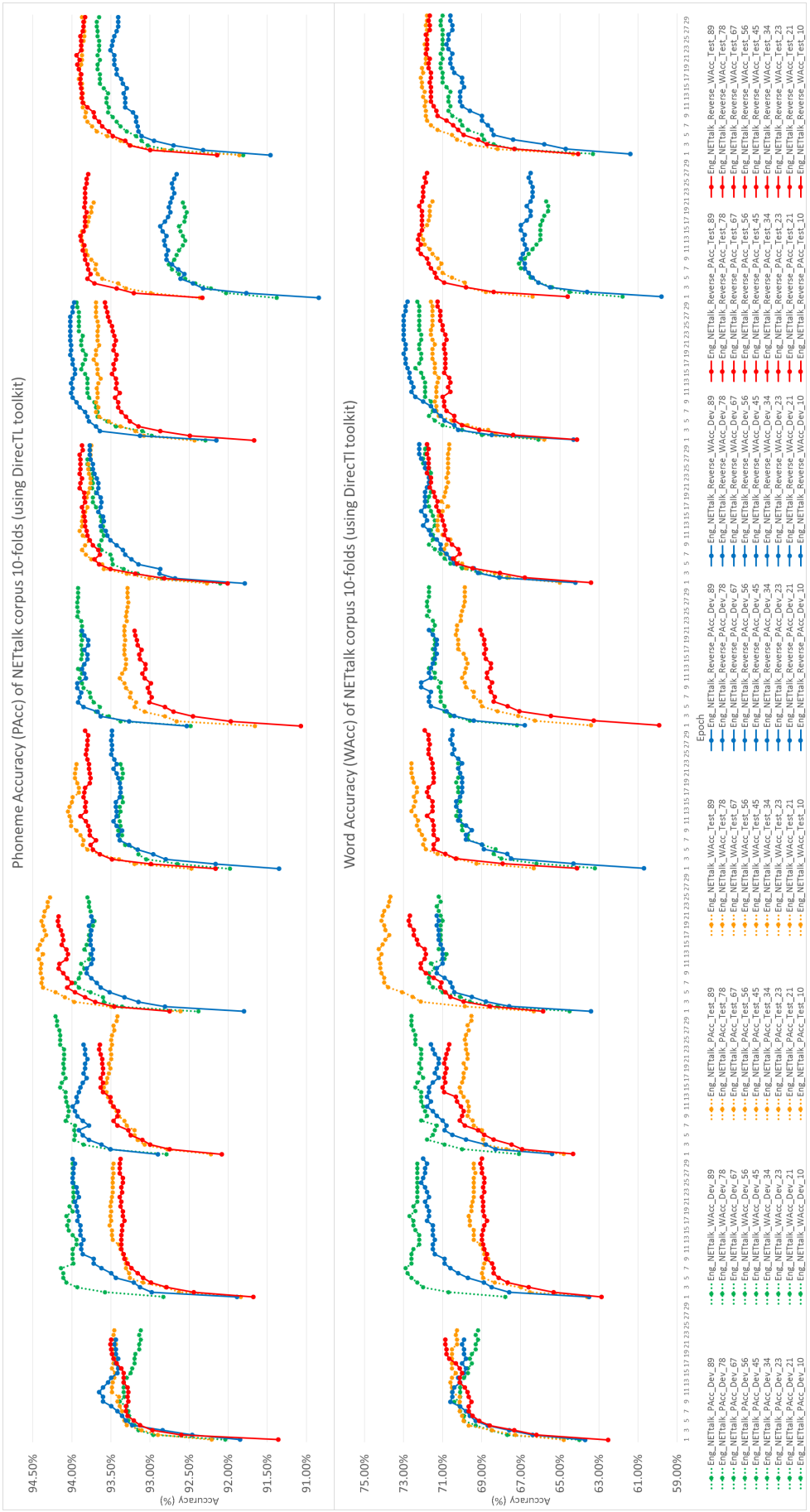


FIGURE B.2: The results of 10-folds cross-validation of the **DIRECTL+** models using conventional and reversed g-p sequences and evaluated using **NETtalk** dataset.

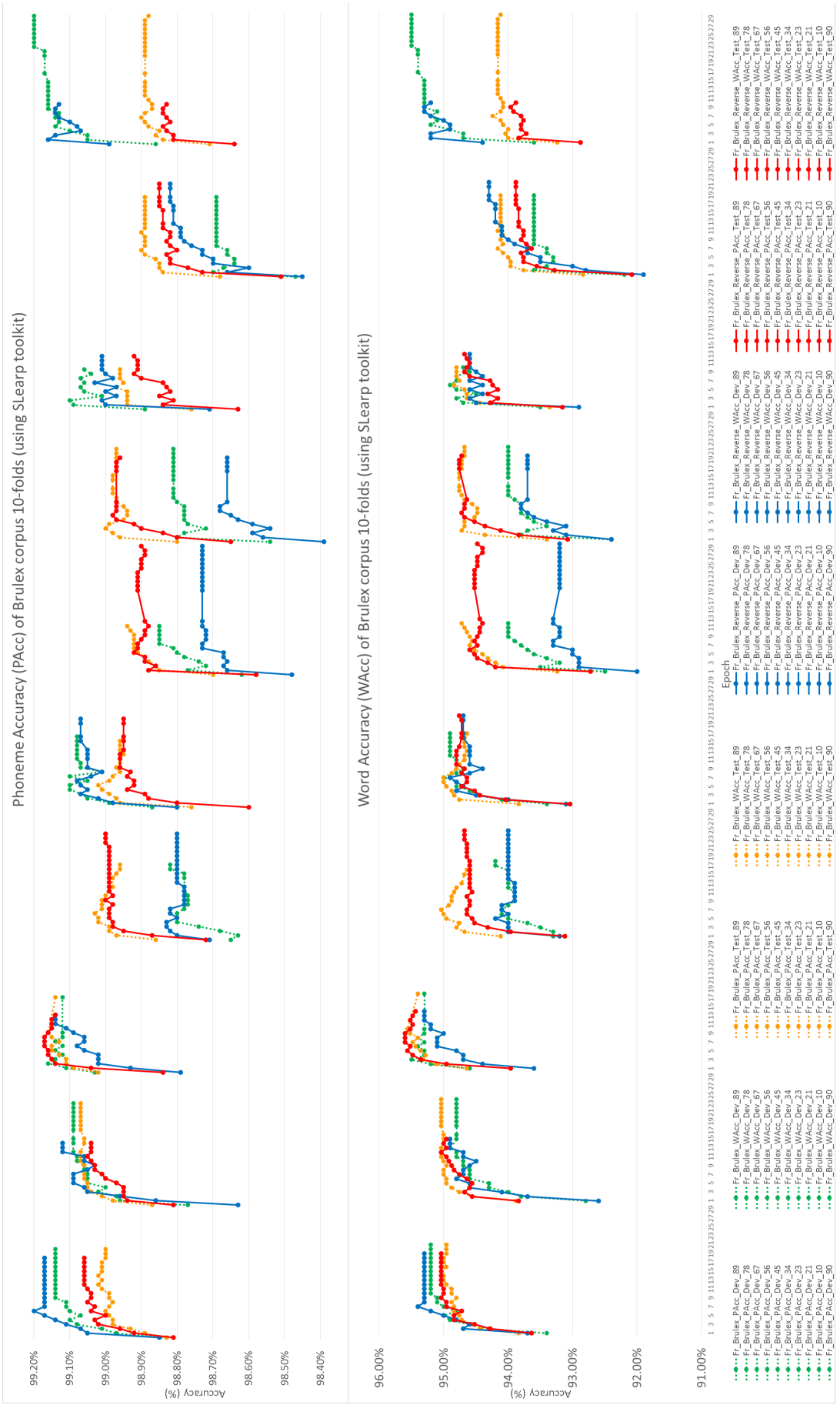


FIGURE B.3: The results of 10-folds cross-validation of the **Slearp** models using conventional and reversed g-p sequences and evaluated using **Brulex** dataset.

