UNIVERSITY OF LORRAINE

UNIVERSITY OF THE BASQUE COUNTRY

MASTER THESIS

Deep Learning For Robust And Reliable Grapheme-To-Phoneme Converter

Author:
Sandipana DOWERAH

Supervisor:
Denis JOUVET
Vincent COLOTTE
Miguel COUCEIRO

A thesis submitted in fulfillment of the requirements for the degree of Erasmus Mundus Master of Science in Language and Communication Technology

Based on work done during an internship with the

Multispeech team, Loria, Inria
University of Lorraine

August 24, 2019

Declaration of Authorship

I, Sandipana DOWERAH, declare that this thesis titled, "Deep Learning For Robust And Reliable Grapheme-To-Phoneme Converter" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Aim for the high mark and you will hit it. No, not the first time, not the second time and maybe not the third. But keep on aiming and keep on shooting for only practice will make you perfect. Finally you'll hit the bull's-eye of success."

Annie Oakley

UNIVERSITY OF LORRAINE

Abstract

Multispeech team, Loria, Inria
University of Lorraine

Erasmus Mundus Master of Science in Language and Communication Technology

Deep Learning For Robust And Reliable Grapheme-To-Phoneme Converter

by Sandipana DOWERAH

Grapheme-to-phoneme (G2P) converters are critical modules for speech recognition and speech synthesis systems. Their role consists in generating the pronunciation of the words; that is converting a sequence of letters to a sequence of phonemes (sounds of the language). For speech synthesis, the G2P converter needs to produce the 'standard' pronunciation of the words, whereas for speech recognition applications, the G2P converter must produce the usual pronunciation variants of the words. Over the time, many different approaches have been elaborated for such tasks. This includes rule-based approach (Divay and Vitale, 1997) where the rules are manually defined and many data-driven approaches relying either on decision trees (Andersen et al., 1996), on joint-sequence models (Bisani and Ney, 2008), on hidden markov models (Taylor, 2005), on conditional random fields (Illina, Fohr, and Jouvet, 2011) and more recently on neural network approaches (Rao et al., 2015); (Yao and Zweig, 2015). All these approaches lead to good performance, especially on common words, but performance tends to degrade on proper names. Also, it was observed that when an approach is wrong on a given word, another one may be correct; and thus combining several approaches was useful in speech recognition (Jouvet, Fohr, and Illina, 2012). The proposed approach relies on a set of G2P converters that is combined to produce results that are more reliable than those provided by a single G2P converter. Such an idea is somewhat similar to the combination of speech recognition systems which is frequently used in speech transcription to obtain improved performances (Fiscus, 1997). The goal of the proposed study is to elaborate a reliable and robust approach for predicting the pronunciation of words for speech synthesis purpose.

The first approach involves analyzing the outputs of two G2P converters. The underlying idea is that when they produce identical results, we assume that the generated pronunciation is correct, whereas, when the pronunciation differs, a manual decision is applied to decide which pronunciation is correct. Besides training various G2P converters, experiments have been carried out to assess the above assumption. As it is impossible to involve human interactions in the text-to-speech synthesis process for a manual decision. In such a case, a fully automatic approach is proposed using deep neural networks. We proposed a novel approach of using ensemble learning with multiple generator adversarial network for a robust framework of choosing the best G2P conversion model output. The proposed approach relies on the application of a Recurrent Neural Network (RNN) based classifier to decide on the best combination (or best choice) in each case. With our approach of combining various G2P models, we have achieved better performance than each individual G2P model on the English dataset.

Acknowledgements

I am thankful to my supervisors Denis Jouvet and Vincent Colotte for their immense help during this whole internship period for sharing their knowledge, experience and wisdom. I would like to express my gratitude to Denis Jouvet for his continual support and guidance throughout the thesis work, he enriched my research experience. I am grateful to my local coordinators Miguel Couceiro and Maxime Amblard for their support and guidance throughout the academic year in Lorraine. I am thankful to my co-supervisor Inma Hernez from my first year University (University of the Basque Country).

I am indebted to my family for their continual support and encouragement ...

Contents

List of Figures

List of Tables

List of Abbreviations

G2P	**G**rapheme **t**o **P**honeme
TTS	**T**ext **T**o **S**peech
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
BLSTM	**B**idirectional **L**ong **S**hort **T**erm **M**emory
GAN	**G**enerative **A**dversarial **N**etwork
HMM	**H**idden **M**arkov **M**odel
WFST	**W**eighted **F**inite **S**tate **T**ransducer
EM	**E**xpectation **M**aximization
CRF	**C**onditional **R**andom **F**ields
CTC	**C**onnectionist **T**emporal **C**lassification
WER	**W**ord **E**rror **R**ate
PER	**P**hone **E**rror **R**ate

Dedicated to my loved ones…

Chapter 1

Introduction

The human communication system consists of mainly two forms: speech and written forms. The smallest unit of the writing system in any language is the orthographic character termed as grapheme and the smallest unit of the sound system describing how a word is pronounced is the phoneme. Over the years, with the evolution of computers, electronic media, etc., technological advancement has changed human communication as well as the thinking process. The technology has enabled the machines today to converse with their creators by endowing them with the power of "intelligence" or "understanding capacity" together with speech recognition and speech synthesis capabilities.

The knowledge of reading and understanding language is of utmost importance to make the speech technology a successful endeavour. It is indeed a difficult and time-consuming task to learn a language. To understand a word or learn to pronounce a word is not a difficult task for humans. But, this is not an easy task for computer application technologies. Therefore, phonological or phonetic knowledge is taken into account for the pronunciation quality of the machines.

As humans have the memorizing and understanding ability, it is natural to acquire the ability to learn the pronunciation rules of a specific language. Although it was like an impossible thing for machines to adapt this quality, the scientists had made this possible by creating the first computer-based speech synthesis system in the late 1950s, and the first sophisticated text-to-speech (TTS) system in 1968 (Polyàkova, 2014). And since then, the text-to-speech system has emerged as one of the interesting research areas by attracting many research scientists in the field of speech processing. One important part of a text-to-speech system is the grapheme-to-phoneme conversion.

The grapheme-to-phoneme conversion aims at finding the proper pronunciation of a given word. It can be defined as the task of converting a sequence of characters into a sequence of pronunciation symbols. For example, given the word book, the task is to output its pronunciation /buk/. The symbolic or phonemic representation of a language's orthographic word is of utmost importance in many Natural Language Processing applications. It plays a vital role in the text-to-speech system for proper pronunciation knowledge of a word enriching with information about emphasis, sentence structure or such other linguistic information. Automatic Speech Recognition system is another such application where the use of grapheme-to-phoneme conversion is used for obtaining knowledge about a language's word pronunciation for allowing a recognition component to distinguish as well as recognize individual words in continuous audio signals.

The speech technologies interface or connect two different forms of communication, i.e. the spoken form and the written form. As a result, these types of technologies need to model the relationship between the sound (phoneme) and the textual unit (grapheme). However, modeling the relationship between phoneme and grapheme directly is not trivial. And, the relationship between phoneme and grapheme depends on whether the language is shallow or deep. It is evident that there is no language where all the sounds are completely represented by its orthographic units (graphemes), but there are some languages where they have a close representation between its sounds and letters. For instance, Italian and Finnish have a shallow writing system representing the close correlation of its orthography with its sound system. Whereas, French and English have a deep orthography depicting the dissimilarity between spelling and pronunciation.

The complexity of French having a deep orthography can be seen in the case of pronouncing one sound or combination of sounds. One sound can be written in various ways but there is one specific way of pronouncing a particular vowel letter or combination of vowel letters. For instance, the letter [e] can be pronounced as /ɛ/ as in 'quel' and in 'femme' the first [e] is pronounced as /a/ and the second one is not pronounced. Whereas in English, the complexity stands from the occurrence of double graphemes and double phonemes. For French and English, the pronunciation may depend on the grammar category or the verb tense: in French "président" (Noun/Verb) and English "read" (present/past tense). However, this kind of difference is not in the scope of this thesis.

1.1 Thesis Objectives

The objective of the proposed study is to elaborate a reliable and robust approach for predicting the pronunciation of words for speech synthesis purpose. Our approach relies on a set of various grapheme-to-phoneme models for producing more reliable results compared to an individual model. We also studied different kinds of deep neural network architectures and methods for grapheme-to-phoneme conversion. We have proposed a novel approach of robust framework for the grapheme-to-phoneme conversion using ensemble learning with multiple generator adversarial network to choose the best model output. The idea is influenced by Ian Goodfellow's Generative Adversarial Network or GAN in short (Goodfellow et al., 2014). But, unlike GAN, our model has multiple generators which are different grapheme-to-phoneme models and a classifier instead of discriminator which will decide the best grapheme-to-phoneme model among them.

1.2 Thesis Overview

The structure of the thesis work is designed as follows: Chapter 2 is an overview of the grapheme-to-phoneme conversion methods; Chapter 3 gives a detailed description about the experimentation of various deep learning approaches that have been carried out in this work as well as an introduction to the terminology that has been used throughout this study; Chapter 4 describes the encoder-decoder framework for the conversion of grapheme-to-phoneme; Chapter 5 presents a detailed description of Ensemble learning and Multiple Generator Adversarial Network and how it has been implemented for the task of grapheme-to-phoneme conversion. In Chapter 6 we presented our experimental setup as well as evaluation and analyze the results obtained. And finally concluding remarks and directions for future work are given in Chapter 7 .

Chapter 2

An Overview of Grapheme-to-Phoneme Approaches

The conversion of letter-to-sound or grapheme-to-phoneme (G2P) goes back centuries and can be found in many of the older descriptive grammars of languages in the world. Many studies have been done for grapheme-to-phoneme conversion applying various approaches over the years. The automatic grapheme-to-phoneme conversion was first considered in the context of text-to-speech (TTS) systems. The input text needs to be converted into a sequence of pronunciation symbols after normalization to be used to control the speech synthesizer. Dictionary look-up being the simplest technique in this aspect has many limitations. It is difficult to make a pronunciation dictionary of significant size by hand as it is a tedious and costly task. The database can be problematic too; also a finite dictionary will have a limited coverage whereas TTS systems are expected to deal with arbitrary words. To overcome the limitations of dictionary look-up, rules were formed for developing grapheme-to-phoneme converters. In this chapter: Section 2.1 discusses rule-based approaches, Section 2.2 gives a detailed description of data-driven approaches, and finally Section 2.3 talks about neural network-based approaches.

2.1 Rule-based Approach

The rule-based systems were developed as a solution to overcome the single dictionary lookup. A rule-based method is based on a set of grapheme-to-phoneme rules (Elovitz et al., 1976). The approach requires some linguists for developing such a system. Such an approach is expensive as building the rules for some languages with complex writing-system is labor-intensive and time-consuming. Moreover, it is not feasible to define a set of rules for covering all the linguistic information of a language. As in the case of English, sometimes there is an ambiguous association of graphemes with phonemes, thus, resulting in the rules approaching the size of the lexicon which is undesirable (Kominek and Black, 2006).

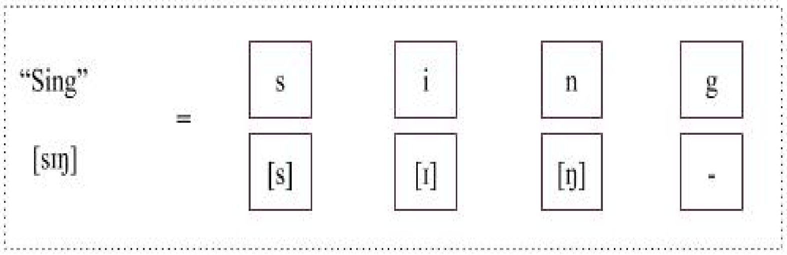
2.2 Data-driven Approach

In the data-driven method, the probabilistic relationship between grapheme and phoneme is learned through the data used as a set of words or expressions with their associated pronunciation. There are mainly two major steps in this type of method, the alignment step, and the phoneme generation step. The data-driven method requires the computation of alignment between the letters and the phonemic symbols (Jiampojamarn, Cherry, and Kondrak, 2008).

The alignment step can be viewed as one of the common processes in the G2P conversion approaches. The utilization of learning and inference methods differs the G2P approaches from one another. Different techniques have been proposed for the G2P conversion and among them, the local classification-based (Sejnowski and Rosenberg, 1987; Pagel, Lenzo, and Black, 1998) and probabilistic sequence-based (Taylor, 2005; Bisani and Ney, 2008) are the ones that gained wide attention.

2.2.1 Techniques based on Local classification

The various G2P approaches require or create an alignment of training data between letters and phonemes in a different preprocessing step. The alignment is established in a way that each alignment element contains exactly one letter. The corresponding phonemes can be zero or one or more than one. The alignment is termed as 1-to-n alignment (Bisani and Ney, 2008). For instance; the alignment for the word "sing" below:



Alignments can be created either manually (handcrafted rules) or by an iterative estimation of alignment probabilities. In general, the input sequence is processed sequentially, for instance, from left to right. A phoneme sequence is chosen from a small set of acceptable references for each input character. The prediction of the output phoneme is based on the context of the current grapheme. Therefore, it can be said that this is termed as local classification because the decision for each position is taken before moving to the next. In local classification-based techniques, a decision tree (Pagel, Lenzo, and Black, 1998) or a neural network (Sejnowski and Rosenberg, 1987) can be trained, given the alignments, to learn the G2P relationships from the trained data. These are the most common techniques used for doing the predictions. Although decisions taken locally about each phoneme is not optimal from the theoretical point of view, this strategy avoids the need for using a search algorithm which is necessary for a globally optimal solution. There are few works on G2P conversion where they have used neural networks methods for the classification problem. (Sejnowski and Rosenberg, 1987) as well as (McCulloch, Bedworth, and Bridle, 1987) have applied neural networks for the classification problem. In their approach, they have used a three-layer network for the classification problem.

2.2.2 Techniques based on Probabilistic based sequence modelling

Another approach that several researchers have applied for carrying out the G2P conversion task is the probabilistic based sequence modeling approach. In the probabilistic approach, the prediction of the next phoneme is based on the symmetric window of graphemes and left-sided window of phonemes. It can be done by creating 1-to-n alignments of the training data by using a context-independent channel model.