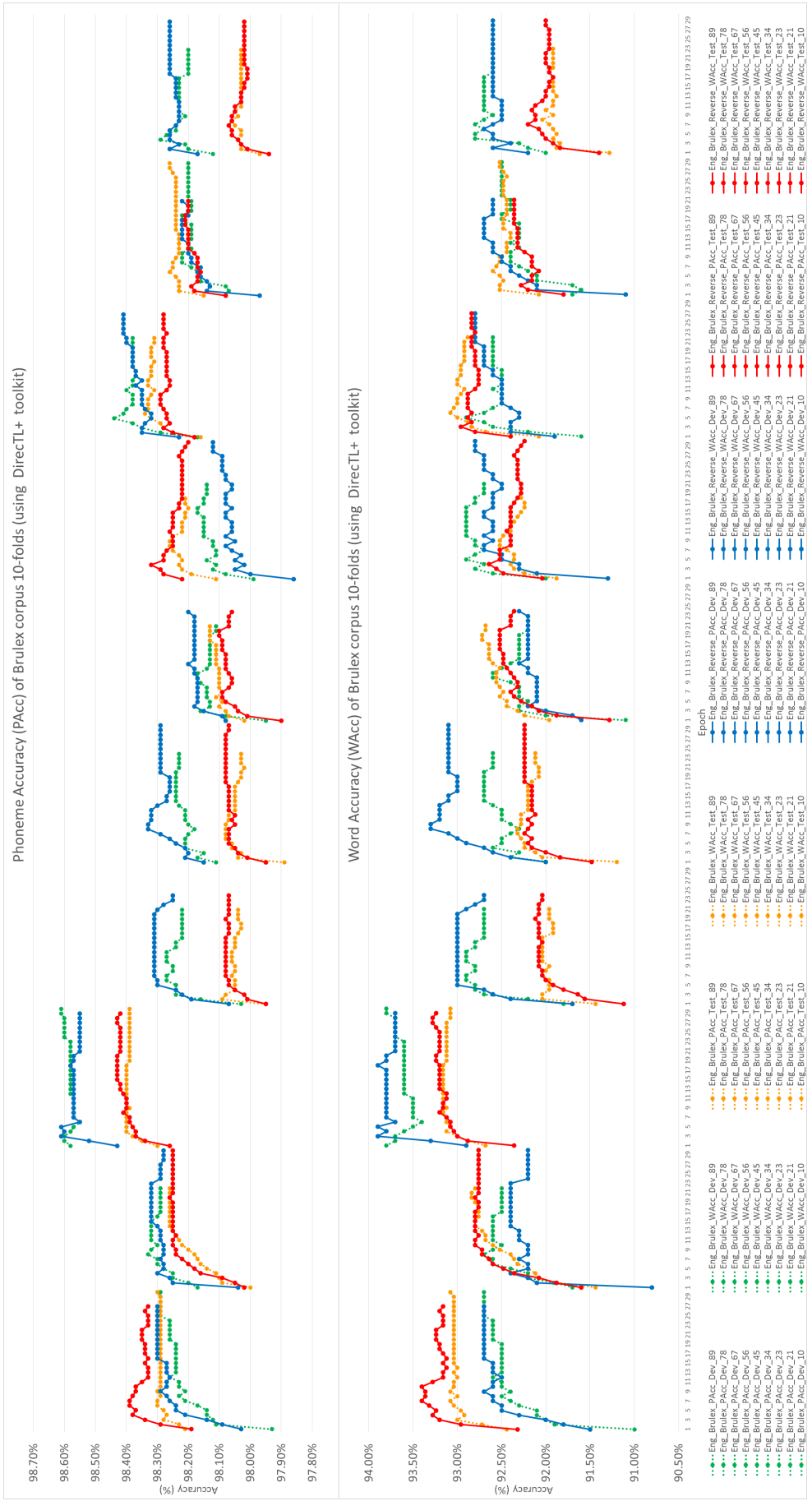


FIGURE B.4: The results of 10-folds cross-validation of the **DIRECTL+** models using conventional and reversed g-p sequences and evaluated using **Brulex** dataset.

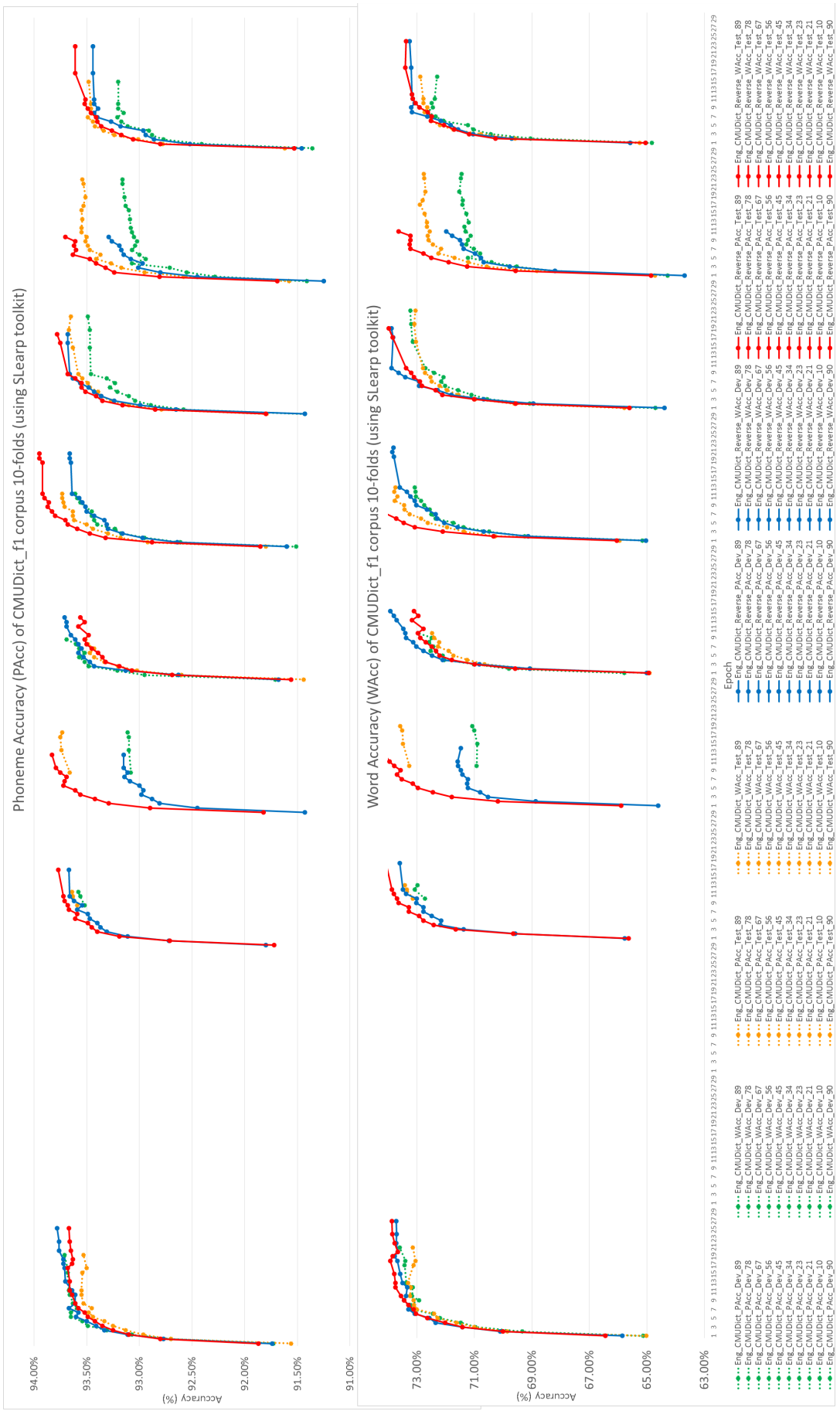


FIGURE B.5: The results of 10-folds cross-validation of the **Slearp** models using conventional and reversed g-p sequences and evaluated using **CMUdict** dataset.

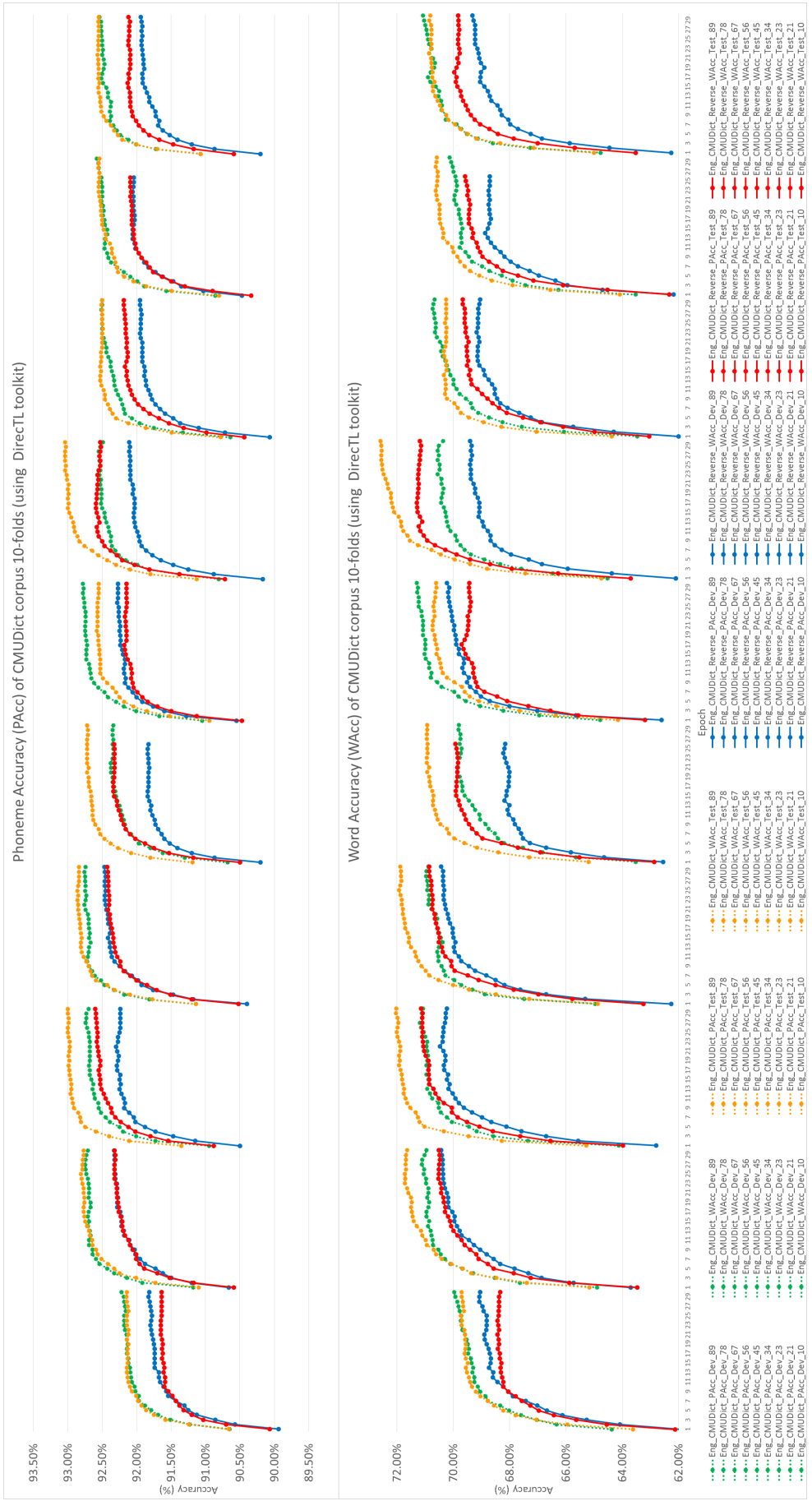


FIGURE B.6: The results of 10-folds cross-validation of the **DIRECTL+** models using conventional and reversed g-p sequences and evaluated using **CMUdict** dataset.

Appendix C

List of English Suffixes

TABLE C.1: List of English Suffixes (http://www.prefixsuffix.com/rootchart.php)

Root, Prefix or Suffix	Meaning	Examples
a, ac, ad, af, ag, al, an, ap, as, at	to, toward, near, in addition to, by	aside, accompany, adjust, aggression, allocate, annihilate, affix, associate, attend, adverb
a-, an-	not, without	apolitical, atheist, anarchy, anonymous, apathy, aphasia, anemia
ab, abs	away from, off	absolve, abrupt, absent
-able, -ible	Adjective: worth, ability	solvable, incredible
acer, acid, acri	bitter, sour, sharp	acerbic, acidity, acrid, acrimony
act, ag	do, act, drive	active, react, agent, active, agitate
acu	sharp	acute, acupuncture, accurate
-acy, -cy	Noun: state or quality	privacy, infancy, adequacy, intimacy, supremacy
-ade	act, product, sweet drink	blockade, lemonade
aer, aero	air, atmosphere, aviation	aerial, aerosol, aerodrome
ag, agi, ig, act	do, move, go	agent, agenda, agitate, navigate, ambiguous, action
-age	Noun: activity, or result of action	courage, suffrage, shrinkage, tonnage
agri, agro	pertaining to fields or soil	agriculture, agroindustry
-al	Noun: action, result of action	referral, disavowal, disposal, festival
-al, -ial, -ical	Adjective: quality, relation	structural, territorial, categorical
alb, albo	white, without pigment	albino, albite
ali, allo, alter	other	alias, alibi, alien, alloy, alter, alter ego, altruism
alt	high, deep	altimeter, altitude
am, ami, amor	love, like, liking	amorous, amiable, amicable, enamoured
ambi	both	ambidextrous
ambul	to walk	ambulatory, amble, ambulance, somnambulist
-an	Noun: person	artisan, guardian, historian, magician
ana, ano	up, back, again, anew	anode, anagram, anagenetic
-ance, -ence	Noun: action, state, quality or process	resistance, independence, extravagance, fraudulence
-ancy, -ency	Noun: state, quality or capacity	vacancy, agency, truancy, latency
andr, andro	male, characteristics of men	androcentric, android
ang	angular	angle
anim	mind, life, spirit, anger	animal, animate, animosity

ann, annu, enni	yearly	annual, annual, annuity, anniversary, perrenial
-ant, -ent	Noun: an agent, something that performs the action	disinfectant, dependent, fragrant
-ant, -ent, -ient	Adjective: kind of agent, indication	important, dependent, convenient
ante	before	anterior, anteroom, antebellum, antedate, antecedent antediluvian
anthrop	man	anthropology, misanthrope, philanthropy
anti, ant	against, opposite	antisocial, antiseptic, antithesis, antibody, antinomies, antifreeze, antipathy
anti, antico	old	antique, antiquated, antiquity
apo, ap, aph	away from, detached, formed	apology, apocalypse, aphagia
aqu	water	aqueous
-ar, -ary	Adjective: resembling, related to	spectacular, unitary
arch	chief, first, rule	archangel, architect, archaic, monarchy, matriarchy, patriarchy, Archeozoic era
-ard, -art	Noun: characterized	braggart, drunkard, wizard
aster, astr	star	aster, asterisk, asteroid, astronomy, astronaut
-ate	Noun: state, office, fuction	candidate, electorate, delegate
-ate	Verb: cause to be	graduate, ameliorate, amputate, colligate
-ate	Adjective: kind of state	inviolate
-ation	Noun: action, resulting state	specialization, aggravation, alternation
auc, aug, aut	to originate, to increase	augment , author, augment, auction
aud, audi, aur, aus	to hear, listen	audience, audio, audible, auditorium, audiovisual, audition, auricular, ausculate
aug, auc	increase	augur, augment, auction
aut, auto	self	automobile, automatic, automotive, autograph, autonomous, autoimmune
bar	weight, pressure	barometer
be	on, around, over, about, excessively, make, cause, name, affect	berate, bedeck, bespeak, belittle, beleaguer
belli	war	rebellion, belligerent, casus belli, bellicose
bene	good, well, gentle	benefactor, beneficial, benevolent, benediction, beneficiary, benefit
bi, bine	two	biped, bifurcate, biweekly, bivalve, biannual
bibl, bibli, biblio	book	bibliophile, bibliography, Bible
bio, bi	life	biography, biology, biometricsm biome, biosphere
brev	short	abbreviate, brevity, brief
cad, cap, cas, ceiv, cept, capt, cid, cip	to take, to seize, to hold	receive, deceive, capable, capacious, captive, accident, capture, occasion, concept, intercept, forceps, except, reciprocate

cad, cas	to fall	cadaver, cadence, cascade
-cade	procession	motorcade
calor	heat	calorie, caloric, calorimeter
capit, capt	head	decapitate, capital, captain, caption
carn	flesh	carnivorous, incarnate, reincarnation, carnal
cat, cata, cath	down, with	catalogue, category, catheter
caus, caut	burn, heat	caustic, cauldron, cauterize
cause, cuse, cus	cause, motive	because, excuse, accusation
ceas, ced, cede, ceed, cess	to go, to yield, move, go, surrender	succeed, proceed, precede, recede, secession, exceed, succession
cent	hundred	centennial, century, centipede
centr, centri	center	eccentricity, centrifugal, concentric, eccentric
chrom	color	chrome, chromosome, polychrome, chromatic
chron	time	chronology, chronic, chronicle chronometer, anachronism, synchronize
cide, cis, cise	to kill, to cut, cut down	fratricide, homicide, incision, incision, circumcision, scissors
circum	around	circumnavigate, circumflex, circumstance, circumcision, circumference, circumorbital, circumlocution, circumvent, circumscribe, circulatory
cit	call, start	incite, citation, cite
civ	citizen	civic, civil, civilian, civilization
clam, claim	cry out	exclamation, clamor, proclamation, reclamation, acclaim
clin	lean, bend	decline, aclinic, inclination
clud, clus claus	to close, shut	include, exclude, clause, claustrophobia, enclose, exclusive, reclusive, conclude
co, cog, col, coll, con, com, cor	with, together	cohesiveness, cognate, collaborate, convene, commitment, compress, contemporary, converge, compact, confluence, convenient, concatenate, conjoin, combine, correct
cogn, gnos	to know	recognize, cognizant, diagnose, agnostic, incognito, prognosis
com, con	fully	complete, compel, conscious, condense, confess, confirm
contr, contra, counter	against, opposite	contradict, counteract, contravene, contrary, counterspy, contrapuntal
cord, cor, cardi	heart	cordial, concord, discord, courage, encourage
corp	body	corporation, corporal punishment, corpse, corpulent, corpus luteum
cort	correct	escort, cortage
cosm	universe, world	cosmos, microcosm, cosmopolitan, cosmonaut
cour, cur, curr, curs	run, course	occur, excursion, discourse, courier, course
crat, cracy	rule	autocrat, aristocrat, theocracy, technocracy
cre, cresc, cret, crease	grow	create, crescent, accretion, increase

crea	create	creature, recreation, creation
cred	believe	creed, credo, credence, credit, credulous, incredulous, incredible
cresc, cret, crease, cru	rise, grow	crescendo, concrete, increase, decrease, accrue
crit	separate, choose	critical, criterion, hypocrite
cur, curs	run	current, concurrent, concur, incur, recur, occur, courier, precursor, cursive
cura	care	curator, curative, manicure
cycl, cyclo	wheel, circle, circular	Cyclops, unicycle, bicycle, cyclone, cyclic
de-	from, down, away, to do the opposite, reverse, against	detach, deploy, derange, decrease, deodorize, devoid, deflate, degenerate
dec, deca	ten, ten times	decimal, decade, decalogue, decimate, decathlon
dec, dign	suitable	decent decorate dignity
dei, div	God	divinity, divine, deity, divination, deify
dem, demo	people, populace, population	democracy, demography, demagogue, epidemic
dent, dont	tooth	dental, denture, orthodontist, periodontal
derm	skin, covering	hypodermic, dermatology, epidermis, taxidermy
di-, dy-	two, twice, double	divide, diverge, diglycerides
dia	through, across, between	diameter, diagonal, dialogue dialect, dialectic, diagnosis, diachronic
dic, dict, dit	say, speak	dictation, dictionary, dictate, dictator, Dictaphone, edict, predict, verdict, contradict, benediction
dis, dif	not, opposite of, reverse, separate, deprive of, away	dismiss, differ, disallow, disperse, dissuade, divide, disconnect, disproportion, disrespect, distemper, disarray
dit	give	credit, audit
doc, doct	teach, prove	docile, doctor, doctrine, document, dogma, indoctrinate
domin	master, that which is under control	dominate, dominion, predominant, domain
don	give	donate, condone
dorm	sleep	dormant, dormitory
dox	thought, opinion, praise	orthodox, heterodox, paradox, doxology
-drome	run, step	syndrome, aerodrome, velodrome
duc, duct	to lead, pull	produce, abduct, product, transducer, viaduct, aqueduct, induct, deduct, reduce, induce
dura	hard, lasting	durable, duration, endure
dynam	power	dynamo, dynamic, dynamite, hydrodynamics