There have been various G2P models based on the probabilistic sequence modeling approach, the below sections gives an overview on some of these approaches;

Hidden Markov Model (HMM) based approach

The G2P task has been formulated in the standard HMM way by applying independent and identical distribution and first-order Markov Model assumptions by (Taylor, 2005). It is expressed formally as;

$$\begin{aligned} S^* &= argmaxP(S,G) \\ &= argmaxP(G|S)P(S) \\ &= argmax\prod_n P(g_n|s_n)P(s_n|s_{n-1}) \end{aligned} \tag{2.1}$$

Where $S = [s_1, ..., s_n, ..., s_N]$ represents the hidden sequence of phoneme and $G = [g_1, ..., g_n, \ldots, g_N]$ denotes the sequence of grapheme observations. In this framework, each HMM represents a phoneme which emits up to four grapheme symbols. Unlike the local classification approach where alignments are obtained as a pre-processing step, the alignments can be obtained during the training of the HMM with the Baum-Welch algorithm. As for the inference, the Viterbi algorithm is used for obtaining the most probable phoneme sequence from the input grapheme sequence.

Joint Multigram approach

The idea of the joint multigram approach is that the input-output relationship can be generated from a common sequence of joint units thus carrying both input and output symbols. The term multigram is used when the units carry multiple input and output symbols and can also be termed as a joint sequence (Deligne, Yvon, and Bimbot, 1995). The joint multigram approach is based on the proposition of graphones. A graphone is a pair of a sequence of graphemes and a sequence of phonemes. The joint probability of a sequence of graphemes G and a sequence of phonemes S are obtained from sequences of graphones Q. It can be formally expressed as;

$$P(S,G) = \sum_{Q \in S(S,G)} p(Q) \tag{2.2}$$

An n-gram approximation can be used for modeling the probability distribution over all matching alignments (graphone sequences). With the use of expectation maximization (EM) algorithm, the parameters of the n-gram model are learned by maximizing the log-likelihood of the data (Bisani and Ney, 2008). Another approach of obtaining the best sequence of phonemes is by using a weighted finite-state transducer (WFST) framework (Novak, Minematsu, and Hirose, 2012).

Sequitur, based on joint multigram framework and phonetisaurus, based on WFST framework are two of the open-source tools for conversion of G2P that have been widely used until now. These tools have been often considered as baseline models while researching the G2P conversion task.

Conditional random fields based approach

Conditional random fields (CRF) are a popular probabilistic approach for discriminative modeling. It has shown that CRFs are well suited for segmenting and labeling sequential data (Lafferty, McCallum, and Pereira, 2001). In conditional random fields approach, the conditional probability is modeled using a log-linear representation. Since CRF is a discriminative model, it can perform global inference, thus, exploiting the advantages of both decision tree-based methods and joint multigram methods. But, it can be more expensive computationally compared to the aforementioned approaches. The parameters of the log-linear CRF model are learned by maximizing the conditional log-likelihood. The Viterbi algorithm is used for decoding the best phoneme sequence.

Acoustic data-driven approach

There are few attempts made in using the acoustic data for extracting the pronunciation for incorporating the G2P conversion process. One of the reasons for interest in this approach is grapheme-based recognition systems (Killer, Stüker, and Schultz, 2003). Rasipuram and Magimai-Doss (Rasipuram and Magimai-Doss, 2012) presented the G2P conversion approach by capturing the relationship between grapheme and phoneme from acoustic data using Kullbeck-Leibler divergence based HMM model system for speech recognition or TTS system. The acoustic data-driven approach based on the Kullback-Leibler divergence framework is used to capture the sequence information in the orthographic transcription to infer the phoneme sequences and its variants for the grapheme-to-phoneme conversion (Rasipuram and Magimai-Doss, 2012).

2.3 Neural Networks based Approach

The artificial neural networks have gained interest long back for the task of G2P conversion. One such network that has attracted immense attention is the error backpropagation network because of its ability to learn the mapping between two sets of patterns. Sejnowski and Rosenberg (Sejnowski and Rosenberg, 1987) used a backpropagation network for mapping the grapheme of American English text with its phonetic transcription as part of the text-to-speech system and named their network as NETtalk (Sejnowski and Rosenberg, 1987). They use a three-layer neural network. The input of the network is a context window of plus/minus three letters. The input layer uses an orthogonal representation which means one input for each type of letter. Finally, the output layer represents the predicted phoneme through articulatory features (McCulloch, Bedworth, and Bridle, 1987).

An interesting approach was proposed by F. Arciniegas and M.J. Embrechts using staged neural networks in 2000 (Arciniegas and Embrechts, 2000). They applied three neural networks for the G2P task. The first network distinguishes the single and dual phoneme cases. In the single phoneme case, one letter is mapped to one phoneme and in dual phoneme case, one letter is mapped to two phonemes. For both the single and dual case the networks from the second stage are trained separately. By implementing this approach, they attained quite a good phoneme accuracy but as they increase the size of the dictionary the accuracy starts decreasing.

(Rao et al., 2015) implemented unidirectional LSTMs with different forms of output delays for the task of G2P conversion. In this approach, the use of various forms of output delays enables the model to see several graphemes before outputting any phoneme (Rao et al., 2015). Another approach they have carried out in their work is the implementation of BLSTM with connectionist temporal classification (CTC) (Graves, Mohamed, and Hinton, 2013) for G2P conversion task. The objective of implementing CTC is to avoid the need for explicit alignment before training. BLSTM with a CTC layer of 512 units gives the best result. Combining this model with a traditional 5-gram WFST model provided some improvement as well. A BLSTM model with encoder-decoder architecture with a side-conditioned language model used to perform G2P task without explicit alignment (Yao and Zweig, 2015).

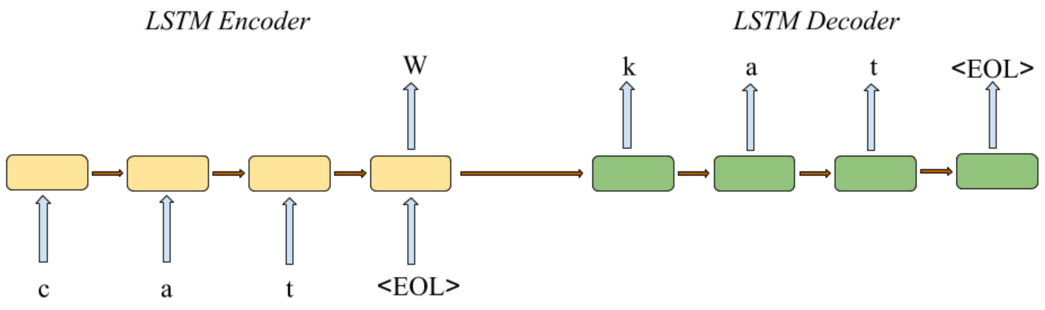


FIGURE 2.1: *The input of the encoder is "cat" grapheme sequence and the decoder produces "k a t" as the phoneme sequences. The left side is the encoder and the right side is the decoder. The model stops making predictions after generating the "end-of-line (EOL)" tag.*

Another G2P conversion task was being experimented by applying attention function with the encoder-decoder framework (Shubham Toshniwal, 2016). An attention function is the mapping of a query and a set of key values to output, where the query, as well as key values along with the outputs, are all vectors (Vaswani et al., 2017). In the attention enabled encoder-decoder model, the model jointly learn to align and convert characters to phonemes in contrast to the G2P models that require explicit alignments. With this type of attention model, the dependency on an external aligner can be removed. The applicability of attention with encoder-decoder framework achieves the state-of-the-art on few datasets. The use of global attention for the G2P task is comparatively a reasonable choice given the short sequence length for the G2P conversion.

The use of convolutional neural networks with residual connections as encoder and BLSTM as decoder outperformed most of the previous solutions on the CMU-Dict and NetTalk datasets in terms of phoneme error rate (PER) (Yolchuyeva, Németh, and Gyires-Tóth, 2019).

With the day-to-day advancement in the neural networks, various other approaches have been applied for the task of G2P conversion. A recent approach has achieved the new state-of-the-art in the G2P conversion task with transformer architecture. The transformer is a novel neural network architecture based on self-attention mechanism. The transformer network is used with knowledge distillation for the G2P task. In knowledge distillation, a light student model can approximate the accuracy of a heavy teacher model (Hinton, Vinyals, and Dean, 2015). The proposed approach of transformer network with knowledge distillation for unlabeled data has acquired the new state-of-the-art result in the G2P conversion task (Sun et al., 2019).

Chapter 3

Deep Learning for Robust and Reliable Grapheme-to-Phoneme Converter

Deep learning has gained immense popularity in the last few years. Recently, with the advancement of deep learning, the grapheme-to-phoneme conversion is viewed as a sequence-to-sequence task and modeled with deep neural networks architectures. In this chapter, we discussed the implementation of deep learning for grapheme-to-phoneme conversion task. As the focus of our work is to build a robust system for providing the best grapheme-to-phoneme conversion model, we examined several neural networks architecture for our task. This chapter is organized as: Section 3.1 gives a brief overview of the evolution of deep learning into research and reviews the implementation of deep learning for our task. Then, in Section 3.2, we discuss the use of attention mechanism and its variants for sequence attention task, Section 3.3 describes the sequence-to-sequence encoder-decoder framework adapted for our task and lastly Section 3.4 describes the transformer network.

3.1 Deep Learning

Deep learning is a field of machine learning based on the learning algorithms inspired by the function of the human brain called artificial neural networks. Artificial neural networks are an approach to computational learning designed to solve various problems in pattern recognition, prediction, optimization, and control. The first paper relevant to artificial neural networks was published by (McCulloch and Pitts, 1943). An attempt for the first time was made at a mathematical-algorithmic description of the signal processing behavior of neurons in the human brain. Research into the replication of these neural approaches to machine learning began in the late 1950s (Rosenblatt, 1958). The structure of an artificial neural network is a network of nodes adjoined to each other by weighted connections. The nodes represent the neurons and the weights represent the strength of the synapses between the neurons (Graves, 2012). Over the years many varieties of artificial neural networks have appeared with varied properties. One such distinction of artificial neural networks are the ones whose connections form cycles and are referred to as feedback, recursive or recurrent neural networks. The ones without the cycles are known as feedforward neural networks.

Though the initial results of artificial neural networks were promising, it resulted as a failure towards the end of the 1960s. The reasons for the failure are not covered in this thesis. However, in the 1980s neural networks started to regain its interest again due to the advent of approaches such as improved backpropagation algorithm for training hidden layers in neural networks (Cortes and Vapnik, 1995). But, again the flow of neural networks experienced stagnation during the mid-1990s because of the limitation of computational resources. It was during the 2000s, the field of research experienced an influential and fruitful surge of neural networks and is continuing.

3.1.1 Feedforward neural network

Feedforward neural network is an artificial neural network to approximate some function $f*$. For instance, for a classifier, $y = f*(x)$ maps an input x to a category y. A feedforward network defines a mapping $y = f(x;)$ and learns the value of the parameters θ that results in the best function approximation.

In a feedforward network, information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. The outputs of the model cannot be fed back into itself as there are no feedback connections. Feedforward networks are multilayer perceptrons.

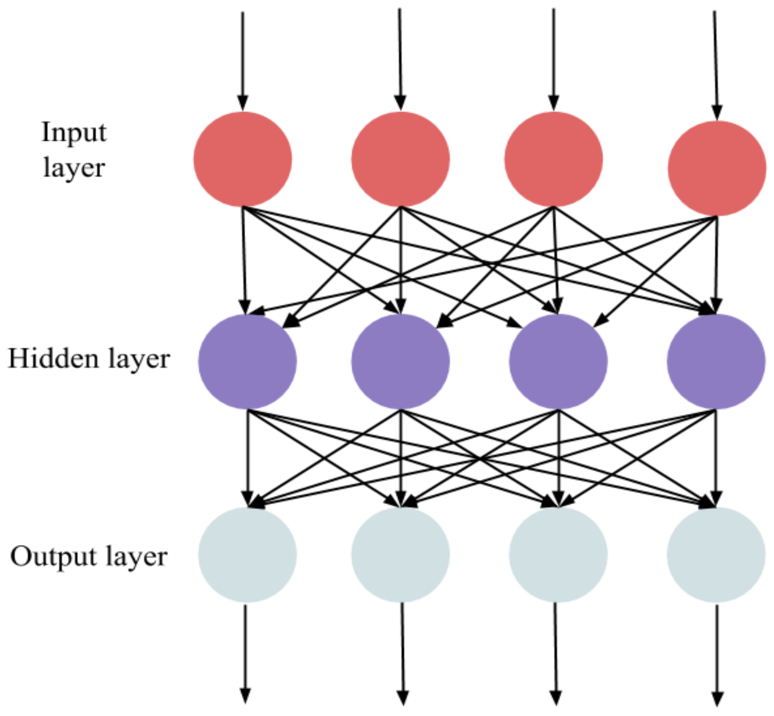


FIGURE 3.1: *Feed forward neural network.*

In Figure 3.1, there are three layers called input layer, hidden layer and output layer. Normally, all nodes of a single layer have the same properties like activation function and type like input, hidden and output.

Sigmoid function

The sigmoid activation function is a very common choice for feedforward neural networks used for binary classification. The sigmoid function is often used in the neural network to introduce non-linearity to the model. The sigmoid function takes a single real value as an input and returns a real-valued output. The output range is from 0 to 1. It's also called the logistic function.

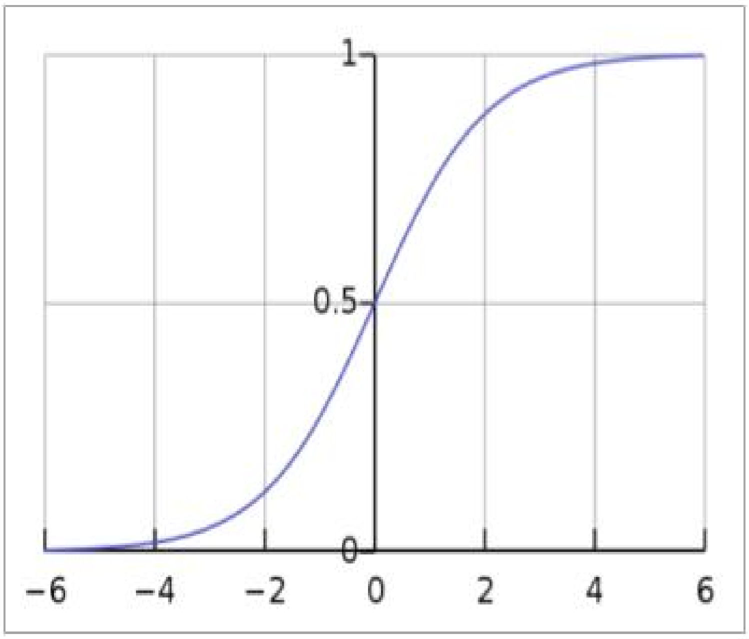


FIGURE 3.2: *Sigmoid activation function (**source: Internet**).*

$$f(s_i) = \frac{1}{1 + e{-}^{s_i}} \tag{3.1}$$

Cross-entropy loss

The cross-entropy loss between two probability distributions over the same set of events estimates an average number of bits required to detect if the encoding scheme used is optimized for target distribution rather than true distribution. In neural network architecture for a task of binary classification, the last (output) layer is the sigmoid activation function. For neural network-based classifier training, we opted for cross-entropy as the loss function.