dys-	bad, abnormal, difficult, impaired, unfavorable	dysfunctional, dyslexia, dyspathy
e-	not, missing, out, fully, away, computer network related	emit, embed, eternal,ether, erase, email, e-tailer
ec-	out of, outside	echo, eclipse, eclectic, ecesis, ecstasy, exzema
eco-	household, environment, relating to ecology or economy	ecology, economize, ecospheres, ecomanagement
ecto-	outside, external	ectomorph, ectoderm, ectoplasm
-ed	Verb: past tense	dressed, faded, patted, closed, introduced
-ed	Adjective: having the quality or characteristics of	winged, moneyed, dogged, tiered
-en	Verb: to cause to become	lengthen, moisten, sharpen
-en	Adjective: material	golden, woolen, silken
en-, em-	put into, make, provide with, surround with	enamor, embolden, enslave, empower, entangle
-ence, -ency	Noun: action or process, quality or state	reference, emergency, dependence, eminence, latency
end-	inside, within	endorse, endocardial, endergonic, endoskeleton, endoscope, endogenous
epi-	upon, close to, over, after, altered	epicenter, epicarp, epilogue, epigone, epidiorite
equi-	equal	equidistant, equilateral, equilibrium, equinox, equation, equator
-er, -ier	Adjective: comparative	better, brighter, sooner, hotter, happier
-er, -or	Noun: person or thing that does something	flyer, reporter, player, member, fryer, collector, concentrator
-er, -or	Verb: action	ponder, dishonor, clamor
erg	work, effect	energy, erg, allergy, ergometer, ergograph, ergophobia
-ery	collective qualities, art, practice, trade, collection, state, condition	snobbery, bakery, geenery, gallery, slavery
-es, -ies	Noun: plural of most nouns ending in -ch, -s, -sh, -o and -z and some in -f and -y	passes, glasses, ladies, heroes
-es, -ies	Verb: third person singular present indicative of verbs that end in -ch, -s, -sh, - and some in -y	blesses, hushes, fizzes, defies
-ess	female	actress, goddess, poetess
-est, -iest	Adjective or Adverb: superlative	latest, strongest, luckiest, lyingest
ev-, et-	time, age	medieval, eternal
ex-	out of, away from, lacking, former	exit, exhale, exclusive, exceed, explosion, ex-mayor
exter-, extra-, extro-	outside of, beyond	external, extrinsic, extraordinary, extrapolate, extraneous, extrovert
fa, fess	speak	fable, fabulous, fame, famous, confess, profess
fac, fact, fec, fect, fic, fas, fea	do, make	difficult, fashion, feasible, feature, factory, fact, effect, manufacture, amplification, confection

fall, fals	deceive	fallacy, falsify, fallacious
femto	quadrillionth	femtosecond
fer	bear, carry	ferry, coniferous, fertile, defer, infer, refer, transfer
fic, feign, fain, fit, feat	shape, make, fashion	fiction, faint, feign
fid	belief, faith	confide, diffident, fidelity
fid, fide, feder	faith, trust	confidante, fidelity, confident, infidelity, infidel, federal, confederacy, semper fi
fig	shape, form	figurem, effigy, figure, figment
fila, fili	thread	filigree, filament, filter, filet, filibuster
fin	end, ended, finished	final, finite, finish, confine, fine, refine, define, finale
fix	repair, attach	fix, fixation, fixture, affix, prefix, suffix
flex, flect	bend	flex, reflex, flexible, flexor, inflexibility, reflect, deflect,circumflex
flict	strike	affliction, conflict, inflict
flu, fluc, fluv, flux	flow	influence, fluid, flue, flush, fluently, fluctuate, reflux, influx
-fold	Adverb: in a manner of, marked by	fourfold
for, fore	before	forecast, fortune, foresee
forc, fort	strength, strong	effort, fort, forte, fortifiable, fortify, forte, fortitude
form	shape, resemble	form, format, conform, formulate, perform, formal, formula
fract, frag, frai	break	fracture, infraction, fragile, fraction, refract, frail
fuge	flee	subterfuge, refuge, centrifuge
-ful	Noun: an amount or quanity that fills	mouthful
-ful	Adjective: having, giving, marked by	fanciful
fuse	pour	confuse, transfuse
-fy	make, form into	falsify, dandify
gam	marriage	bigamy, monogamy, polygamy
gastr, gastro	stomach	gastric, gastronomic, gastritis, gastropod
gen	kind	generous
gen	birth, race, produce	genesis, genetics, eugenics, genealogy, generate, genetic, antigen, pathogen
geo	earth	geometry, geography, geocentric, geology
germ	vital part	germination, germ, germane
gest	carry, bear	congest, gestation
giga	billion	gigabyte, gigaflop
gin	careful	gingerly
gloss, glot	tongue	glossary, polyglot, epiglottis

glu, glo	lump, bond, glue	glue, agglutinate, conglomerate
gor	to gather, to bring together	category, categorize
grad, gress, gree	to gather, to bring together, step, go	grade, degree, progress, gradual, graduate, egress
graph, gram, graf	write, written, draw	graph, graphic, autograph, photography, graphite, telegram, polygraph, grammar, biography, lithograph, graphic
grat	pleasing	congratulate, gratuity, grateful, ingrate
grav	heavy, weighty	grave, gravity, aggravate, gravitate
greg	herd	gregarious, congregation, segregate, gregarian
hale, heal	make whole, sound	inhale, exhale, heal, healthy, healthiness
helio	sun	heliograph, heliotrope, heliocentric
hema, hemo	blood	hemorrhage, hemoglobin, hemophilia, hemostat
her, here, hes	stick	adhere, cohere, cohesion, inherent, hereditary, hesitate
hetero	other, different	heterodox, heterogeneous, heterosexual, heterodyne
hex, ses, sex	six	hexagon, hexameter, sestet, sextuplets
homo	same	homogenize, homosexual, homonym, homophone
hum, human	earth, ground, man	humus, exhume, humane
hydr, hydra, hydro	water	dehydrate, hydrant, hydraulic, hydraulics, hydrogen, hydrophobia
hyper	over, above	hyperactive, hypertensive, hyperbolic, hypersensitive, hyperventilate, hyperkinetic
hypn	sleep	hypnosis, hypnotherapy
-ia	Noun: names, diseases	phobia
-ian, an	Noun: related to, one that is	pedestrian, human
-iatry	Noun: art of healing	psychiatry
-ic	Adjective: quality, relation	generic
-ic, ics	Noun: related to the arts and sciences	arithmetic, economics
-ice	Noun: act	malice
-ify	Verb: cause	specify
ignis	fire	ignite, igneous, ignition
-ile	Adjective: having the qualities of	projectile
in, im	into, on, near, towards	instead, import
in, im, il, ir	not	illegible, irresolute, inaction, inviolate, innocuous, intractable, innocent, impregnable, impossible, imposter
infra	beneath	infrared, infrastructure

-ing	Noun: material made for, activity, result of an activity	flooring, swimming, building
-ing	Verb: present participle	depicting
-ing	Adjective: activity	cohering
inter	between, among	international, intercept, interject, intermission, internal, intermittent
intra	within, during, between layers, underneath	intramural, intranet, intranatal
intro	into, within, inward	interoffice, introvert, introspection, introduce
-ion	Noun: condition or action	abduction
-ish	Adjective: having the character of	newish
-ism	Noun: doctrine, belief, action or conduct	formalism
-ist	Noun: person or member	podiatrist
-ite	Noun: state or quality	graphite
-ity, ty	Noun: state or quality	lucidity, novelty
-ive	Noun: condition	native
-ive, -ative, -itive	Adjective: having the quality of	festive, cooperative, sensitive
-ize	Verb: cause	fantasize
jac, ject	throw	reject, eject, project, trajectory, interject, dejected, inject, ejaculate, adjacent
join, junct	join	adjoining, enjoin, juncture, conjunction, injunction, conjunction
judice	judge	prejudice
jug, junct, just	to join	junction, adjust, conjugal
juven	young	juvenile, rejuvenate
labor	work	laborious, belabor
lau, lav, lot, lut	wash	launder, lavatory, lotion, ablution, dilute
lect, leg, lig	choose, gather, select, read	collect, legible, eligible
leg	law	legal, legislate, legislature, legitimize
-less	Adjective: without, missing	motiveless
levi	light	alleviate, levitate, levity
lex, leag, leg	law	legal, college, league
liber, liver	free	liberty, liberal, liberalize, deliverance
lide	strike	collide, nuclide
liter	letters	literary, literature, literal, alliteration, obliterate
loc, loco	place, area	location, locally, locality, allocate, locomotion
log, logo, ology	word, study, say, speech, reason, study	catalog, prologue, dialogue, zoology, logo

loqu, locut	talk, speak	eloquent, loquacious, colloquial, circumlocution
luc, lum, lun, lus, lust	light	translucent, luminary, luster, luna, illuminate, illustrate
lude	play	prelude
-ly	Adverb: in the manner of	fluently
macr-, macer	lean	emaciated, meager
magn	great	magnify, magnificent, magnanimous, magnate, magnitude, magnum
main	strength, foremost	mainstream, mainsail, domain, remain
mal	bad, badly	malformation, maladjusted, dismal, malady, malcontent,malfunction, malfeasance, maleficent
man, manu	hand, make, do	manual, manage, manufacture, manacle, manicure, manifest, maneuver, emancipate, management
mand	command	mandatory, remand, mandate
mania	madness	mania, maniac, kleptomania, pyromania
mar, mari, mer	sea, pool	marine, marsh, maritime, mermaid
matri	mother	matrimony, maternal, matriarchate, matron
medi	half, middle, between, halfway	mediate, medieval, Mediterranean, mediocre, medium
mega	great, million	megaphone, megaton, megaflop, megalomaniac, megabyte, megalopolis
mem	recall, remember	memo, commemoration, memento, memoir, memorable
ment	mind	mental, mention
-ment	Noun: condition or result	document
meso	middle	mesomorph, mesoamerica, mesosphere
meta	beyond, change	metaphor, metamorphosis, metabolism, metahistorical, metainformation
meter	measure	meter, voltammeter, barometer, thermometer
metr	admeasure, apportion	metrics, asymmetric, parametric, telemetry
micro	small, millionth	microscope, microfilm, microcard, microwave, micrometer, microvolt
migra	wander	migrate, emigrant, immigrate
mill, kilo	thousand	millennium, kilobyte, kiloton
milli	thousandth	millisecond, milligram, millivolt
min	little, small	minute, minor, minuscule
mis	wrong, bad, badly	misconduct, misinform, misinterpret, mispronounce, misnomer, mistake, misogynist
mit, miss	send	emit, remit, submit, admit, commit, permit, transmit, omit, intermittent, mission, missile
mob, mov, mot	move	motion, remove, mobile, motor
mon	warn, remind	monument, admonition, monitor, premonition

mono	one	monopoly, monotype, monologue, mononucleosis, monorail, monotheist,
mor, mort	mortal, death	mortal, immortal, mortality, mortician, mortuary
morph	shape, form	amorphous, dimorphic, metamorphosis, morphology, polymorphic, morpheme, amorphous
multi	many, much	multifold, multilingual, multiped, multiply, multitude, multipurpose, multinational
nano	billionth	nanosecond, nanobucks
nasc, nat, gnant, nai	to be born	nascent, native, pregnant, naive
nat, nasc	to be from, to spring forth	innate, natal, native, renaissance
neo	new	Neolithic, nuveau riche, neologism, neophyte, neonate
-ness	Noun: state, condition, quality	kindness
neur	nerve	neuritis, neuropathic, neurologist, neural, neurotic
nom	law, order	autonomy, astronomy, gastronomy, economy
nom, nym	name	nominate, synonym
nomen, nomin	name	nomenclature, nominate, ignominious
non	nine	nonagon
non	not	nonferrous, nonsense, nonabrasive, nondescript
nov	new	novel, renovate, novice, nova, innovate
nox, noc	night	nocturnal, equinox, noctilucent
numer	number	numeral, numeration, enumerate, innumerable
numisma	coin	numismatics
ob, oc, of, op	toward, against, in the way	oppose, occur, offer, obtain
oct	eight	octopus, octagon, octogenarian, octave
oligo	few, little	Oligocene, oligosaccharide, oligotrophic, oligarchy
omni	all, every	omnipotent, omniscient, omnipresent, omnivorous
onym	name	anonymous, pseudonym, antonym, synonym
oper	work	operate, cooperate, opus
-or	Noun: condition or activity	valor, honor, humor, minor
ortho	straight, correct	orthodox, orthodontist, orthopedic, unorthodox
-ory	Noun: place for, serves for	territory, rectory
-ous, -eous, -ose, -ious	Adjective: having the quality of, relating to	adventurous, courageous, verbose, fractious
over	excessive, above	overwork, overall, overwork
pac	peace	pacifist, pacify, pacific ocean

pair, pare	arrange, assemblage, two	repair, impair, compare, prepare
paleo	old	Paleozoic, Paleolithic, paleomagnetism, paleopsychology
pan	all	Pan-American, pan-African, panacea, pandemonium (place of all the demons),
para	beside	paradox, paraprofessional, paramedic, paraphrase, parachute
pat, pass, path	feel, suffer	patient, passion, sympathy, pathology
pater, patr	father	paternity, patriarch, patriot, patron, patronize
path, pathy	feeling, suffering	pathos, sympathy, antipathy, apathy, telepathy
ped, pod	foot	pedal, impede, pedestrian, centipede, tripod, podiatry, antipode, podium
pedo	child	orthopedic, pedagogue, pediatrics
pel, puls	drive, push, urge	compel, dispel, expel, repel, propel, pulse, impulse, pulsate, compulsory, expulsion, repulsive
pend, pens, pond	hang, weigh	pendant, pendulum, suspend, appendage, pensive, append
per	through, intensive	persecute, permit, perspire, perforate, persuade
peri	around	periscope, perimeter, perigee, periodontal
phage	eat	macrophage, bacteriophage
phan, phas, phen, fan, phant, fant	show, make visible	phantom, fantasy
phe	speak	blaspheme, cipher, phenomenon, philosopher
phil	love	philosopher, philanthropy, philharmonic, bibliophile
phlegma	inflammation	phlegm, phlegmatic
phobia, phobos	fear	phobia, claustrophobia, acrophobia, aquaphobia, ergophobia, homophobia
phon	sound	telephone, phonics, phonograph, phonetic, homophone, microphone, symphony, euphonious
phot, photo	light	photograph, photoelectric, photogenic, photosynthesis, photon
pico	trillionth	picofarad, picocurie, picovolt
pict	paint, show, draw	picture, depict
plac, plais	please	placid, placebo, placate, complacent
pli, ply	fold	reply, implicate, ply
plore	cry out, wail	implore, exploration, deploring
plu, plur, plus	more	plural, pluralist, plus
pneuma, pneumon	breath	pneumatic, pneumonia,
pod	foot, feet	podiatry, tripod
poli	city	metropolis, police, politics, Indianapolis, megalopolis, acropolis
poly	many	polytheist, polygon, polygamy, polymorphous
pon, pos, pound	place, put	postpone, component, opponent, proponent, expose, impose, deposit, posture, position, expound, impound
pop	people	population, populous, popular

port	carry	porter, portable, transport, report, export, import, support, transportation
portion	part, share	portion, proportion
post	after, behind	postpone, postdate
pot	power	potential, potentate, impotent
pre, pur	before	precede
prehendere	seize, grasp	apprehend, comprehend, comprehensive, prehensile
prin, prim, prime	first	primacy, prima donna, primitive, primary, primal, primeval, prince, principal
pro	for, foward	propel
proto	first	prototype, protocol, protagonist, protozoan, Proterozoic, protoindustrial
psych	mind, soul	psyche, psychiatry, psychology, psychosis
punct	point, dot	punctual, punctuation, puncture, acupuncture, punctuation
pute	think	dispute, computer
quat, quad	four	quadrangle, quadruplets
quint, penta	five	quintet, quintuplets, pentagon, pentane, pentameter
quip	ship	equip, equipment
quir, quis, quest, quer	seek, ask	query, inquire, exquisite, quest
re	back, again	report, realign, retract, revise, regain
reg, recti	straighten	regiment, regular, rectify, correct, direct, rectangle
retro	backwards	retrorocket, retrospect, retrogression, retroactive
ri, ridi, risi	laughter	deride, ridicule, ridiculous, derision, risible
rog, roga	ask	prerogative, interrogation, derogatory
rupt	break	rupture, interrupt, abrupt, disrupt, ruptible
sacr, sanc, secr	sacred	sacred, sacrosanct, sanction, consecrate, desecrate
salv, salu	safe, healthy	salvation, salvage, salutation
sanct	holy	sanctify, sanctuary, sanction, sanctimonious, sacrosanct
sat, satis	enough	satient, saturate, satisfy
sci, scio, scientia	know	science, conscious, omniscient, cognocienti
scope	see, watch	telescope, microscope, kaleidoscope, periscope, stethoscope
scrib, script	write	scribe, scribble, inscribe, describe, subscribe, prescribe, manuscript
se	apart, move away from	secede
sect, sec	cut	intersect, transect, dissect, secant, section
sed, sess, sid	sit	sediment, session, obsession, possess, preside, president, reside, subside

semi	half, partial	semifinal, semiconscious, semiannual, semimonthly, semicircle
sen, scen	old, grow old	senior, senator, senile, senescence, evanescent
sent, sens	feel, think	sentiment, consent, resent, dissent, sentimental, sense, sensation, sensitive, sensory, dissension
sept	seven	septet, septennial
sequ, secu, sue	follow	sequence, consequence, sequel, subsequent, prosecute, consecutive, second, ensue, pursue
serv	save, serve, keep	servant, service, subservient, servitude, preserve, conserve, reservation, deserve, conservation, observe
-ship	Noun: status, condition	relationship, friendship
sign, signi	sign, mark, seal	signal, signature, design, insignia, significant
simil, simul	like, resembling	similar, assimilate, simulate, simulacrum, simultaneous
sist, sta, stit	stand, withstand, make up	assist, insist, persist, circumstance, stamina, status, state, static, stable, stationary, substitute
soci	to join, companions	sociable, society
sol, solus	alone	solo, soliloquy, solitaire, solitude, solitary, isolate
solv, solu, solut	loosen, explain	solvent, solve, absolve, resolve, soluble, solution, resolution, resolute, dissolute, absolution
somn	sleep	insomnia, somnambulist
soph	wise	sophomore (wise fool), philosophy, sophisticated
spec, spect, spi, spic	look, see	specimen, specific, spectator, spectacle, aspect, speculate, inspect, respect, prospect, retrospective, introspective, expect, conspicuous
sper	render favorable	prosper
sphere	ball, sphere	sphere, stratosphere, hemisphere, spheroid
spir	breath	spirit, conspire, inspire, aspire, expire, perspire, respiration
stand, stant, stab, stat, stan, sti, sta, st, stead	stand	stature, establish, stance
-ster	person	mobster, monster
strain, strict, string, stige	bind, pull, draw tight	stringent, strict, restrict, constrict, restrain, boa constrictor
stru, struct, stroy, stry	build	construe, structure, construct, instruct, obstruct, destruction, destroy, industry, ministry
sub, suc, suf, sup, sur, sus	under, below, from, secretly, instead of	sustain, survive, support, suffice, succeed, submerge, submarine, substandard, subnormal, subvert