3.1.2 Recurrent neural network

A recurrent neural network or RNN is a part of neural network family for processing sequential data; introduced in the 80s for modeling time series (Rumelhart, 1986). The recurrent neural network is a powerful and simple network that is specialized in processing a sequence of values $(x_1, ..., x_T)$, RNNs operating on a sequence that contains $x(t)$ vectors with the time step index t, ranging from 1 to T). The recurrent neural network can not only take their input from the current input example but also the previous inputs. RNN maps previous inputs from the entire history to each output. The recurrent connections of an RNN allow memorizing previous inputs in the networks internal state, that can further be used for influencing the network output. RNN is specially designed for sequence learning tasks by forming recurrently connected networks, in which hidden networks and input-output networks are connected cyclically to each other (Graves, 2012).

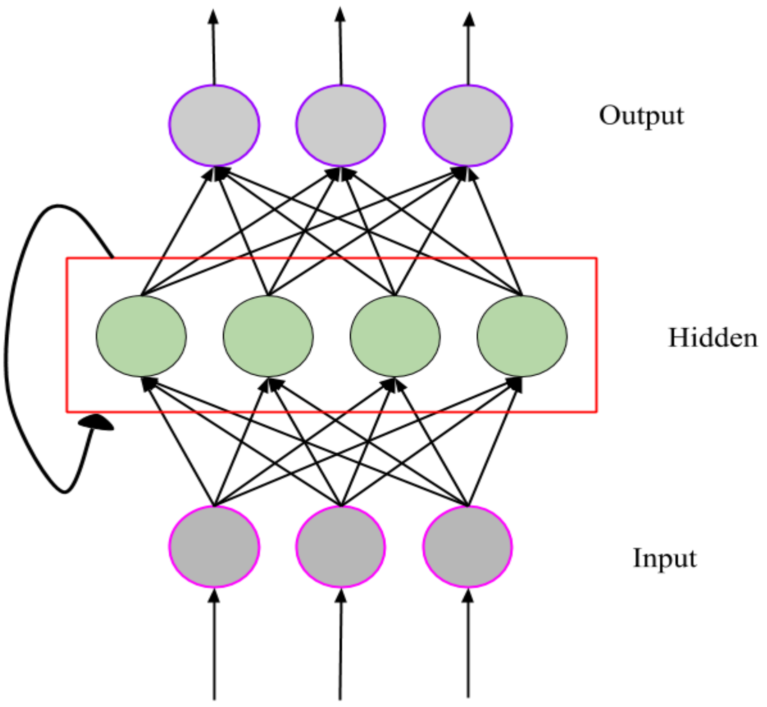


FIGURE 3.3: *Schematic view of a recurrent neural network with self-connected hidden layer. The recurrent connections in the hidden layer allow information to persist from one time input to the next one.*

The following equations 3.2, 3.3 are iterated over time, which is, from t=1 to $t = T$ to calculate intermediate hidden representation as $h = (h_1, h_2 \ldots h_T)$, for a single hidden layer neural network.

$$h_t = f(w_{xh} \cdot x_t + W_{hh} \cdot h_t - 1 + b_h) \tag{3.2}$$

$$y_t = w_{hy} \cdot h_t + b_y \tag{3.3}$$

In Equations 3.2and 3.3, W_{xh} is the weight matrix between input vector x and hidden representation in hidden layer h; W_{hh} is the weight matrix between hidden layer at time $t-1$ and hidden layer at time t and W_{hy} is the weight matrix between hidden layer and output layer; $f(.)$ is non linear activation function; b_h and b_y are the bias vectors for hidden layer and output layer. The ability of processing current, as well as previous context inputs, makes RNN well suited for sequence classification tasks where context within the sequence is useful. Sequence classification is a predictive modeling problem where there is a sequence of inputs over space or time and the task is to predict a category for the sequence. RNNs can build a dynamic temporal context window instead of a fixed context window as they store activations from previous steps in their internal state.

Long short-term memory (LSTM)

There are mainly two issues while training recurrent neural networks. The vanishing gradient problem and the exploding gradient problem (Bengio, Simard, and Frasconi, 1994; Hochreiter, 1991). The problem arises as the network tries to learn long-term dependencies during training which either decays or blows up exponentially as it cycles around the network's recurrent connections. Several attempts have been made to solve the problem of gradient descent in RNN in the 1990s. (Bengio, Simard, and Frasconi, 1994) tried to solve the gradient descent problem with simulated annealing (SA) and discrete error propagation, with the introduction of time delays by (Lang, Waibel, and Hinton, 1990; Lin, Sontag, and Wang, 1996), and with hierarchical sequence compression by (Schmidhuber, 1992) were some of the other notable attempts made in regard to solve the problem of gradient descent. However, long short-term memory or LSTM proposed by Hochreiter and Schmidhuber in 1997 (Hochreiter and Schmidhuber, 1997), proved to be the most effectively successful solution for the gradient descent so far.

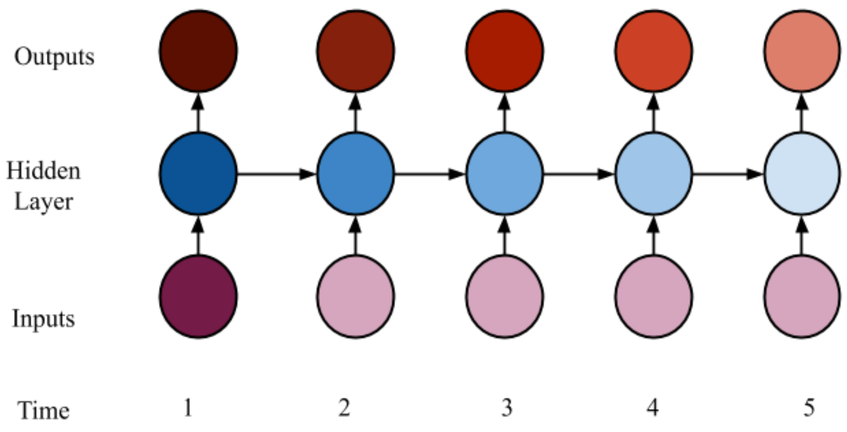


FIGURE 3.4: *Graphical representation of vanishing gradient descent problem in RNN. (**Graves, 2012**).*

Figure 3.4 is a graphical representation of the vanishing gradient problem in RNN. The shades of the nodes indicate the sensitivity over time of the network nodes to the input at time-step 1. It is harder for the network to update the weights if the gradient is low, thus, resulting in the delay to the final results. By the different shades, it means, the lighter the shade, the lower the gradient. The network eventually 'forgets' the first input as the new inputs overwrite the activation of the hidden unit.

LSTMs are designed to be able to avoid the problem of long-term dependencies. One of the main properties of LSTMs is that they can hold onto any information for a long time. They specifically consist of memory blocks, i.e. a set of recurrently connected subnets. LSTMs are framed with the ability to remove or add information to the cell state, termed as gates. The vanishing gradient problem is averted with the multiplicative gates, the input, output and forget gates. The analogs read, write and reset of these gates allow LSTM memory cells to store and access information for a long time, thus, averting the issue of gradient descent. For instance, until the input gate is closed, the new inputs will not overwrite the activation of the cell and by opening the output gate the new inputs can be made available later in the sequence to the network. Forget gate has been enhanced to the LSTM architecture with the ability to forget the previous inputs when necessary. For the capability of dealing with sequential data, LSTMs are a good approach for the task of grapheme-to-phoneme conversion. LSTMs with encoder-decoder framework performed quite well (Sutskever, Vinyals, and Le, 2014). The LSTM encoder encoded the sequence of graphemes and the other LSTM decodes the phoneme sequences.

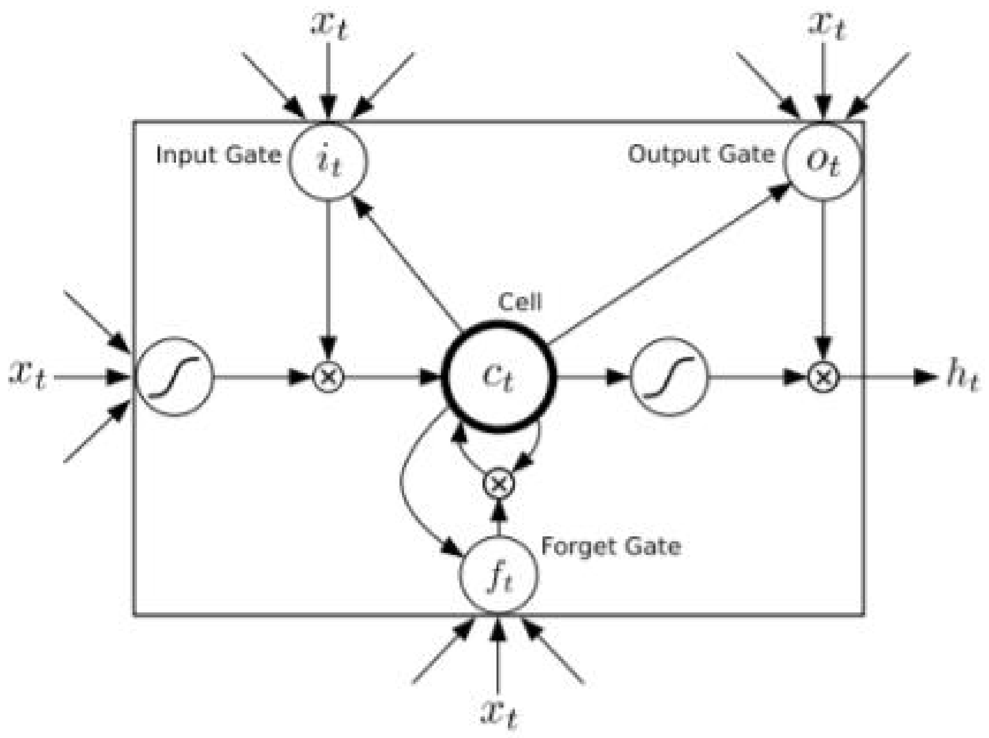


FIGURE 3.5: *Schematic view of Long short-term memory cell (**Graves, Mohamed, and Hinton, 2013**).*

The above diagram 3.5, has an input gate, output gate and forget gate. The three gates collect activations from both outside and inside the block. The small circles denote the multiplicative units. The arrow indicates the flow of information.

$$i_t = \sigma(W_{xi} \cdot x_t + W_{hi} \cdot h_{t-1} + W_{ci} \cdot c_{t-1} + b_i) \tag{3.4}$$

$$f_t = \sigma(W_{xf} \cdot x_t + W_{hf} \cdot h_{t-1} + W_{cf} \cdot c_{t-1} + b_f) \tag{3.5}$$

$$o_t = \sigma(W_{xo} \cdot x_t + W_{ho} \cdot h_{t-1} + W_{co} \cdot c_{t-1} + b_o) \tag{3.6}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot tanh(W_{xc} \cdot x_t + W_{hc} \cdot h_{t-1} + b_c) \tag{3.7}$$

$$h_t = o_t \cdot tanh(c_t) \tag{3.8}$$

where i_t , f_t, o_t are activations of the input gate, forget gate and output gate, c_t is cell state or cell memory. σ is the sigmoid function acting as non linear function.

Bi-directional long short-term memory (BLSTM)

In Figure 3.6, the input sequence x maps to the output sequence y. The forward arrow (towards the right) propagates the forward information on time and the backward arrow (towards the left) propagates information backward.

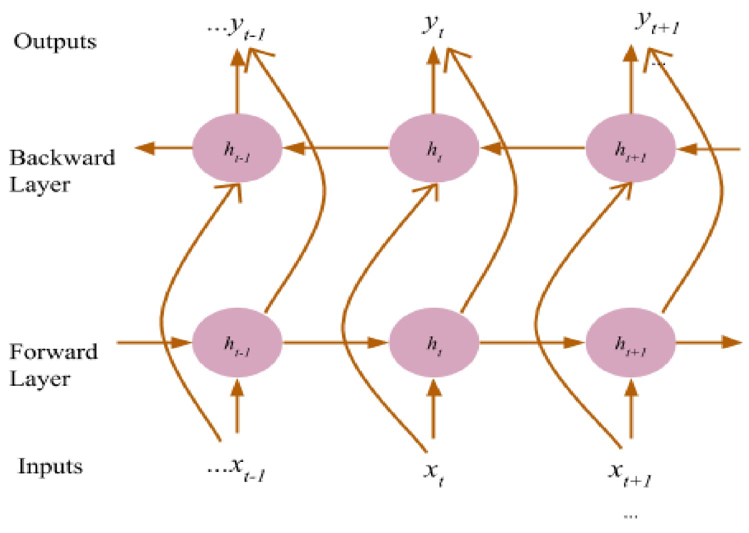


FIGURE 3.6: *Bi-directional Long short-term memory(**Graves, Mohamed, and Hinton, 2013**).*

The RNN architectures that have been discussed so far take into account the information from the past input at time step $x_1, ..., x_{t-1}$ as well as current input x_t. Suppose, in the case of speech recognition systems, due to coarticulation in languages like English, the sound of the current phoneme depends on the following phoneme sounds. Bi-directional LSTMs (BLSTM) have been introduced for processing such issues (Schuster and Paliwal, 1997). BLSTM based architectures shown improved results in applications like handwriting recognition (Graves et al., 2009), speech recognition (Graves et al., 2009; Graves, Mohamed, and Hinton, 2013).

Bi-directional LSTM or BLSTM, in short, processes the input sequence in both directions with two sub-layers in order to account for the full input context. The two sub-layers compute forward and backward hidden sequences and the two hidden sequences are then combined to compute the output sequences (Graves, Mohamed, and Hinton, 2013). By taking into account the information from the past as well as from future, it acquires the information on the whole sequence of inputs in the network. BLSTMs have proved to be very efficient for sequential tasks. (Mousa and Schuller, 2016) used complex many-to-many alignments with BLSTM for G2P conversion and achieved improved results on the publicly available CMU dictionary.