sume, sump	take, use, waste	consume, assume, sump, presumption
super, supra	over, above	superior, suprarenal, superscript, supernatural, superimpose, supercede
syn, sym	together, at the same time	sympathy, synthesis, synchronous, syndicate
tact, tang, tag, tig, ting	touch	tactile, contact, intact, intangible, tangible, contagious, contiguous, contingent
tain, ten, tent, tin	hold, keep, have	retain, continue, content, tenacious
tect, teg	cover	detect, protect, tegular, tegument
tele	distance, far, from afar	telephone, telegraph, telegram, telescope, television, telephoto, telecast, telepathy, telepathy
tem, tempo	time	tempo, temporary, extemporaneously, contemporary, pro tem, temporal
ten, tin, tain	hold	tenacious, tenant, tenure, untenable, detention, retentive, content, pertinent, continent, obstinate, contain, abstain, pertain, detain
tend, tent, tens	stretch, strain	tendency, extend, intend, contend, pretend, superintend, tender, extent, tension, pretense
tera	trillion	terabyte, teraflop
term	end, boundary, limit	exterminate, terminal
terr, terra	earth	terrain, terrarium, territory, terrestrial
test	to bear witness	testament, detest, testimony, attest, testify
the, theo	God, a god	monotheism, polytheism, atheism, theology
therm	heat	thermometer, theorem, thermal, thermos bottle, thermostat, hypothermia
thesis, thet	place, put	antithesis, hypothesis, synthesis, epithet
tire	draw, pull	attire, retire, entire
tom	cut	atom (not cutable), appendectomy, tonsillectomy, dichotomy, anatomy
tor, tors, tort	twist	torture, retort, extort, distort, contort, torsion, tortuous, torturous
tox	poison	toxic, intoxicate, antitoxin
tract, tra, trai, treat	drag, draw, pull	attract, tractor, traction, extract, retract, protract, detract, subtract, contract, intractable
trans	across, beyond, change	transform, transoceanic, transmit, transportation, transducer
tri	three	tripod, triangle, trinity, trilateral
trib	pay, bestow	tribute, contribute, attribute, retribution, tributary
tribute	give	contribute, distribute, tributary
turbo	disturb	turbulent, disturb, turbid, turmoil
typ	print	type, prototype, typical, typography, typewriter, typology, typify
ultima	last	ultimate, ultimatum
umber, umbraticum	shadow	umbra, penumbra, (take) umbrage, adumbrate

un	not, against, opposite	unceasing, unequal
uni	one	uniform, unilateral, universal, unity, unanimous, unite, unison, unicorn
-ure	Noun: act, condition, process, function	exposure, conjecture, measure
vac	empty	vacate, vacuum, evacuate, vacation, vacant, vacuous
vade	go	evade, invader
vale, vali, valu	strength, worth	equivalent, valiant, validity, evaluate, value, valor
veh, vect	to carry	vector, vehicle, convection, vehement
ven, vent	come	convene, intervene, venue, convenient, avenue, circumvent, invent, convent, venture, event, advent, prevent
ver, veri	TRUE	very, aver, verdict, verity, verify, verisimilitude
verb, verv	word	verify, veracity, verbalize, verve
vert, vers	turn, change	convert, revert, advertise, versatile, vertigo, invert, reversion, extravert, introvert, diversion, introvert, convertible, reverse, controversy
vi	way	viable, vibrate, vibrant
vic, vicis	change, substitute	vicarious, vicar, vicissitude
vict, vinc	conquer	victor, evict, convict, convince, invincible
vid, vis	see	video, evident, provide, providence, visible, revise, supervise, vista, visit, vision, review, indivisible
viv, vita, vivi	alive, life	revive, survive, vivid, vivacious, vitality, vivisection, vital, vitamins, revitalize
voc, voke	call	vocation, avocation, convocation, invocation, evoke, provoke, revoke, advocate, provocative, vocal
vol	will	malevolent, benevolent, volunteer, volition
volcan	fire	volcano, vulcanize, Vulcan
volv, volt, vol	turn about, roll	revolve, voluble, voluminous, convolution, revolt, evolution
vor	eat greedily	voracious, carnivorous, herbivorous, omnivorous, devour
-ward	Adverb: in a direction or manner	homeward
-wise	Adverb: in the manner of, with regard to	timewise, clockwise, bitwise
with	against	withhold, without, withdraw, forthwith
-y	Noun: state, condition, result of an activity	society, victory
-y	Adjective: marked by, having	hungry, angry, smeary, teary
zo	animal	zoo (zoological garden), zoology, zodiac, protozoan

Bibliography

[1] M. Tomasello, *Origins of human communication: A Focus on Infrastructure*, pp. 1–6. A Bradford Book, The MIT press, 2008.

[2] D. Diringer, *The Book Before Printing: Ancient, Medieval and Oriental*, p. 27. Courier Dover Publications, 1982.

[3] R. I. Damper, *Chapter 1: Learning About Speech from Data: Beyond NETtalk*, pp. 1–25. Kluwer Academic, 2001.

[4] T. V. Polykova, *Grapheme-to-Phoneme Conversion in the Era of Globalization*. PhD thesis, Universitat Politcnica de Catalunya, Barcelona, Spain, November 2014.

[5] U. D.Reichel and H. R. Pfitzinger, "Text preprocessing for speech synthesis," in *Proc. of TC-Star Speech-to-Speech Translation Workshop*, (Barcelona, Spain), 2006.

[6] L. Cherry and W. Vesterman, "Writing tools : the style and diction programs," *In 4.3 BSD UNIX System Documentation*, 1981.

[7] A. Mikheev, "Periods, capitalized words, etc.," *Computational Linguistics*, vol. 28, pp. 289–318, September 2002.

[8] M. D. Riley, "Some applications of tree-based modelling to speech and language," in *Proc. of HLT '89 Proceedings of the workshop on Speech and Natural Language*, (Stroudsburg, PA, USA), 1989.

[9] R. I. Damper, *Self-Learning and Connectionist Approaches to Text-Phoneme Conversion*, pp. 117–144. London: UCL Press, 1995.

[10] N. Chomsky and M. Halle, *The Sound Pattern of English*. New York: NY: Harper and Row, 1968.

[11] G. Miller, *Language and Speech*, p. 49. San Francisco: W.H. Freeman and Company, 1981.

[12] A. van den Bosch, T. Weijters, H. J. van de Herik, and W. Daelemans, "When small disjunts abound, try lazy learning," in *Proc. of the 7th Belgian-Dutch Conference on Machine Learning (BENELEARN-97)*, (Tilburg, Netherlands), 1997.

[13] R. L. Venezky, *A Study of English Spelling-to-sound Correspondences on Historical Principles*. Ann Arbor, MI: Ann Arbor Press, 1965.

[14] E. Carney, *A Survey of English Spelling*. London: UK: Routledge, 1994.

[15] E. B. Bilcu, *Text-To-Phoneme Mapping Using Neural Networks*. PhD thesis, Tampere University of Technology, Tampere, October 2008.

[16] K. U. Ogbureke, C. Peter, and B. C. Julie, "Hidden markov models with context-sensitive observations for grapheme-to-phoneme conversion," in *Proc. of Interspeech*, (Japan), 2010.

[17] H. Che, J. Tao, and S. Pan, "Letter-to-sound conversion using coupled hidden markov models for lexicon compression," in *Proc. of the Oriental COCOSDA*, (Macau, China), pp. 141–144, 20.

[18] S. Bartlett, G. Kondrak, and C. Cherry, "Automatic syllabification with structured svms for letter-to-phoneme conversion," in *Proc. of The Association for Computational Linguistics (ACL) with the Human Language Technology Conference (HLT)*, (Ohio, USA), pp. 568–576, 2008.

[19] S. Kheang, K. Katsurada, Y. Iribe, and T. Nitta, "Solving the phoneme conflict in grapheme-to-phoneme conversion using a two-stage neural network-based approach," *IEICE Transactions on Information and Systems*, vol. E97-D, pp. 901–910, April 2014.

[20] E. B. Bilcu and J. Astola, "Neural networks with random letter codes for text-to-phoneme mapping and small training dictionary," in *Proc. of The 14th European Signal Processing Conference (EUSIPCO)*, (Florence, Italy), September 2006.

[21] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, pp. 434–451, January 2008.

[22] S. F. Chen, "Conditional and joint models for grapheme-to-phoneme conversion," in *Proc. of The European Conference on Speech Communication and Technology*, (Geneva, Switzerland), p. 20332036, 2003.

[23] S. Jiampojamarn, A. Bhargava, and Q. Dou, "Directl: a language independent approach to transliteration," in *Proc. of ACL-IJCNLP Nammed Entities Workshop*, pp. 28–31, 2009.

[24] S. Jiampojamarn, K. Dwyer, S. Bergsma, A. Bhargava, Q. Dou, M. Y. Kim, and G. Kondrak, "Transliteration generation and mining with limited training resources," in *Proc. of The Named Entities Workshop (NEWS)*, (Sweden), pp. 39–47, 2010.

[25] S. Jiampojamarn, C. Cherry, and G. Kondrak, "Integrating joint n-gram features into a discriminative training framework," in *Proc. of the Annual Conference of the North American Chapter of the ACL*, (California), pp. 697–700, June 2010.

[26] 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," in *Proc. of Interspeech*, (Portland, Oregon), 2012.

[27] J. R. Novak, N. Minematsu, and K. Hirose, "Failure transitions for joint n-gram models and g2p conversion," in *Proc. of Interspeech*, 2013.

[28] D. Wang and S. King, "Letter-to-sound pronunciation prediction using conditional random fields," *IEEE Signal Processing Letters*, no. 2, pp. 122–125, 2011.