Chapter 4

Encoder-decoder framework for Grapheme-to- Phoneme Conversion

Encoder-decoder is a neural networks architecture proposed by (Cho et al., 2014) for sequence-to- sequence learning. In the encoder-decoder networks architecture, two networks are used: an encoder that encodes the input sequence into a compressed embedding vector and the other network decoder decodes the input sequence by using the dense representation of input sequence's data and at each time-step produce output. The output from the previous time-step is then used for producing an output sequence. In this chapter, we will first analyze the sequence-to-sequence model for the G2P conversion in section 4.1 and section 4.2 narrates the attention mechanism and finally, section 4.3 describes briefly the transformer network.

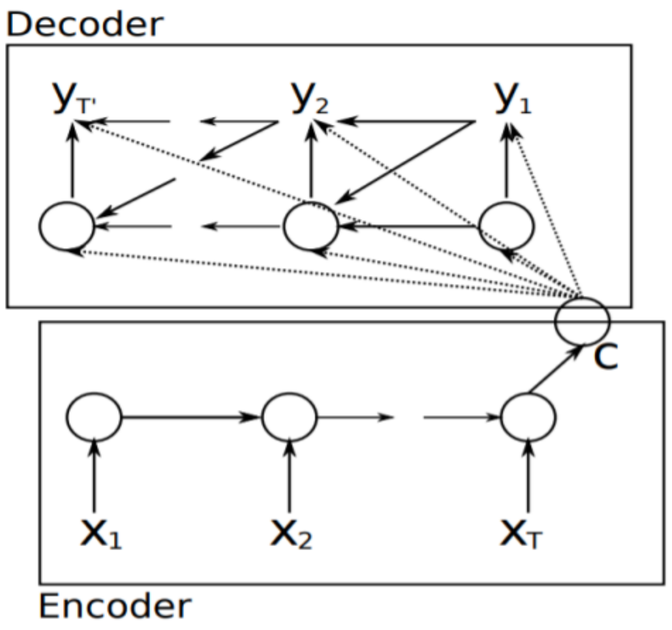


FIGURE 4.1: *RNN encoder-decoder framework* (**Cho et al., 2014**).

4.1 Sequence-to-sequence Architecture

In OpenNMT the encoder-decoder framework is used for machine translation. The input is a source word in one language and the output is the target word in another language. *"NMT takes a conditional language modeling view of translation by modeling the probability of a target sentence $w_{1:T}$ given a source sentence $x_{1:S}$ as $p(w_{1:T}|x) = \Pi_1^T p(w_t|w_{1:t-1}, x; \theta)$ where the distribution is parameterized with θ.(Klein et al., 2017)"*. In the sequence-to-sequence model architecture of OpenNMT, the source encoder which is a recurrent neural network model maps each source word to a word vector and process them to a sequence of hidden vectors. The target decoder combines the hidden representation of previously generated words with source hidden vectors for predicting the scores for each possible next word. *"The default decoder applies attention over the source sequence and implements input feeding by default" (OpenNMT.net)* [1].

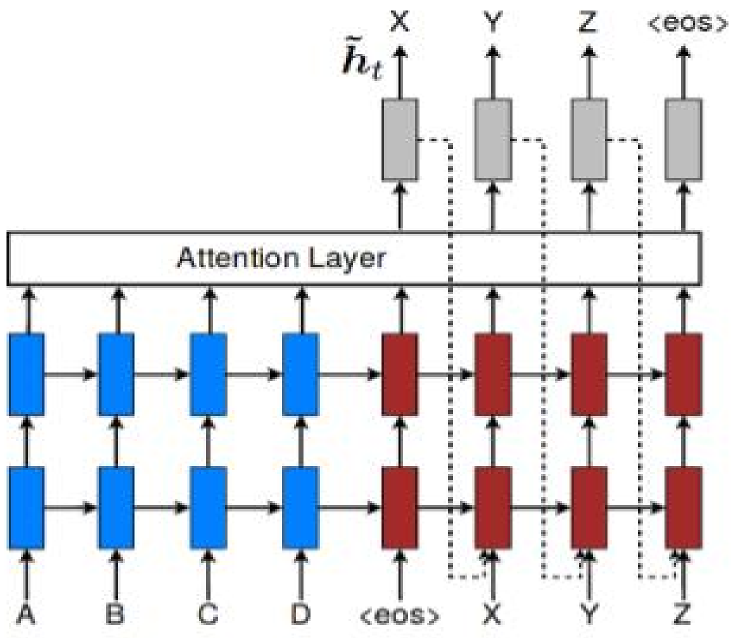


FIGURE 4.2: *Encoder-decoder architecture of OpenNMT (**Luong, Pham, and Manning, 2015**).*

Figure 4.2, is an attention-based encoder-decoder network. In this figure, the encoder states are in blue and decoder states are in red, through an attention layer, jointly fed into an output layer represented by color grey.

Adaptation of sequence-to-sequence architecture for G2P

We describe briefly the general framework of RNN encoder-decoder in this section that has been adapted from the OpenNMT proposed by (Klein et al., 2017). For the grapheme-to-phoneme task, we modified the data preprocessing step used for neural machine translation task. We provided the source input as a sequence of graphemes and the target output as a sequence of phonemes. In the encoder-decoder framework, the encoder reads the input character sequences and mapped them to the character vectors $x = (x_1, ..., x_{Tx})$. The most common approach is to use an RNN such that $h_t = f(x_t, h_{t-1})$ and $c = q(h_1, ..., h_{Tx})$, where $h_t \in R^n$ is a hidden state at time t, and c is a vector generated from the sequence of the hidden states, f and q are some non-linear functions. The decoder is often trained to predict the next character y_t, given the character vector c and all the previously predicted characters

[1]http://opennmt.net/

$y_1, ..., y_{t-1}$. The decoder defines a probability over the conversion y by decomposing the joint probability into the ordered conditionals. Thus, with a RNN decoder each conditional probability is modelled as, $p(y_t|y_1, ..., y_{t-1}, c) = g(y_{t-1}, s_t, c)$, where g is a non-linear, potentially multi-layered function that outputs the probability of y_t, and s_t is the hidden state of the RNN.

In our adaptation, the source character sequences are first mapped to character vectors and then fed into an RNN. Upon seeing the <eos> (end of sentence as used in the OpenNMT library) symbol, the final time step initializes a target. At each target time step, attention is applied over the source RNN and combined with the current hidden state to produce a prediction $(w_t|w_{1:t-1}, x)$ of the next character. This prediction is then fed back into the target RNN. A source encoder RNN maps each source character to a character vector, and process these to a sequence of hidden vectors $(h_1, ..., h_s)$. The target decoder combines an RNN hidden representation of previously generated vectors $(w_1, ..., w_{t-1})$with source hidden vectors to predict scores for each possible next pronunciation symbol.

4.2 Attention Mechanism

Attention mechanism has been introduced in the context of neural machine translation (Bahdanau, Cho, and Bengio, 2014). As neural machine translation is being designed within the framework of encoder-decoder, the encoder encodes a source input into a fixed-length context vector from which the decoder generates the target output. The translation models require alignment of the symbol positions between the source and the target. In the case of long sentences the translation model often has difficulty in memorizing the whole sentence. It forgets the first part while processing the whole input sentence. Attention mechanism was developed mainly to deal with the long source sentences in neural machine translation. Attention creates shortcuts between the context vector and the whole input which makes it easy for the network to have access to the whole input sequence and stops it from forgetting any part of it. The context vector, thus, learned the alignment between the source and the target. With the successful implementation of attention mechanism in neural translation, it has been used in other natural language processing applications for training the neural networks architecture in recent times (Luong, Pham, and Manning, 2015; Vaswani et al., 2017).

In the words of (Vaswani et al., 2017), *"An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key"*.

To compute the output, let's assume that there is a score function a which measures the similarity between the query and a key. Then we compute all n scores $a_1, ..., a_n$ by

$$a_i = \alpha(q, k_i) \tag{4.1}$$

Then, softmax is used to obtain the attention weights

$$b_1, \dots, b_n = softmax(a_1, ..., a_n) \tag{4.2}$$

The output is then a weighted sum of the values

$$o = \Sigma_{i=1}^{n} b_i v_i \tag{4.3}$$

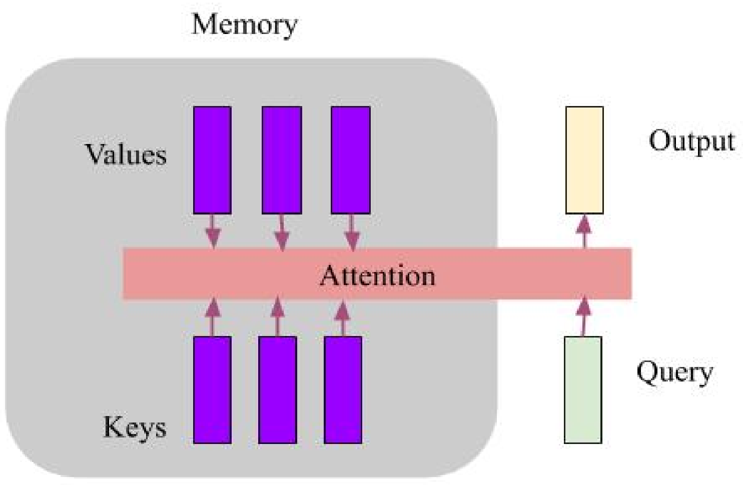


FIGURE 4.3: *The attention layer returns an output based on the input query and its memory (**Zhang et al., 2019**).*

Global attention

(Luong, Pham, and Manning, 2015) extended the attention mechanism proposed by (Bahdanau, Cho, and Bengio, 2014) for neural machine translation by simplifying it and named it as global attention. The idea of a global attentional model is to consider all the hidden states of the encoder when deriving the context vector c_t. In this model type, at each time step t, the model infers a variable-length alignment vector a_t based on the current target hidden state h_t with each source hidden state h_s. A global context vector c_t is then computed as the weighted average, according to a_t, overall the source states.

There are different models available from "Global attention model" by (Luong, Pham, and Manning, 2015) in the OpenNMT library [2]. We have applied the general attention model, one of the alternatives of global attention, the score function is:

$$score(h_t, h_s) = h_t^T W_a h_s \tag{4.4}$$

Where h_t is the current target hidden state; h_s is the source hidden state and W_a is the weighted average. Given the alignment vector as weights, the context vector is computed as the weighted average over all the source hidden states (Luong, Pham, and Manning, 2015).

As we have used only the global attention, the other variants of the attention mechanism, i.e. local attention is not discussed in this thesis.

4.3 Transformer Network

Google introduced the transformer network in 2017 (Vaswani et al., 2017). The transformer network has also the encoder-decoder framework. However, the architecture

[2]`http://opennmt.net/OpenNMT-py/Library.html`

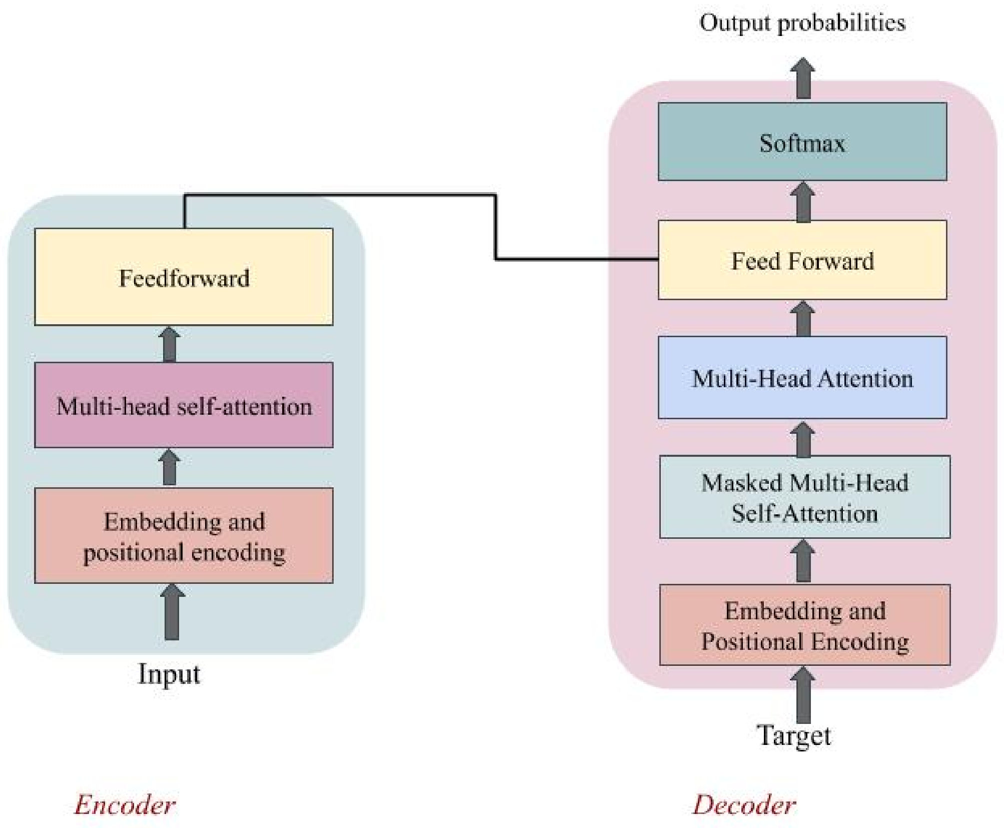

Output probabilities
Softmax
Feed Forward
Multi-Head Attention
Masked Multi-Head Self-Attention
Embedding and Positional Encoding
Target
Feedforward
Multi-head self-attention
Embedding and positional encoding
Input
Encoder
Decoder

FIGURE 4.4: *Simplified graphical representation of the transformer model.*

of the transformer network is based on a self-attention mechanism. Transformer network emerged to deal with the sequential task as well. The transformer transforms one sequence into another within the framework of encoder and decoder; but unlike the sequence-to-sequence model with the encoder-decoder framework, transformer performs without using sequence-aligned recurrent architecture.

Transformer's architecture completely relies on the attention mechanism. Figure 4.4 is a graphical representation of the architecture of transformer. The encoder is on the left side and the decoder is on the right. The transformer can be seen mainly composed of multi-head attention and feed forward layers. Input and output (target) sequences are projected into an embedded vector space. Positional encoding is added to the embedded input vector for capturing the token position within the sequence. Multi-head self-attention computes multiple attention blocks over the source, concatenates them and projects them linearly back onto space with the initial dimensionality. The decoder operates similarly, but generates one phoneme at a time, from left to right and is composed of five stages. Multi-head attention not only attends to the past items but also the final representations generated by the encoder. Finally, a softmax layer to map target symbol scores into target symbol probabilities.

Multi-head attention

"Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions" (Vaswani et al., 2017).

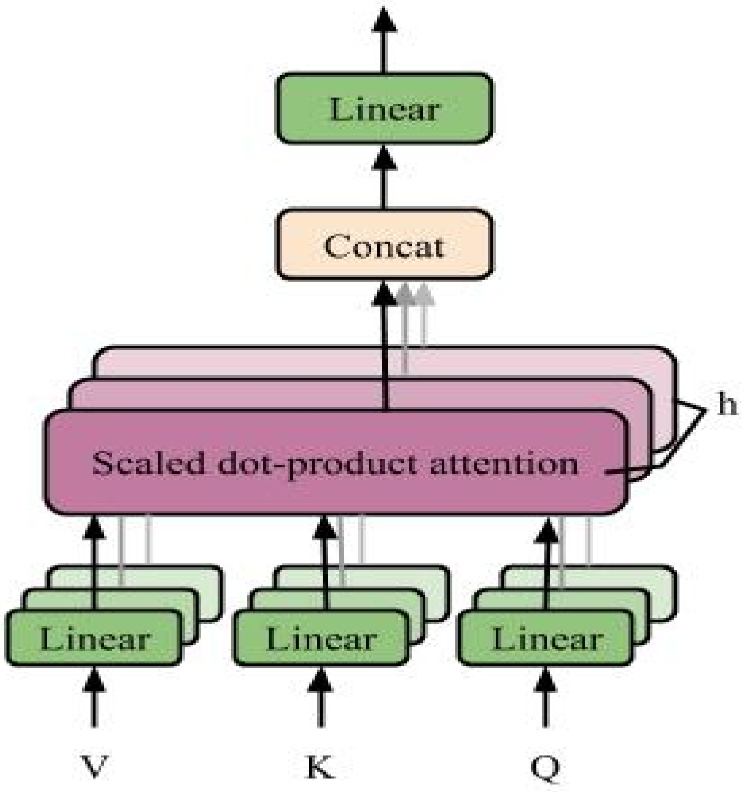


FIGURE 4.5: *Multi-Head Attention consists of several attention layers running in parallel* (**Vaswani et al., 2017**).

The encoder-decoder framework of the Transformer model with the use of stacked multi-head attention mechanism showed successful results in many machine learning tasks, such as machine translation, image caption, etc. (Vaswani et al., 2017) stated that multi-head attention is more efficient in using the model's capacity comparing to the same size model with single-head attention.

Multi-head attention computes attention more than once. A multi-head attention layer consists of h parallel attention layers, each one is called a head. For each head, any number of dense layers with hidden sizes use to project the queries, keys, and values, respectively, before feeding into the attention layer. The outputs of these h heads are concatenated and then projected by another dense layer as shown in Figure 4.5.

The multi-head attention relies on scaled dot-product attention, which operates on a query Q, a key K and value V:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{dk}})V \tag{4.5}$$

where dk is the key dimensionality. In self- attention, queries, keys, and values come from the output of the previous layer. After computing the dot-product of the query with all the keys; the query and the keys are then divided by the $\sqrt{dk}$. A softmax function is then applied to obtain the weights on the values.

The multi-head attention obtains h i.e. one per head, different representations of Q, K, V. It then computes scaled dot-product attention for each representation, concatenates the outputs and projects the concatenation through a feed-forward layer.

It can be expressed as:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{4.6}$$

$$Multihead(Q, K, V) = Concat_i(head_i)W^O \tag{4.7}$$

where W_i and W^O are parameter matrices.

The implementation and the results of these architectures will be discussed in chapter 6.

Chapter 5

Ensemble Learning with Multiple Generator Adversarial Network

In this chapter, we have discussed our approach of implementing ensemble learning with multiple generator adversarial network for selecting the best grapheme-to-phoneme model. The chapter is designed as; Section 5.1 briefly describes the ensemble learning and how it has been used in our work. Then, Section 5.2 gives a brief description of the multiple generator adversarial network. Section 5.3 gives a comprehensive description of our model and finally section 5.4 detailed about the architecture of our classifier.

5.1 Ensemble Learning

Ensemble learning can be seen used successfully for machine learning applications. It has been shown that multiple model combination resulted in improving the performance (Rokach, 2010). In statistical machine translation (SMT), ensemble learning can be applied to combine several systems as well (Och and Ney, 2002; Matusov, Ueffing, and Ney, 2006). The ensemble of models has also been applied to character-based sequential neural machine translation (Ling et al., 2015). To our knowledge, ensemble learning has not been used widely in the context of grapheme-to-phoneme conversion so far. The ensemble of various neural network models is applied with knowledge distillation on unlabeled data for grapheme-to-phoneme conversion only by (Sun et al., 2019) until now.

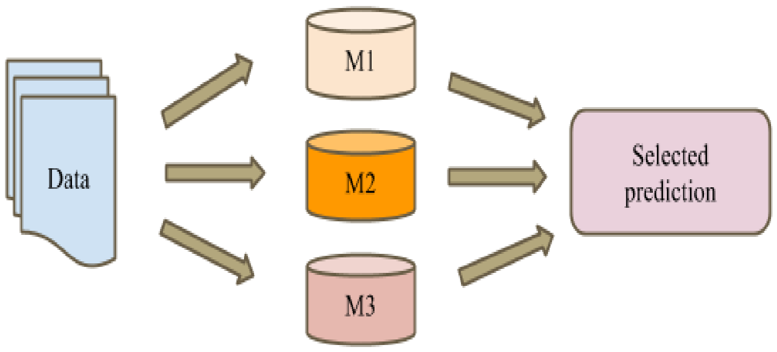


FIGURE 5.1: *Schematic view of ensemble learning. The three models M1, M2 and M3 are trained for solving the same problem. The idea is to combine the individual predictions of the three models in order to obtain the best predictive performance.*

Ensemble learning is a machine learning technique that combines several base models to produce one optimal selected prediction. The method of combining multiple models predicts better performance than a single predictive model and resulted in improving the performance of classification and prediction of a model. The different models are trained on the same dataset to make predictions by each of them individually. The individual predictions made by each model is then combined to make a final prediction.

5.2 Multiple Generator Adversarial Network

Training of multiple generators is not a new thing in image processing. In image processing, a novel approach of using multiple generators has been proposed by (Hoang et al., 2018) for addressing the issue of model collapsing. The use of multiple generators is highly influenced by Ian Goodfellow's generative adversarial network or GAN in short. In 2014 (Goodfellow et al., 2014) proposed the framework of adversarial nets for generative models. As we have not used generative adversarial networks in our work, so in this case, the chapter is entitled to multiple generators adversarial networks. Although, the idea of implementing the framework of adversarial nets is being influenced by (Hoang et al., 2018).

5.3 Model Description

The proposed approach is to create an adversarial network with multiple generators and a classifier to choose the best model output. Our approach relies on several multiple grapheme-to-phoneme conversion models which are used as generators. We have implemented an LSTM based classifier whose main task is to predict the best model output. Figure 5.2 below presents the graphical representation of our model.

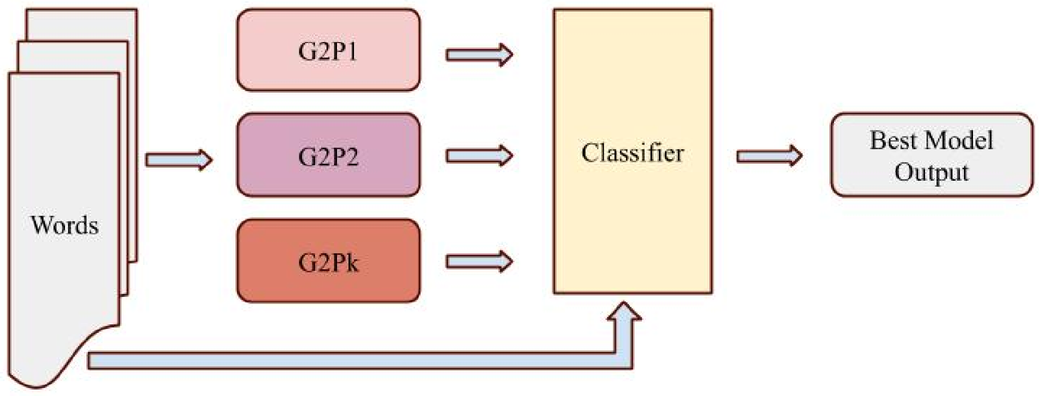


FIGURE 5.2: *Multiple generator network with ensemble learning for best model output.*

In diagram 5.2 above, the classifier will select the best model output from various grapheme-to-phoneme models depicting as generators. The input is the grapheme sequences $(x_1, ..., x_n)$ and the output is the pronunciation sequences $(z_1, ..., z_n)$. The pronunciation sequences (outputs) will then be given to the classifier along with the grapheme sequence (input). The task of the classifier can be termed as the sequence classification. Sequence classification is a predictive modeling problem where there is some sequence of inputs over space or time and the task is to predict a category for the sequence.

For our classifier, we will rely on a neural network approach. In this work, we have developed an RNN LSTM based classifier for addressing the classification problem.

5.4 Architecture of the Classifier

The input of our classifier is the phoneme sequences which are the outputs of the various grapheme-to-phoneme conversion models along with the input grapheme sequences. Sequence classification is a predictive modeling problem with the varied length of input entries. The model faces difficulty in reading the long-term context as well as dependencies between input sequences due to the varied length of the input. We developed an RNN LSTM based classifier for dealing with the sequence classification problems. The LSTM architecture of the RNN is designed to deal with long-term dependencies and RNN is well suited for studying contextual information. Figure 5.3 presents the graphical view of our classifier which is discussed in detail in the following sections.

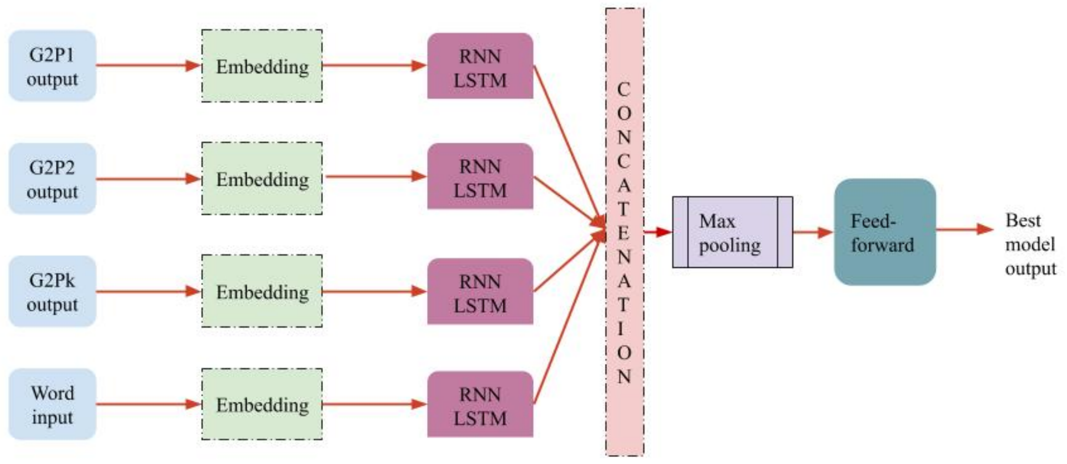


FIGURE 5.3: *Graphical representation of the RNN LSTM classifier.*

5.4.1 Embedding

We map the pronunciation sequence produced by the G2P models to a non-linear representation of phoneme identity using the embedding matrix. An embedding matrix is a list of all letters and their corresponding embeddings. The concept of an embedding matrix is an attempt to solve the relationship of representation problem. So, we will pick a dimensionality for the length of the pronunciation symbols (embeddings). Suppose, we have 50 pronunciation symbols, so we will use 50 dimensions of embedding. Therefore, all the pronunciations will be mapped to some point in this 50 dimension space.

5.4.2 RNN LSTM

We implement an RNN LSTM based architecture to processes sequential output phoneme embeddings and transform it into a fixed dimensional output vector. After representing each pronunciation symbol by its corresponding vector with the embedding matrix, the sequence of pronunciation symbols is given as input to the RNN LSTM one by one in a sequence.

The given character sequence can be considered as token sequences $s = s_1, ..., s_n$. The LSTM unit takes the embedding x_t of each token s_t as input and outputs a hidden state h_t. After the LSTM unit finished recurrent computation along with all the tokens from left to right we get the hidden state sequence $h = h_1, ...h_n$. The hidden state sequence h_t does not only capture the information of tokens s_t but also that of its predecessors.

The final representation of all the tokens i.e. $h = h_1, ...h_n$ is generated by concatenating all the hidden states across all the G2P models as well as the character input $s = s_1, ..., s_n$.

5.4.3 Max pooling layer

Max pooling is a widely used RNN structure for classification. Max pooling selects the max value of each position in all hidden state vectors. That implies the value of the $i'th$ position in vector h is calculated as:

$$h_i^{maxpooling} = max(h_{ji}) 1 \leq j \leq m \tag{5.1}$$

The phoneme sequences, as well as grapheme sequences across the G2P models, might have varying lengths. So, to have a fixed length of sequences we use max pooling. Also, max pooling layer helps to reduce the overfitting of the model.

5.4.4 Feed forward layer

The output of the max pooling layer is given to the feed forward layer as input. We designed the feed forward layer in such a way that the output dimension of feed forward is equal to the number of models we are combining with the classifier.

Furthermore, we used the Sigmoid layer as an activation function. To avoid the problem of overfitting of the model we used dropout layer in the feed forward neural network.

Chapter 6

Experimentation and Results

We have proposed a novel approach of a robust framework for G2P conversion. We have experimented with two methods, i.e. multigram approach (sequitur [1] and phonetisaurus [2]) and neural networks approach (sequence-to-sequence) at first and opted for a more automated approach by implementing ensemble learning with multiple generator adversarial network. This chapter investigates the approaches that we have discussed in Chapter 3, 4 and Chapter 5 to verify the effectiveness of our models. There are mainly three major parts that are discussed in this chapter. First, we give a detailed description of the experimental setup in Section 6.1, where we introduce the datasets used for conducting our experiments, the training procedures, and the implementation methods. Secondly, in Section 6.2, we present the training and evaluation procedures and finally in Section 6.3, we report the results of our methods and give a brief analysis of our observations.

6.1 Experimental Setup

We carried out all our experiments on English and French datasets. First, we experimented the individual models for the G2P conversion followed by analysis on the same predictions by different G2P models and then we applied our method of combining models for the G2P task. We report the performance of our experimentation and visualize the results obtained.