[29] P. Lehnen, S. Hahn, A. Guta, and H. Ney, "Incorporating alignments into conditional random fields for grapheme-to-phoneme conversion," in *Proc. of ICASSP*, 2011.

[30] S. Hahn, P. Vozila, and M. Bisani, "Comparison of grapheme-to-phoneme methods on large pronunciation dictionaries and lvcsr tasks," in *Proc. of Interspeech*, 2012.

[31] P. Lehnen, S. Hahn, V.-A. Guta, and H. Ney, "Hidden conditional random fields with m-to-n alignments for grapheme-to-phoneme conversion," in *Proc. of Interspeech*, (Portland), 2012.

[32] P. Lehnen, A. Allauzen, and T. Lavergne, "Structure learning in hidden conditional random fields for grapheme-to-phoneme conversion," in *Proc. of Interspeech*, 2013.

[33] K. Kubo, S. Sakti, G. Neubig, T. Toda, and S. Nakamura, "Grapheme-to-phoneme conversion based on adaptive regularization of weight vectors," in *Proc. of Interspeech*, (Lyon, France), pp. 1946–1950, 2013.

[34] K. Kubo, S. Sakti, G. Neubig, T. Toda, and S. Nakamura, "Narrow adaptive regularization of weights for grapheme-to-phoneme conversion," in *Proc. of ICASSP*, (Australia), pp. 2608–2612, 2014.

[35] K. Kubo, S. Sakti, G. Neubig, T. Toda, and S. Nakamura, "Structured soft margin confidence weighted learning for grapheme-to-phoneme conversion," in *Proc. of Interspeech*, (Singapore), pp. 1263–1267, September 2014.

[36] F. Malfrre, T. Dutoit, and P. Mertens, "Automatic prosody generation using suprasegmental unit selection," in *Proc. of 3rd European Speech Communication Association (ESCA)/COCOSDA International Workshop on Speech Synthesis*, (Jenolan Caves, Australia), 1998.

[37] M. Tatham, K. Morton, and E. Lewis, "Modelling speech prosodics for synthesis-perspectives and trials," in *IEEE Seminar on State of the Art in Speech Synthesis*, (London), 2000.

[38] G. Bailly, T. Barbe, and H.-D. Wang, "Automatic labelling of large prosodic databases: Tools, methodology and links with a text-to-speech system," in *Proc. of the ESCA Workshop on Speech Synthesis*, (Autrans, France), September 1990.

[39] A. W. Black and P. Taylor, "Assigning intonation elements and prosodic phrasing for english speech synthesis from high level linguistic input," in *Proc. of International Conference on Spoken Language Processing (ICSLP'94)*, (Yokohama, Japan), 1994.

[40] W. Daelemans, S. Gillis, and G. Durieux, "The acquisition of stress: A data-oriented approach," *Computational Linguistics*, vol. 20, no. 3, pp. 421–451, 1994.

[41] C. Wightman and N. Campbell, "Automatic labeling of prosodic structure," in *Technical Report TR-IT-0061, ATR Interpreting Telecommunications Laboratories*, (Kyoto, Japan), 1994.

[42] K. Kadowaki, T. Ishihara, N. Hojo, and H. Kameoka, "Speech prosody generation for text-to-speech synthesis based on generative model of f0 contours," in *Proc. of Interspeech*, 2009.

[43] Y. Nishigaki, S. Takamichi, T. Toda, G. Neubig, S. Sakti, and S. Nakamura, "Prosody-controllable hmm-based speech synthesis using speech input," in *Proc. of Machine Learning in Spoken Language Processing (MLSLP)*, (Fukushima, Japan), 2015.

[44] J. Makhoul, "Linear prediction: A tutorial review," in *Proc. of the IEEE 63(4)*, 1975.

[45] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*. Berlin, Germany: Springer-Verlag, 1976.

[46] A. Varga and F. Fallside, "A technique for using multipulse linear predictive speech synthesis in text-to-speech type systems," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, no. 4, pp. 586–587, 1987.

[47] F. Charpentier and M. Stella, "Diphone synthesis using an overlap-add technique for speech waveforms concatenation," in *Proc. of the IEEE International Conference on ICASSP*, (Tokyo, Japan), 1986.

[48] E. Moulines and F. Charpentier, "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones," *Speech Communication*, vol. 9, no. 5/6, pp. 313–324, 1990.

[49] M. Macchhi, J. Altom, D. Kahn, S. Singhal, and M. Spiegel, "Intelligibility as a function of speech coding method for template-based speech synthesis," in *Proc. of 3rd European Conference on Speech Communication and Technology*, (Berlin, Germany), 1993.

[50] N. K. Kim, W. K. Seong, and H. K. Kim, *Natural Language Dialog Systems and Intelligent Assistants: Lexicon Optimization for WFST-Based Speech Recognition Using Acoustic Distance Based Confusability Measure and G2P Conversion*, pp. 119–127. Springer International Publishing, 2015.

[51] M. Levy, "Computer-assisted language learning: Context and conceptualization," (Oxford: Clarendon), 1997.

[52] R. Prabhavalkar, J. Keshet, K. Livescu, and E. Fosler-Lussier, "Discriminative spoken term detection with limited data," in *Proc. of the Symposium on Machine Learning in Speech and Language Processing*, (Portland, Oregon, USA), September 2012.

[53] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddingtion, "Results of the 2006 spoken term detection evaluation," in *Proc. of Interspeech*, 2006.

[54] H. yi Lee, Y. Zhang, E. Chuangsuwanich, and J. Glass, "Graph-based re-ranking using acoustic feature similarity between search results for spoken term detection on low-resource languages," in *Proc. of Interspeech*, (Singapore), 2014.

[55] D. Can, E. Cooper, A. Sethy, C. White, B. Ramabhadran, and M. Saraclar, "Effect of pronunciations on oov queries in spoken term detection," in *Proc. of ICASSP*, 2009.

[56] C. Parada, A. Sethy, and B. Ramabhadran, "Query-by-example spoken term detection for oov terms," in *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ISRU)*, December 2009.

[57] K. Katsurada, K. Katsuura, K. Seng, Y. Iribe, and T. Nitta, "Using multiple speech recognition results to enhance std with suffix array on the ntcir-10 spokendoc-2 task," in *The 10th NTCIR Conference, SpokenDoc-02*, June 2013.

[58] S. Kheang, Y. Iribe, and T. Nitta, "Letter-to-phoneme conversion based on two-stage neural network focusing on letter and phoneme contexts," in *Proc. of Interspeech*, (Florence, Italy), pp. 1885–1888, 2011.

[59] M. J. Embrechts and F. Arcinegas, "Neural networks for text-to-speech phoneme recognition," in *Proc. of The IEEE International Conference on Systems, Man, and Cybernetics*, (Nashville, TN), 2000.

[60] S. Kheang, K. Katsurada, Y. Iribe, and T. Nitta, "New grapheme generation rules for two-stage model-based grapheme-to-phoneme conversion," *Journal of ICT Research and Applications*, vol. 8, no. 2, pp. 157–174, 2014.

[61] T. Dutoit, *An Introduction to Text-to-Speech Synthesis*. Springer Netherlands, 1997.

[62] H. S. Elovitz, R. W. Johnson, A. McHugh, and J. E. Shore, "Letter-to-sound rules for automatic translation of english text to phonetics," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 24, no. 6, pp. 446–459, 1976.

[63] W. Ainsworth, "A systems for converting english text into speech," *IEEE Transactions on Audio and Electro-acoustics*, vol. 21, no. 3, pp. 288–290, 1973.

[64] S. Hunnicut, "Phonological rules for a text-to-speech system," *American Journal of Computational Linguistics*, pp. 1–72, 1976.

[65] R. I. Damper, Y. Marchand, M. J. Adamson, and K. Gustafson, "Comparative evaluation of letter-to-sound conversion techniques for english text-to-speech synthesis," in *Proc. of the 3rd ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis*, 1998.

[66] E. L. Thomdike and I. Lorge, *The teacher's word book of 30,000 words*. Columbia University: New York: Teachers College, 1994.

[67] W. Daelemans, A. V. D. Bosch, and T. Weijters, "Igtree: Using trees for compression and classification in lazy learning algorithms," *Artificial Intelligence Review*, vol. 11, pp. 407–423, 1997.

[68] A. W. Black, K. Lenzo, and V. Pagel, "Issues in building general letter-to-sound rules," in *Proc. of the 3rd ESCA Workshop in Speech Synthesis*, 1998.

[69] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce english text," *Complex Systems*, vol. 1, 1987.

[70] P. Taylor, "Hidden markov models for grapheme to phoneme conversion," in *Proc. of Interspeech*, 2005.

[71] M. Bisani and H. Ney, "Investigations on joint-multigram models for grapheme-to-phoneme conversion," in *Proc. of the International Conference on Spoken Language Processing*, (Denver, CO, USA), pp. 105–108, 2002.

[72] Y. Marchand and R. I. Damper, "A multi-strategy approach to improving pronunciation by analogy," *Computational Linguistics*, vol. 26, pp. 195–219, June 2000.

[73] S. Jiampojamarn, *Grapheme-to-phoneme conversion and its application to transliteration.* PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2011.

[74] R. Damper and J. Eastmond, "Pronunciation by analogy: impact of implementational choices on performance," *Language and Speech*, vol. 40, no. 1, 1997.

[75] A. V. D. Bosch and S. Canisius, "Improved morpho-phonological sequence processing with constraint satisfaction inference," in *Proc. of the 8th meeting of the ACL Special Interest Group in Computational Phonology (SIGPON)*, 2006.

[76] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proc. of the IEEE*, 1989.

[77] M. Collins, "Discriminative training methods for hidden markov models: theory and experiements with perceptron algorithms," in *Proc. of the ACL-02 conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[78] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proc. of the 21st International Conference on Machine Learning (ICML)*, 2004.