6.1.1 Datasets

We use two datasets for evaluating the G2P approaches, the CMUdict for English [3] and CMU pronunciation dictionary for French [4]. For both datasets, we use the same train (80%), validation (10%) and test (10%) split using our write-up code. The data has been split in such a way that the variants of pronunciations for any given word are always grouped into either the train, or the validation, or the test set. The number of pronunciation variants in English dataset is 7846 (6.43%) and the number of pronunciation variants in French dataset is 42652 (40.61%). The size of the phoneme vocabulary for English is 69 and grapheme vocabulary is 51 and similarly, for French phoneme vocabulary is 35 and grapheme vocabulary is 72 including punctuations, stress marks, umlauts.

[1] `https://github.com/sequitur-g2p/sequitur-g2p`

[2] `https://github.com/AdolfVonKleist/Phonetisaurus`

[3] (`http://www.speech.cs.cmu.edu/cgi-bin/cmudict`)

[4] (`https://sourceforge.net/projects/cmusphinx/files/Acoustic$%20and%20Language%20Models/French/$`

TABLE 6.1: Data split.

Language	Train (80%)	Validation (10%)	Test (10%)	Total Words	Unique Words
English	97555	12197	12196	121954	114108
French	83929	10521	10553	105003	62351

6.1.2 Model Configuration

We use the implmentation of sequitur mentioned in (Bisani and Ney, 2008). For phonetisaurus WFST driven G2P version 1.8, OpenFST based many to many aligner used prior to training G2P system, language model toolkit MITLM used. The rest of the configurations used are same as mentioned in the toolkit [5].

Sequence-to-sequence (seq2seq)

We adapted the sequence- to-sequence architecture from Open NMT to implement on our task. We train the sequence-to-sequence model with various model structures. LSTM, as well as BLSTM, has been trained with attention for our task. We use general attention (Luong, Pham, and Manning, 2015) with LSTM and BLSTM. We also vary the layers for both the LSTM and BLSTM encoder and decoder (2-2, 4-4) to analyze the impact on accuracy. We set the size of the hidden states for encoder and decoder to the same value (500 hidden RNN states). We use the same configurations as LSTM in the BLSTM model.

Transformer Model

We train the transformer model for the G2P conversion task as well. For both the encoder and the decoder, we use depth 2. The number of attention head is 10 and the embedding network dimension is 500 for both the encoder and the decoder network. The dropout is same as well for both the network, i.e. 0.3. The multihead dimension with a hidden dimension is 500 and the positionwise embedding is 2048. We use layer normalization of 500 hidden dimension. The last layer of the decoder is softmax.

Although we implemented the transformer model, the could not achieve any performance. Therefore, we have not included any results in the report.

Classifier

We use the embedding output dimension 50 in both the phoneme and the grapheme embedding network. Afterward, we give these embeddings from model1, model2, and grapheme to the respective BLSTM network of 4 layers with 50 hidden units. The outputs of these 3 BLSTM networks are then concatenated and given to the max pooling layer of output dimension 50 hidden units. Thereupon, the output of max pooling layer is given to the feed forward neural network whose output is equal to the number of models combining in the ensemble classifier. We use sigmoid as activation unit in the last layer.

We use the same configurations when we combined all the four models (seq2seq, BLSTM, sequitur, phonetisaurus) altogether. We have four phoneme embedding network and four BLSTM network each corresponding to the individual G2P model.

[5]https://github.com/AdolfVonKleist/Phonetisaurus

6.2 Training

For experimenting, we have relied on multiple G2P conversion models. We have considered sequitur and phonetisaurus as the baseline model for carrying out the research task. They both rely on statistical modeling of spelling and pronunciation subsequences. Sequitur has been trained until 8 iterations. It is to be noted that phonetisaurus runs only in Python 2.7. For training the sequence-to-sequence and BLSTM G2P model, data has been assembled into two different files; one having the graphemes as the source and in the other phonemes as target file across the split datasets for training. We use stochastic gradient descent (sgd) optimization method and set the learning rate at 0.001. The maximum batch size used for validation is 32 and for training is 62. The dropout probability for LSTM and BLSTM is 0.3.

For the ensemble classifier, we train the different G2P models; first in pairs and then all the models altogether. We trained 6 pairs of different G2P models in total. After the training, from all epochs, we choose the ensemble classifier based on the performance (classifier accuracy) on the validation set. Suppose, model1 is sequence-to-sequence and model2 is BLSTM while combining. In this case, we give a sequence of phonemes to the phoneme embedding network and sequence of graphemes to the grapheme embedding network. For training the ensemble classifier we use Adam optimizer, set the learning rate at 0.001. We use dropout layer of 0.5 dropout probability.

Moreover, when we combine two different G2P models to create the ensemble classifier the main objective of the classifier is to select the best G2P model out of the two models. Therefore, during the training phase if the output of both the models are same we have not considered it for training. It is irrelevant to make the classifier to choose between the classifiers as both of them have the same output. Thus such examples have been ignored as it will not affect the ensemble classifier's performance.

So, when we combine all the G2P model outputs there are certain cases in which more than one G2P model have the correct phoneme outputs. To handle such cases, first we estimated the model output as a predicted class and then we checked if the given predicted class have the lowest PER (phoneme error rate) or not. If given predicted class have lowest PER then we calculated the loss by giving it as a target class.

In other cases, if given predicted class don't have lowest PER then we gave target as a class having lowest PER to the loss function.

6.3 Evaluation

In G2P conversion task, two performance metrics are commonly used: **WER** i.e. word error rate and **PER** i.e. phoneme error rate (Bisani and Ney, 2008).

The **WER** is defined simply as the ratio of words with at least one phonetization error over the total number of words in the test set. The **PER** is the sum of the edit distances of all word phonetization to their correct counterparts over the total number of phonemes in the test set. So, in case of multiple pronunciations, the variant with the smallest edit distance is used. The **PER** is calculated as:

$$PER = \frac{S + D + I}{N} \tag{6.1}$$

Where,

- S = the number of substitutions,
- D = the number of deletions,
- I = the number of insertions,
- N = the number of phonemes in the groundtruth.

For calculating **WER**, word error occurs only if the predicted pronunciation doesn't match with any reference considering the multiple pronunciations. Thus, word error is obtained by calculating the number of wrong predictions. The **WER** is calculated as:

$$\frac{Words\ Incorrect}{Total\ Words} \tag{6.2}$$

We computed accuracy to evaluate our classifier. Accuracy is a metric for evaluating classification models. Informally, accuracy can be described as the fraction of predictions the model got right. It can be expressed formally as:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \tag{6.3}$$

6.4 Result and Analysis

In this section, we present the results obtained by the various approaches that we have implemented to examine and compare with our proposed approach of model combination. It is to be mentioned that, even after implementing the transformer network, our model failed to achieve any results.

6.4.1 Performance of individual G2P models

First, we carried out experiments using the individual models and compared our approach of adapting varied structures of neural networks with previous works.

TABLE 6.2: Results obtained by individual models.

Model	English (WER/PER)	French (WER/PER)
Sequitur	37.2% / 9.54%	16.3% / 3.75%
Phonetisaurus	34.6% / 9.16%	15.4% / 4.44%
Seq2seq	30.3% / 8.02%	10.6% / 2.93%
BLSTM	29.3% / 7.61%	9.06% / 2.51%

Our adaptation of BLSTM achieved comparatively the best result among the individual G2P models on the test set of the CMUdict for English with a WER of 29.3% and PER of 7.61% and on the test set of CMU pronunciation dictionary for French with WER of 9.06% and PER of 2.51%. Table 6.2 shows the results obtained by individual G2P models on the respective test sets (CMUdict for English and CMU pronunciation dictionary for French).

The number of items for which no pronunciation is predicted by sequitur is 33, and 4 by phonetisaurus for English test set (*refer appendix (A)*) ; sequence-to-sequence and BLSTM could successfully predict pronunciations for all the items (11409 in total).

6.4.2 Performance analysis of G2P models without the classifier

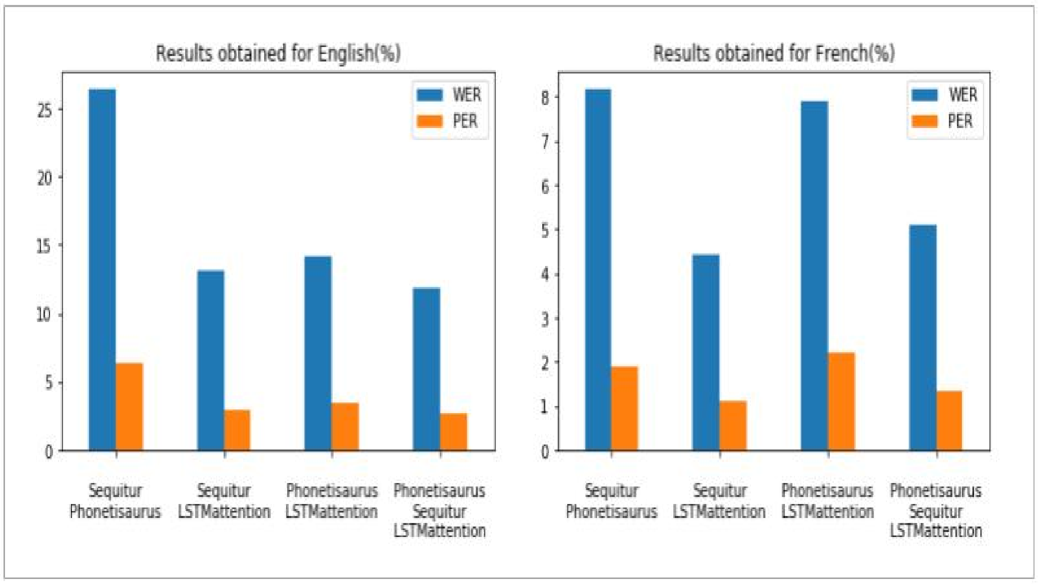


FIGURE 6.1: *WER and PER on test data having the same predicted pronunciations by two or more G2P systems.*

Figure 6.1 shows graphical representation of the results obtained by considering the identical outputs of different G2P models. Initially, we examined our method by taking into account the outputs (predicted pronunciations) that are same as two or more than two G2P models. First, we computed evaluation by merging the same outputs of two different G2P models into one. The outputs of sequitur and sequence-to- sequence has achieved the best results among them on identical predictions with a WER of 13.07% and PER of 2.98% on the CMUdict for English and WER of 4.44% and PER of 1.11% on the CMU pronunciation dictionary for French respectively. Then, we computed the evaluation by taking the same predicted outputs of three different G2P models and merged them as one. We achieved WER of 11.91% and PER of 2.70% on the English dataset and WER of 5.11% and PER of 1.33% on the French dataset on merging the same predictions of sequitur, phonetisaurus and sequence-to-sequence altogether.

We have also computed the number of the different pronunciations generated by two models on combining. It has also taken into account the pronunciations that either one of the models failed to generate. Table 6.3 below presents the number of the different pronunciations generated by two G2P models on combining for French dataset.

TABLE 6.3: Different pronunciations by 2 models for French data.

Models	No. of different predictions
Sequitur-Seq2seq	2688
Sequitur-Phonetisaurus	1458
Seq2seq-Phonetisaurus	2547

6.4.3 Analysis of the grapheme-to-phoneme models

In the case of sequitur and phonetisaurus, certain limitations have been observed while implementing the models. Sequitur and phonetisaurus generate words that are different from the ground-truth such as "Bretonne" which is generated as "Bretonnes" with an extra character /s/ in the final position of the word. It is observed that sequitur and phonetisaurus struggles with irregular pronunciation cases, for instance, abbreviations, hyphenated words or words with underscores, etc. Table 6.4 and 6.5 below presents the number of underscores and hyphenated words in the training and test data. Sequitur produces new pronunciations on its own by using the python command line. In the simplest case, during the network building phase, the G2P initiated a wordlist with the command:

$ get-pronunciation− − −wordlist words. Wl-model input. demo−− reference input. demo. lexicon output− − −

Reference words are looked up, OOV (out-of-vocabulary) are checked and optional parameters are provided to generate additional pronunciations for reference words and n-best for OOV's. Thus, resulting in the different size of the lexicon for each model even after using the same dataset. Sequence-to-sequence model doesn't encounter such difficulties in generating the pronunciations. It successfully generated pronunciations for all words.

TABLE 6.4: Underscore and hyphenated words in English dataset.

Dataset	Underscore	Hyphenated
Training	76	792
Test	4	100

TABLE 6.5: Underscore and hyphenated words in French dataset.

Dataset	Underscore	Hyphenated
Training	1654	1414
Test	196	194

6.4.4 Result and analysis of the G2P models with the classifier

We combined the outputs of different G2P models and give it to the classifier to select the best model output combination. First, we examined our approach of model

TABLE 6.6: Results obtained by combination of two different G2P models on the English dataset. Improvement is estimated with respect to the best individual model's WER and PER.

Model	Classifier (WER/PER)	Classifier accuracy	Improvement (WER/PER)
Seq2seq+BLSTM	28.36%/ 7.43%	54.81%	0.99% / 0.18%
Seq2seq+Sequitur	30.13%/ 7.97%	55.23%	0.17% / 0.05%
Seq2seq+Phonetisaurus	29.90%/ 7.91%	58.48%	0.40% / 0.11%
Sequitur+BLSTM	28.64%/ 7.42%	59.88%	0.71% / 0.19%
Sequitur+Phonetisaurus	34.27%/ 8.76%	51.97%	−0.21% / 0.40%
BLSTM+Phonetisaurus	28.52%/ 7.45%	61.18%	0.83% / 0.16%

combination by combining two different G2P models and then by combining all the four G2P models we have considered for carrying out our research experiments.

Table 6.6 and 6.7 shows results of model combination in pairs for the English and the French dataset. The combination of sequence-to-sequence with BLSTM achieved the best performance on the English dataset with an improvement of 0.99% in the WER (28.36%) and an improvement of 0.18% in the PER (7.43%) with our method of ensemble classifier. The estimated improvement is compared with respect to the best WER and PER from the combination of G2P models. Sequitur combined with BLSTM achieved almost similar performance like the combination of sequence-to-sequence with BLSTM with a slight difference in the WER and PER. Sequitur combined with BLSTM achieved an improvement of 0.71% in the WER (28.64%) and an improvement of 0.19% in the PER (7.42%) on the English dataset. Sequitur on combining with phonetisaurus achieves the lowest performance among all the G2P model combinations. The WER decreased by −0.21% while there is a slight improvement in the PER by 0.4%. The combination of all the G2P model outputs achieved WER of 28.80% and PER of 7.47% on the English dataset. Similarly, on the French dataset sequitur combined with phonetisaurus gives the best performance with an improved WER (14.35%) by 1.05% but the PER (3.99%) is decreased by −0.24%. However, this model combination failed to outperform the best individual G2P model BLSTM which achieved WER of 9.06% and PER of 2.51%.

TABLE 6.7: Results obtained by combination of two different G2P models on the French dataset. Improvement is estimated with respect to the best individual model's WER and PER.

Model	**Classifier (WER/PER)**	**Classifier accuracy**	**Improvement (WER/PER)**
Seq2seq+BLSTM	12.82%/ 3.68%	84.78%	−3.76% /−1.17%
Seq2seq+Sequitur	13.29%/ 3.81%	77.28%	−2.61%/ −0.88%
Seq2seq+Phonetisaurus	12.91%/ 3.77%	80.36%	−2.23%/ −0.84%
Sequitur+BLSTM	12.29%/ 3.32%	64.28%	−3.23%/ −0.81%
Sequitur+Phonetisaurus	14.35%/ 3.99%	79.97%	1.05%/ −0.24%
BLSTM+Phonetisaurus	12.90%/ 3.68%	60.96%	−3.84%/ −1.17%

Our method of ensemble classifier showed competitive results on the English dataset. The model combination has achieved better performance in contrast to the individual models on the CMU dictionary for English. The models sequence-to-sequence and BLSTM have shown very good performance across all the other G2P models on the English dataset which is why the combination of both the models can be termed as the best classifier. On the other hand, it is noticed that the sequitur and phonetisaurus have the lowest WER and PER among other G2P systems which suggest that classifier performs better when better model outputs are given to it.

It is to be noted that, we cannot evaluate the model performance based on the classifier accuracy because each pair of combination of G2P systems might have different number of incorrect and correct phoneme outputs. Furthermore, the WER and PER for French G2P systems are quite low. This makes difficult for the classifier to further enhance the performance.

TABLE 6.8: Results obtained by combining all the G2P models on the English dataset.

Model	**Seq2Seq**	**BLSTM**	**Sequitur**	**Phonetisaurus**	**All G2P model combination**
WER	30.3%	29.35	37.22%	34.06%	28.80%
PER	8.02%	7.61%	9.54%	9.16%	7.47%
Classifier Accuracy	-	-	—	—	54.64%

TABLE 6.9: Results obtained by combining all the G2P models on the French dataset.

Model	**Seq2Seq**	**BLSTM**	**Sequitur**	**Phonetisaurus**	**All G2P model combination**
WER	10.68%	9.06	16.34%	15.40%	12.60%
PER	2.93%	2.51%	3.75%	4.44%	3.60%
Classifier Accuracy	—	-	—	—	73.79%

We experimented our approach by combining all the G2P models for English as well as French for comparing the accuracy across the methods we applied for choosing the optimal one. Table 6.8 and 6.9 shows the results obtained by combining all the G2P models for both the dataset. When we combined all the G2P model outputs, we achieve WER of 28.80% and PER of 7.47% on the English dataset and WER of 12.60% and PER of 3.60% on the French dataset. Even after combining all the G2P models, sequence-to-sequence combined with BLSTM still performs the best on the English dataset. For French dataset the combination of all the models failed to achieve any improvements with respect to individual G2P model. Thus, for French dataset BLSTM individual G2P model provide the best performance in terms of WER and PER.

Chapter 7

Conclusion

In this thesis, we have investigated various grapheme-to-phoneme conversion models and implemented a robust framework of ensemble learning with multiple generator adversarial network. First of all, we examined the multigram approach for grapheme-to-phoneme conversion and considered sequitur and phonetisaurus as the baseline model. Then, we decided to apply deep neural network architectures for G2P conversion. With recent neural networks advancement, grapheme-to-phoneme conversion is viewed as a sequential task and modeled with the encoder and decoder framework. We carried out several experiments before providing an optimal solution to our problem. We adapted the architecture of sequence-to-sequence and transformer from OpenNMT library and modified accordingly to implement in our task. We train sequence-to-sequence model with various neural networks architecture, viz., recurrent neural network - long short-term memory (RNN LSTM), bi-directional LSTM; we applied the attention mechanism as well. Afterward, we analyzed the outputs (the identical pronunciations) of different G2P models and merged them as one to conduct our experiment on the test data and evaluated with our write-up code. We observed that neural network-based models i.e. sequence-to-sequence and BLSTM have performed quite well on both the datasets.

Our main objective is to provide a robust framework for choosing the best G2P model. Thus, we proposed a novel approach to implementing an ensemble classifier for choosing the best model output. We decided to combine the different G2P model outputs. Our approach relies on multiple generative adversarial network and ensemble learning, which is a completely new approach in the context of grapheme-to-phoneme conversion. Our core model is a deep neural network-based model.

Our approach of model combination has successfully shown very good results on the English dataset. Four out of six model combinations have achieved very good performance in terms of WER and PER outperforming the baseline models (sequitur and phonetisaurus). It also showed that by combining different G2P models, we can achieve better results than an individual G2P model. For instance, our three models i.e. sequence-to-sequence combined with BLSTM, sequitur combined with BLSTM, and phonetisaurus combined with BLSTM achieved better results then the best individual G2P model (BLSTM) on the English dataset. However, our model of ensemble classifier failed to perform well on the French data. The reason for underperformance by the classifier on the French data was mainly due to a large number of words having two or more pronunciation variants. Nevertheless, the combination of sequitur and phonetisaurus could improved the performance of WER by 1.05% on the French dataset. Thus, our ensemble classifier with the combination of sequence-to-sequence with BLSTM is the best G2P model on the English dataset. As we mentioned our ensemble classifier failed to achieve a good permformance on the French dataset, therefore, the individual BLSTM model is the best G2P model for French.

Among various experimentations, we implemented the transformer network as well for our research. Due to time constraint, we were not able to discover the prime reason for the failure of the transformer network in our work. Whether the network needs more training time or is there any drawback in the implementation, we couldn't figure it out as of now. For future work, we will be focusing on solving the problem of the transformer network.

Another work experiment that we would be interested in carrying out in the future is training the generators (G2P models) and the classifier at the same time in the same network. We would be interested in implementing a complete deep learning-based network architecture for our robust framework of ensemble multiple adversarial networks for the G2P conversion.

However, experimentation on the publicly available CMU dictionary for English has demonstrated the effectiveness of our approach in increasing the improvement of G2P models by combining different models comparing to an individual G2P model. We will leverage our method by extending it to carry out experimentation in other languages as well.

Bibliography

Andersen, Ove et al. (1996). “Comparison of two tree-structured approaches for grapheme-to-phoneme conversion”. In: *International Conference on Spoken Language Processing, ICSLP, Proceedings* 3. DOI: 10.1109/ICSLP.1996.607954.

Arciniegas, F. and M. J. Embrechts (2000). “Phoneme recognition with staged neural networks”. In: 5, 259–264 vol.5. ISSN: 1098-7576. DOI: 10.1109/IJCNN.2000.861467.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: ICLR 2015 as oral presentation, arxiv:1409.0473. URL: http://arxiv.org/abs/1409.0473.

Bengio, Y., P. Simard, and P. Frasconi (1994). “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. ISSN: 1045-9227. DOI: 10.1109/72.279181.

Bisani, Maximilian and Hermann Ney (2008). “Joint-sequence models for grapheme-to-phoneme conversion.” In: *Speech Communication* 50.5, pp. 434–451. URL: http://dblp.uni-trier.de/db/journals/speech/speech50.html#BisaniN08.

Cho, Kyunghyun et al. (2014). “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *CoRR* abs/1406.1078. arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

Cortes, Corinna and Vladimir Vapnik (1995). “Support-vector networks”. In: *Machine Learning* 20.3, pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. URL: https://doi.org/10.1007/BF00994018.

Deligne, Sabine, François Yvon, and Frédéric Bimbot (1995). “Variable-length sequence matching for phonetic transcription using joint multigrams.” In: URL: http://dblp.uni-trier.de/db/conf/interspeech/eurospeech1995.html#DeligneYB95.

Divay, Michel and Anthony J. Vitale (1997). “Algorithms for Grapheme-phoneme Translation for English and French: Applications for Database Searches and Speech Synthesis”. In: *Comput. Linguist.* 23.4, pp. 495–523. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=972791.972796.

Elovitz, H. et al. (1976). “Letter-to-sound rules for automatic translation of english text to phonetics”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24.6, pp. 446–459. ISSN: 0096-3518. DOI: 10.1109/TASSP.1976.1162873.

Fiscus, J. G. (1997). “A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER)”. In: pp. 347–354. DOI: 10.1109/ASRU.1997.659110.

Goodfellow, Ian J. et al. (2014). “Generative Adversarial Nets”. In: NIPS’14, pp. 2672–2680. URL: http://dl.acm.org/citation.cfm?id=2969033.2969125.

Graves, A. et al. (2009). “A Novel Connectionist System for Unconstrained Handwriting Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.5, pp. 855–868. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.137.

Graves, Alex (2012). “Supervised Sequence Labelling with Recurrent Neural Networks”. In: *Stud Comput Intell* 385. DOI: 10.1007/978-3-642-24797-2.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey E. Hinton (2013). "Speech Recognition with Deep Recurrent Neural Networks". In: *CoRR* abs/1303.5778. arXiv: 1303.5778. URL: http://arxiv.org/abs/1303.5778.

Hinton, Geoffrey, Oriol Vinyals, and Jeffrey Dean (2015). "Distilling the Knowledge in a Neural Network". In: URL: http://arxiv.org/abs/1503.02531.

Hoang, Quan et al. (2018). "MGAN: Training Generative Adversarial Nets with Multiple Generators". In: URL: https://openreview.net/forum?id=rkmu5b0a-.

Hochreiter, Sepp (1991). "Untersuchungen zu dynamischen neuronalen Netzen". In:

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-term Memory". In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Illina, Irina, Dominique Fohr, and Denis Jouvet (2011). "Grapheme-to-Phoneme Conversion using Conditional Random Fields". In: URL: https://hal.inria.fr/inria-00614981.

Jiampojamarn, Sittichai, Colin Cherry, and Grzegorz Kondrak (2008). "Joint Processing and Discriminative Training for Letter-to-Phoneme Conversion". In: pp. 905–913. URL: https://www.aclweb.org/anthology/P08-1103.

Jouvet, Denis, Dominique Fohr, and Irina Illina (2012). "Evaluating grapheme-to-phoneme converters in automatic speech recognition context". In: pp. 4821 – 4824. DOI: 10.1109/ICASSP.2012.6288998. URL: https://hal.inria.fr/hal-00753364.

Killer, Mirjam, Sebastian Stüker, and Tanja Schultz (2003). "Grapheme based Speech Recognition". In: URL: https://www.csl.uni-bremen.de/cms/images/documents/publications/Euro03-KillerSchultz.pdf.

Klein, Guillaume et al. (2017). "OpenNMT: Open-Source Toolkit for Neural Machine Translation". In: *CoRR* abs/1701.02810. arXiv: 1701.02810. URL: http://arxiv.org/abs/1701.02810.

Kominek, John and Alan W Black (2006). In: pp. 232–239. URL: https://www.aclweb.org/anthology/N06-1030.

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: ICML '01, pp. 282–289. URL: http://dl.acm.org/citation.cfm?id=645530.655813.

Lang, Kevin J., Alex H. Waibel, and Geoffrey E. Hinton (1990). "A Time-delay Neural Network Architecture for Isolated Word Recognition". In: *Neural Netw.* 3.1, pp. 23–43. ISSN: 0893-6080. DOI: 10.1016/0893-6080(90)90044-L. URL: http://dx.doi.org/10.1016/0893-6080(90)90044-L.

Lin, Y., E. Sontag, and Y. Wang (1996). "A Smooth Converse Lyapunov Theorem for Robust Stability". In: *SIAM Journal on Control and Optimization* 34.1, pp. 124–160. DOI: 10.1137/S0363012993259981. eprint: https://doi.org/10.1137/S036301299325998. URL: https://doi.org/10.1137/S0363012993259981.

Ling, Wang et al. (2015). "Character-based Neural Machine Translation". In: *CoRR* abs/1511.04586. arXiv: 1511.04586. URL: http://arxiv.org/abs/1511.04586.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning (2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *CoRR* abs/1508.04025. arXiv: 1508.04025. URL: http://arxiv.org/abs/1508.04025.

Matusov, Evgeny, Nicola Ueffing, and Hermann Ney (2006). "Computing Consensus Translation for Multiple Machine Translation Systems Using Enhanced Hypothesis Alignment". In: URL: https://www.aclweb.org/anthology/E06-1005.

McCulloch, N., Mark Bedworth, and John S. Bridle (1987). "NETspeak — A re-implementation of NETtalk". In:

McCulloch, Warren S. and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

Mousa, Amr and Björn Schuller (2016). “Deep Bidirectional Long Short-Term Memory Recurrent Neural Networks for Grapheme-to-Phoneme Conversion Utilizing Complex Many-to-Many Alignments”. In: pp. 2836–2840. DOI: 10.21437/Interspeech.2016-1229.

Novak, Josef R., Nobuaki Minematsu, and Keikichi Hirose (2012). “WFST-Based Grapheme-to-Phoneme Conversion: Open Source tools for Alignment, Model-Building and Decoding”. In: pp. 45–49. URL: https://www.aclweb.org/anthology/W12-6208.

Och, Franz Josef and Hermann Ney (2002). “Discriminative Training and Maximum Entropy Models for Statistical Machine Translation”. In: pp. 295–302. DOI: 10.3115/1073083.1073133. URL: https://www.aclweb.org/anthology/P02-1038.

Pagel, Vincent, Kevin A. Lenzo, and Alan W. Black (1998). “Letter to Sound Rules for Accented Lexicon Compression”. In: *CoRR* cmp-lg/9808010. URL: http://arxiv.org/abs/cmp-lg/9808010.

Polyàkova, Tatyana V. (2014). “Grapheme-to-Phoneme Conversion in the Era of Globalization”. In: *TALP Research Center, Speech Processing Group Department of Signal Theory and CommunicationsUniversitat Politècnica de Catalunya* 62.1, pp. 1–176. URL: https://upcommons.upc.edu/bitstream/handle/2117/95670/TTVP1de1.pdf.

Rao, Kanishka et al. (2015). “Grapheme-to-phoneme conversion using Long Short-Term Memory recurrent neural networks”. In: pp. 4225–4229. DOI: 10.1109/ICASSP.2015.7178767.