[79] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. of 18th International Conference on Machine Learning (ICML)*, 2001.

[80] K. Kubo, S. Sakti, G. Neubig, T. Toda, and S. Nakamura, "Structured adaptive regularization of weight learning for a robust grapheme-to-phoneme conversion model," *IEICE Transactions on Information and Systems*, vol. E97-D, no. 6, pp. 1468–1476, 2014.

[81] N. McCulloch, M. Bedworth, and J. Bridle, "Netspeak-a re-implementation of nettalk," *Computer Speech and Language*, vol. 2, no. 3-4, pp. 289–302, 1987.

[82] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorihtm," *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1997.

[83] S. Deligne, F. Yvon, and F. Bimbot, "Variable-length sequence matching for phonetic transcription using joint multigrams," in *Proc. of ICASSP*, (Madrid), pp. 2243–2246, September 1995.

[84] G. Bakiri and T. G. Dietterich, *Chapter 2: Constructing high-accuracy letter-to-phoneme rules with machine learning*, pp. 27–44. Kluwer Academic, 2001.

[85] T. J. Sejnowski and C. R. Rosenberg, "Nettalk corpus: ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/," 1993.

[86] J. Bullinaria, "Text-to-phoneme alignment and mapping for speech technology: A neural networks approach," in *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, (San Jose, CA), pp. 625–632, 2011.

[87] M. Davel and E. Barnard, "Extracting pronunciation rules for phonemic variants," in *ISCA Tutorial and Research Workshop on Multilingual Speech and Language Processing*, 2006.

[88] M. Davel and E. Barnard, "Pronunciation prediction with default&refine," *Computer Speech and Language*, vol. 22, pp. 374–393, January 2008.

[89] K. Knill and S. Young, "Hidden markov models in speech and language processing," in *Corpus-Based Methods in Language and Speech Processing*, 1997.

[90] S. H. Parfitt and R. A. Sharman, "A bi-directional model of english pronunciation," in *Proc. of the 2nd European Conference on Speech Communication and Technology (Eurospeech)*, vol. 2, (Genova, Italy), 1991.

[91] R. E. Donavan and P. C. Woodland, "A hidden markov-model-based trainable speech synthesizer," *Computer Speech and Lanauge*, vol. 13, no. 3, pp. 223–241, 1999.

[92] F. Jelinek, *Statistical methods for speech recognition*. The MIT Press, 1997.

[93] M. D. Calms and G. Prennou, "Bdlex: a lexicon for spoken and written french," in *Proc. of the 1st International Conference on Lanauge Resources and Evaluation*, 1998.

[94] L. Galescu and J. Allen, "Bi-directional conversion between graphemes and phonemes using a joint n-gram model," in *Proc. of the 4th ISCA Speech Synthesis Workshop (SSW)*, (Perthshire, Scotland), September 2001.

[95] S. Jiampojamarn and G. Kondrak, "Online discriminative training for grapheme-to-phoneme conversion," in *Proc. of Interspeech*, (Brighton), September 2009.

[96] K. Crammer and Y. Singer, "Ultraconservative online algorithms for multiclass problems," *Journal of Machine Learning Research*, vol. 3, pp. 951–991, 2003.

[97] D. Caseiro, I. Trancoso, and L. Oliveira, "Grapheme-to-phone using finite-state transducers," in *Proc. of IEEE Workshop on Speech Synthesis*, 2002.

[98] R. Sproat, "Pmtools: A pronunciation modeling toolkit," in *Proc. of the 4th ISCA TRWSS*, 2001.

[99] D. Yang, P. Dixon, and S. Furui, "Rapid development of a g2p system based on wfst framework," in *Proc. of ASJ*, pp. 111–112, 2009.

[100] T. Mikolov, M. Karafiat, L. Burget, J. H. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. of Interspeech*, (Makuhari, Chiba, Japan), pp. 1045–1048, September 2010.

[101] 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 *Proc. of The NAACL HLT Student Research Workshop and Doctoral Consortium*, (Colorado), June 2009.

[102] V. Claveau, "Letter-to-phoneme conversion by inference of rewriting rules," in *Proc. of Interspeech*, (UK), September 2009.

[103] P. C. Bagshaw, "Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression," *Computer Speech & Language*, vol. 12, pp. 119–142, April 1998.

[104] S. Jiampojamarn, G. Kondrak, and T. Sherif, "Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion," in *Proc. of NAACL-HLT*, (Rochester, New York), April 2007.

[105] S. Jiampojamarn and G. Kondrak, "Letter-phoneme alignment: An exploration," in *Proc. of The 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, (Uppsala, Sweden), July 2010.

[106] M. Libossek and F. Schiel, "Syllable-based text-to-phoneme conversion for german," in *Proc. of ICSLP*, pp. 283–286, 2000.

[107] P. Yu and F. Seide, "A hybrid-word/phoneme-based approach for improved vocabulary-independent search in spontaneous speech," in *Proc. of Interspeech*, (Korea), 2004.

[108] Y. Furuya, S. Natori, H. Nishizaki, and Y. Sekiguchi, "Introduction of false detection control parameters in spoken term detection," in *Proc. of Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, (Hollywood, CA), December 2012.

[109] S. Nakagawa, K. Iwami, Y. Fujii, and K. Yamamoto, "A robust/fast spoken term detection method based on a syllable n-gram index with a distance metric," *Speech Communication*, vol. 55, pp. 470–485, March 2013.

[110] N. Kanda, K. Itoyama, and H. G. Okuno, "Multiple index combination for japanese spoken term detection with optimum index selection based on oov-region classifier," in *Proc. of ICASSP*, (Vancouver, BC), May 2013.

[111] N. Bertoldi, R. Zens, and M. Federico, "Speech translation by confusion network decoding," in *Proc. of ICASSP*, (Honolulu, HI), April 2007.

[112] S. Xie and Y. Liu, "Using confusion networks for speech summarization," in *Proc. of the Human Language Technologies: Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (USA), 2010.

[113] J. G. Ficus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding*, (Canada), pp. 347–354, 1997.

[114] T. Schlippe, W. Quaschningk, and T. Schultz, "Combining grapheme-to-phoneme converter outputs for enhanced pronunciation generation in low-resource scenarios," in *The 4th Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU)*, (St. Petersburg, Russia), May 2014.

[115] S. Kheang, K. Katsurada, Y. Iribe, and T. Nitta, "Model prioritization voting schemes for phoneme transition network-based grapheme-to-phoneme conversion," in *Proc. of the International Conference on Computer and Information Science and Technology (CIST'15)*, (Ottawa, Canada), May 2015.

[116] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. of Neural Information Processing Systems (NIPS)*, (Canada), pp. 3104–3112, July 2014.

[117] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. of ICML*, 2006.

[118] K. Kubo, H. Kawanami, H. Saruwatari, and K. Shikano, "Unconstrained many-to-many alignment for automatic pronunciation annotation," in *Proc. of APSIPA ASC*, (Xi'an, China), October 2011.

[119] Y. Iribe, T. Mori, K. Kasturada, and T. Nitta, "Pronunciation instruction using cg animation based on articulatory feature," in *Proc. of The 18th International Conference on Computers in Education (ICCE2010)*, (Putrajaya, Malaysia), pp. 501–508, 2010.

[120] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICCASP)*, 2015.

Acknowledgements

The research presented in this thesis has been carried out at the Kasturada laboratory of the Department of Computer Science and Engineering, Toyohashi University of Technology (TUT), in Toyohashi city, Aichi prefecture, Japan.

First and foremost, my sincerest gratitude goes to my supervisor, Assoc. Prof. Kouichi KATSURADA, for his endless patience, for continuous guidance and support, for giving me a privilege to work in a good atmosphere at the laboratory, and also for his unfailing interest to my research, accompanied by excellent ideas and comments.

Distinguished thanks are due to my former supervisor Prof. Tsuneo NITTA and Asst. Prof. Yurie IRIBE for their great helps, advices, encouragements and fruitful technical discussions during my master and doctoral degrees in Japan. I also want to thank Prof. Shigeru MASUYAMA and Prof. Junsei HORIKAWA for reviewing my thesis.

Specially, I really appreciate the Japan International Cooperation Agency (JICA) and AUN/SEED-Net program which have provided a good scholarship to help strengthening the human resources in the developing country like Cambodia. Without them, I would not have a second chance to come to Japan for my doctoral degree. I am also thankful to Amano Scholarship for providing the financial support during my last six months in Japan. Personally, I am so grateful to give special thanks to Ms. Yumiko YOSHII, Ms. Mikayo KURONO, Mr. Hiroshi MOCHIZUKI, Ms. Chihiro TAKENAGA, Mr. Hideki ITO, Ms. Junko SAIGO, Ms. Kiyo KAWABUCHI, and other current and former JICA staffs for their advices and kind supports since the beginning of my arrival in Japan.

The people from the Department of Computer Science and Engineering have greatly contributed to the great atmosphere and working conditions of which I took benefit. I would like to thank them all. Moreover, I am profoundly thankful to all my friends in Toyohashi University of Technology for providing me the friendly supports, for the unforgettable friendships and very nice time spent together. I would also like to thank all my Cambodian friends in both Cambodia and Japan for their warm friendships and kind supports during my stay in Japan. I will never ever forget all of them, of course.

Last but not least, I would like to express my deep gratitude from the bottom of my heart to my beloved parents, my beloved grandparents who have been living in a peaceful heaven, and everybody in my whole family, for every moment of my life, for encouraging me in my studies and giving me the freedom, advices, happiness and supports to become the person I am today. Without all of you, I could never achieved this goal of my life.
I LOVE YOU ALL VERY MUCH INDEED!

Toyohashi, JAPAN, 20 January 2016
Seng KHEANG