Rasipuram, Ramya and Mathew Magimai-Doss (2012). “Acoustic data-driven grapheme-to-phoneme conversion using KL-HMM”. In: *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pp. 4841–4844. DOI: 10.1109/ICASSP.2012.6289003.

Rokach, Lior (2010). “Ensemble-based classifiers”. In: *Artificial Intelligence Review* 33.1, pp. 1–39. ISSN: 1573-7462. DOI: 10.1007/s10462-009-9124-7. URL: https://doi.org/10.1007/s10462-009-9124-7.

Rosenblatt, F. (1958). “The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain”. In: *Psychological Review*, pp. 65–386.

Rumelhart David E., Hinton Geoffrey E. Williams Ronald J. (1986). “Learning representations by back-propagating errors”. In: *Nature*. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

Schmidhuber, J. (1992). “Learning Complex, Extended Sequences Using the Principle of History Compression”. In: *Neural Computation* 4.2, pp. 234–242. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.2.234.

Schuster, M. and K.K. Paliwal (1997). “Bidirectional Recurrent Neural Networks”. In: *Trans. Sig. Proc.* 45.11, pp. 2673–2681. ISSN: 1053-587X. DOI: 10.1109/78.650093. URL: http://dx.doi.org/10.1109/78.650093.

Sejnowski, Terrence J. and Charles R. Rosenberg (1987). “Parallel Networks That Learn to Pronounce English Text”. In: *Complex Systems* 1, pp. 145–168.

Shubham Toshniwal, Karen Livescu (2016). “Read, Attend and Pronounce:An Attention-Based Approach for Grapheme-To-Phoneme Conversion”. In: p. 2. DOI: MLSLPWorkshop, InterSpeech2016.

Sun, Hao et al. (2019). “Token-Level Ensemble Distillation for Grapheme-to-Phoneme Conversion”. In: *CoRR* abs/1904.03446. arXiv: 1904.03446. URL: http://arxiv.org/abs/1904.03446.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). “Sequence to Sequence Learning with Neural Networks”. In: *CoRR* abs/1409.3215. arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

Taylor, Paul (2005). “Hidden Markov models for grapheme to phoneme conversion”. In: pp. 1973–1976.

Vaswani, Ashish et al. (2017). “Attention Is All You Need”. In: *CoRR* abs/1706.03762. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

Yao, Kaisheng and Geoffrey Zweig (2015). “Sequence-to-Sequence Neural Net Models for Grapheme-to-Phoneme Conversion”. In: *CoRR* abs/1506.00196. arXiv: 1506.00196. URL: http://arxiv.org/abs/1506.00196.

Yolchuyeva, Sevinj, Géza Németh, and Bálint Gyires-Tóth (2019). “Grapheme-to-Phoneme Conversion with Convolutional Neural Networks”. In: *Applied Sciences* 9.6. ISSN: 2076-3417. DOI: 10.3390/app9061143. URL: https://www.mdpi.com/2076-3417/9/6/1143.

Zhang, Aston et al. (2019). “Dive into Deep Learning”. In: URL: http://www.d2l.ai.

Appendix A

List of items that failed to be to generated by G2P systems for English

TABLE A.1: List of items that failed to be to generated by Sequitur for English.

Word	Pronunciation
DJ	D EY2 JH AA1
DOUGLAS'	D AH1 G L AH0 S
NAPPED	N AE1 P T
GOTTSCHALL	G AA1 CH AH0 L
%PERCENT	P ER0 S EH1 N T
AIREDALES	EH1 R D EY2 L Z
MCLEISH	M AH0 K L IY1 SH
BRONCHITIS	B R AA0 NG K AY1 T AH0 S
LIFERS	L AY1 F ER0 Z
DEBELL	D IY1 B EH0 L
GARANT	G AA1 R AH0 N T
POLAR	P OW1 L ER0
KAISERTECH	K AY1 Z ER0 T EH2 K
ARDOR	AA1 R D ER0
CHIDE	CH AY1 D
PRECLUDES	P R IY0 K L UW1 D Z
EAVES	IY1 V Z
KOHNE	K OW1 N
SUNSTATES	S AH1 N S T EY2 T S
FALSIFYING	F AO1 L S AH0 F AY2 IH0 NG
ACIDOSIS	AE2 S AH0 D OW1 S AH0 S
THAILAND	T AY1 L AE2 N D
MINDEN	M AY1 N D AH0 N
?QUESTION-MARK	K W EH1 S CH AH0 N M AA1 R K
DARKENED	D AA1 R K AH0 N D
DOBER	D OW1 B ER0
AMELIO	AH0 M IY1 L IY0 OW0
BUYING	B AY1 IH0 NG
BEHAVE	B IH0 HH EY1 V
BRAZZAVILLE	B R AE1 Z AH0 V IH0 L

Continued on next page

Table A.1 – continued from previous page

Word	Pronunciation
STREED	S T R IY1 D
RELEND	R IY0 L EH1 N D
LANDIS	L AE1 N D IH0 S
HAGUE	HH EY1 G

TABLE A.2: **List of items that failed to be to generated by Phonetisaurus for English.**

Word	Pronunciation
ST_MARY	S EY1 N T M EH1 R IY0
ST_LOUIS	S EY1 N T L UW1 AH0 S
ST_LOUIS(1)	S EY1 N T L UW1 IY0
H EY1 CH ST_MARTIN	S EY1 N T M AA1 R T IH0 N

Appendix B

List of items that failed to be generated by G2P systems for French

TABLE B.1: List of items that failed to be to generated by Sequitur for French.

Word	Pronunciation
hula	uu ll aa
hamadé	aa mm aa dd ei
derghal	dd ai rr gg aa ll
àproximité edu	aa pp rr oo kk ss ii mm ii tt ei dd uu
gloucestershire	gg ll ou ss ai ss tt ai rr ch ii rr
mouche	mm ou ch ee
boycottée	bb oo yy kk oo tt ei
zélés	zz ai ll ei zz
contreforts	kk on tt rr ee ff oo rr zz
insoumis	in ss ou mm ii zz
albeniz	aa ll bb ee nn ii zz
àlafaçon de	aa ll aa ff aa ss on dd ee
brillait	bb rr ii yy ai tt
àpropos de	aa pp rr oo pp au dd ee
àmi-chemin	aa mm ii ch ee mm in
àraison d'	aa rr ai zz on dd
pilotage	pp ii ll oo tt aa jj ee
lahaye	ll aa ai yy
arcachon	aa rr kk aa ch on
centièmes	ss an tt yy ai mm zz
détecté	dd ai tt ai kk tt ei
min	mm in
àlafaçondes	aa ll aa ff aa ss on dd ei zz
hum	oe mm
hypothécaire	ii pp oo tt ai kk ai rr ee
tenet	tt ee nn ai
àl'égarddes	aa ll ei gg aa rr dd ei
plumes	pp ll uu mm zz
àdemi-mot	aa dd ee mm ii mm au
retombe	rr ee tt on bb ee
epargne	ei pp aa rr gn ee

Continued on next page

Table B.1 – continued from previous page

Word	Pronunciation
ysl	ii ss ai ll
synchroniser	ss in kk rr oo nn ii zz ei rr
àtoutprix	aa tt ou pp rr ii
portables	pp oo rr tt aa bb ll zz
venise	vv nn ii zz ee
blessant	bb ll ai ss an tt
àreculons	aa rr ee kk uu ll on
àsupposerqu′	aa ss uu pp au zz ei kk
financé	ff ii nn an ss ei
affamés	aa ff aa mm ei zz
àgauchede	aa gg au ch ee dd ee
àl′encontrede	aa ll an kk on tt rr ee dd ee
effacera	ai ff aa ss ee rr aa
delfont	dd ai ll ff on
àlafind′	aa ll aa ff in dd
domiciliation	dd oo mm ii ss ii ll yy aa ss yy on
baissera	bb ai ss ee rr aa
yakuzas	yy aa kk uu zz aa
ultimatums	uu ll tt ii mm aa tt oo mm zz
finals	ff ii nn aa ll zz
kipkirui	kk ii pp kk ii rr uu ii
enfoncée	an ff on ss ei
grâceà	gg rr aa ss aa
moscou	mm oo ss kk ou
disséminées	dd ii ss ai mm ii nn ei zz
désespérées	dd ai zz ai ss pp ai rr ei zz
mauresmo	mm oo rr ai ss mm au
sédition	ss ai dd ii ss yy on
altusfinance	aa ll tt uu ss ff ii nn aa nn ss
pilotis	pp ii ll oo tt ii zz
mouille	mm ou yy ee
stoufflet	ss tt ou ff ll ai
castela	kk aa ss tt ei ll aa
convenait	kk on vv ee nn ai tt
recomposition	rr ee kk on pp oo zz ii ss yy on
bretonne	bb rr tt oo nn ee
monge	mm on jj
salvateur	ss aa ll vv aa tt oe rr
endeçà	an dd ee ss aa
guérit	gg ai rr ii tt
àcompterde	aa kk on tt ei dd ee
lugubre	ll uu gg uu bb rr ee
grands-parents	gg rr an pp aa rr an zz
domont	dd oo mm on
meurtrières	mm ee rr tt rr ii yy ai rr zz
euégardà	uu ei gg aa rr tt aa
nettoyée	nn ai tt ww aa yy ei

TABLE B.2: **List of items that failed to be to generated by Phonetisaurus for French.**

Word	Pronunciation
jusqu'_au	jj uu ss kk au
vis-à-vis_des	vv ii zz aa vv ii ss dd ei zz
aussitôt_qu'	au ss ii tt au kk
par_la_suite	pp aa rr ll aa ss uy ii tt
à_proximité_du	aa pp rr oo kk ss ii mm ii tt ei dd uu
en_quel que_sorte	an kk ai ll kk ee ss oo rr tt
bien_entendu	bb yy in nn an tt an dd uu
quelle_que_soit	kk ai ll ee kk ee ss ww aa
d'_autant_plus_que	dd au tt an pp ll uu ss kk ee
d'_autant_plus_qu'	dd au tt an pp ll uu ss kk
cap_canaveral	kk aa pp kk aa nn aa vv ei rr aa ll
de_tous_côtés	dd ee tt ou kk au tt ei
en_goguette an	gg oo gg ai tt
au_beau_milieu_du au bb au	mm ii ll yy eu dd uu
c'_est_ainsi_qu'	ss ai ss tt in ss ii kk
de_loin	dd ee ll ww in
au_sein_d'	au ss in dd
à_demi	aa dd ee mm ii
buenos_aires	bb uu ei nn oo ss aa ii rr ai ss
au_demeurant	oo dd ee mm ee rr an
près_des	pp rr ai dd ei zz
un_peu_plus_que	un pp eu pp ll uu ss kk ee
à_la_façon_de	aa ll aa ff aa ss on dd ee
à_propos_de	aa pp rr oo pp au dd ee
à_mi-chemin	aa mm ii ch ee mm in
grosso_modo	gg rr oo ss oo mm oo dd au
à_raison_d'	aa rr ai zz on dd
la_haye	ll aa ai yy
pour_qu'	pp ou rr kk
tout_à_fait	tt ou tt aa ff ai
jusqu'à_présent	jj uu ss kk aa pp rr ei zz an
mises_en_oe uvre	mm ii zz ee zz an nn oe vv rr
même_si	mm ai mm ee ss ii
peu_à_peu	pp eu aa pp eu
les_miens	ll ai mm yy in zz
pro_domo	pp rr oo dd oo mm au
dans_le_but_de	dd an ll ee bb uu tt dd ee
dans_la_plupart_des_cas	dd an ll aa pp ll uu pp aa rr dd ei kk aa
en_dehors_d'	an dd ee oo rr dd
le_caire	ll ee kk ai rr
autre_part	au tt rr ee pp aa rr
à_la_façon_des	aa ll aa ff aa ss on dd ei zz
dès_lors_que	dd ai ll oo rr kk ee
à_l'égard_des	aa ll ei gg aa rr dd ei
au_travers_de	au tt rr aa vv ai rr dd ee
en_cours_de	an kk ou rr dd ee

Continued on next page

Table B.2 – continued from previous page

Word	Pronunciation
en_conséquence	an kk on ss ei kk an ss
ad_hoc	aa dd oo kk
non_plus	nn on pp ll uu ss
liban_sud	ll ii bb aa nn ss uu dd
alors_que	aa ll oo rr kk ee
à_demi-mot	aa dd ee mm ii mm au
en_vertu_d'	an vv ai rr tt uu dd
plus_tôt	pp ll uu ss tt au
à_tout_prix	aa tt ou pp rr ii
qu'_on_le_veuille_ou_non	kk on ll ee vv oe yy ou nn on
en_outre_de	an nn ou tt rr ee dd ee
made_in	mm aa dd in
àculons	aa rr ee kk uu ll on
hors_de	oo rr dd ee
à_supposer_qu'	aa ss uu pp au zz ei kk
parce_que	pp aa rr ss ee kk ee
sans_qu'	ss an kk
beaucoup_plus_que	bb au kk ou pp ll uu ss kk ee
quel_que_soit	kk ai ll kk ee ss ww aa
à_gauche_de	aa gg au ch ee dd ee
à_l'encontre_de	aa ll an kk on tt rr ee dd ee
la_vôtre	ll aa vv au tt rr ee
à_la_fin_d'	aa ll aa ff in dd
pour_ce_qu'	pp ou rr ss ee kk
d'_une_manière_générale	dd uu nn ee mm aa nn yy ai rr ee jj ei nn ei rr aa ll
de_la_part_	de dd ee ll aa pp aa rr dd ee
au_de	au ss oo rr tt ii rr dd ee
in_utero	in uu tt ee rr au
en_rapport_avec	an rr aa pp oo rr aa vv ai kk
la_sienne	ll aa ss yy ai nn ee
loin_qu'	ll ww in kk
indépendamment_de	in dd ei pp an dd aa mm an dd ee
grâce_à	gg rr aa ss aa
aux_dépens_du	au dd ei pp ai nn dd uu
la_plupart_du_temps	ll aa pp ll uu pp aa rr dd uu tt an
altus_finance	aa ll tt uu ss ff ii nn aa nn ss
d'_autant_qu'	dd au tt an kk
mise_en_oeuvre	mm ii zz an nn oe vv rr
en_cours	an kk ou rr
porto_rico	pp oo rr tt au rr ii kk au
agence_reuters	aa jj an ss ee rr eu tt ei
herald_tribune	ei rr aa ll dd tt rr ii bb yy ou nn
au_sujet_de	au ss uu jj ai dd ee
phnom_penh	pp nn oo mm pp ai nn
du_fait_qu'	dd uu ff ai kk
les_uns	ll ai zz un