Acoustic Data Based Grapheme to Phoneme Conversion

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Deshpande Akshay Ajay

Graduate Program in Computer Science and Engineering

The Ohio State University

2013

Abstract

Grapheme to Phoneme conversion (G2P), also called Letter to Sound mapping (LTS) is defined as the task of learning the relationship between the written word and its phonetic transcription. It is a necessary part of Text to Speech systems and plays a vital role in handling Out of Vocabulary (OOV) words in Automatic Speech Recognition systems. G2P is a complex task, because for many languages, the correspondence between the orthography (spelling) and its phonetic transcription is not completely consistent. Over time, the techniques used to tackle this problem have evolved, from earlier rule based systems to the current more sophisticated machine learning approaches.

All the existing systems for G2P rely on pronunciation dictionaries as their source of training data. In this thesis we explore the G2P problem from the perspectives of using speech data to aid an existing system and possibly improve its performance. In order to do that, we will look at some of the current G2P techniques and reproduce established results as a part of the background work. We then utilize a measure of closeness between graphemes and phonemes that formulates an acoustic score. We conduct experiments by combining our score with existing systems and analyze the results. The current results do not surpass the baseline system but point the way towards future innovations.

Dedication

This document is dedicated to my family.

Acknowledgments

I am deeply grateful to my advisor, Dr Eric Fosler for his guidance and continuous support throughout my research. I would also like to thank the committee member, Dr Kulis, for his constructive suggestions to improve the thesis.

This research was supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12- C-0014. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

Vita

2009	B.E. Instrumentation and Control, University of Pune.
2011 to present	Graduate Research Assistant, Department of Computer Science, The Ohio State University.

Fields of Study

Major Field: Computer Science and Engineering

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Speech and the written word are the two main ways humans communicate. The fundamental units that make up written text, be it in any language, are called graphemes, while phonemes are descriptors of how a word in a particular language is pronounced. The aim of grapheme-to-phoneme conversion (G2P) is to find the most appropriate pronunciation, given the written word. This is by no means a trivial task for most languages.

What we can observe by studying various languages is that the relationship between a word and its pronunciation is not completely consistent. The measure of strength of grapheme-phoneme correspondence for a language is called its orthographic transparency. (Borgwaldt, Hellwig, & Groot, 2005) did a comparative study of seven languages with regards to their orthographic transparency. For example, they discovered that languages like Italian and Hungarian exhibit shallow orthographies, which means they show strong correlation between written words and their pronunciations. Other languages like English exhibit deep orthographies, wherein grapheme-phoneme relations are more complex.

This complexity stems from two factors. First, multiple graphemes can correspond to multiple phonemes. Second, sometimes the grapheme-phoneme relations are ambiguous. As an example to demonstrate the first point, consider the English language, which has many instances of what we term as double grapheme and double phonemes. A double grapheme would be a case where two contiguous graphemes correspond to a single phoneme (For example, $[T][H] \rightarrow th$). Likewise, a double phoneme would be an example of two phonemes corresponding to one grapheme(For example, $X \rightarrow [k][s]$). To demonstrate the ambiguity in grapheme-phoneme correspondences consider the following two word pronunciation pairs as shown in table .

Word	Pronunciation
ENOUGH	iy n ah f
GHOST	g ow s t

Table 1: Example of word pronunciation ambiguity

It is evident that for different examples the grapheme chunk 'GH' can be associated with different phonetic units, like 'f' or 'g'. Thus we would need some form of contextual information to resolve the ambiguity. Another aspect that makes G2P difficult is the fact that orthographies themselves evolve, as languages borrow words from other languages. Pronunciation variation for the same written word is common because of existence of dialects and accents. All these factors make G2P a non trivial and an interesting problem. G2P plays a vital role in many applications. It is an integral part of speech synthesis applications. The core step in speech synthesis is to first convert the text into its phonetic form. Then we use the phoneme sequence to synthesize appropriate waveforms. G2P is used in handling pronunciations for Out of Vocabulary (OOV) words in speech recognition systems. (Hahn, Vozila, & Bisani, 2012) compare performance of various G2P techniques for handling OOV words. Another interesting application of G2P is in designing spell checking systems. Errors in spelling can stem from either typographic or cognitive sources. G2P can be used to correct cognitive errors. (Toutanova & Moore, 2002) provide a solution for spell checking by modeling pronunciation similarities between words.

Several data driven techniques like Joint Multigram Models (JMM) and Conditional Random Fields (CRF) have been formulated for G2P. We note that all the existing methods for G2P rely on a dictionary consisting of word pronunciation pairs as the only source of data. We hypothesize that in addition to the dictionary, we could use a labeled speech corpus as a potential data source to model grapheme-phoneme relationships. The central idea here is that the same speech data represents the pronunciation (expressed as a sequence of phonemes) and the orthography (underlying sequence of graphemes). We believe we can use speech data as a link to connect grapheme and phoneme labels. Thus the aim of my thesis is to develop a measure of grapheme-phoneme correspondence using a speech corpus and integrate it in current state of art techniques to improve their performance on the G2P task.

1.1 Outline

The structure of the thesis is as follows. First in Chapter 2, I review various approaches to G2P in literature. In Chapter 3, I present Joint Multigrams, Conditional Random Fields and Phonetisaurus (a WFST based G2P tool) as the state of art techniques for G2P and describe the experiments done using these techniques on the English dataset. Then in Chapter 4, I present acoustic G2P approach described earlier which is a novel method to use acoustic information to augment existing G2P methods. Finally in Chapter 5, I provide the conclusion of this thesis.

Chapter 2: Related Work

In this chapter, we will give an overview of the existing literature for G2P. Then we will explore in detail three techniques which are the current standard in terms of their effectiveness in giving good results, as well as document the results of the experiments done on English dictionary data using existing open source tools.

The earliest solutions used to address the problem of G2P were knowledge based. For example, we can simply store a dictionary of word pronunciation pairs in memory. Given a word, we can search in this dictionary for its pronunciation. However this is too simplistic as manually storing an ever increasing list of words is costly, and this method does not include any mechanism to find the pronunciation for unseen words.

2.1 Rule Based Methods

Rule based systems were developed as a more efficient way to capture the grapheme-phoneme relation than having an exhaustive dictionary (Kaplan & Kay, 1994). Although most systems had a list of outlier examples to accommodate known exceptions to the rules, they still had drawbacks. They needed experts in the specific language being dealt with, in order to come up with a rule set which will give good results. This is not possible for every case. (Kominek & Black, 2006) showed that for languages like English, where the association between graphemes and phonemes is sometimes ambiguous, the number of rules approaches the lexicon size, which is undesirable.

2.2 Data driven methods

In contrast, there is another group of methods for G2P which focus on the idea of "let the system discover the rules by learning from data examples". This idea of learning by analogy is more generalized than the idea of rule based methods and frees us from the language specific expertise. There are two major steps in data driven methods, namely the alignment step and the phoneme generation step. As a first step, we need to be able to compute an alignment between letters and phonemes which make up the training examples. This is required because although we want to predict phonemes given graphemes, the training data that we start with is in the form of word-pronunciation pairs. Alignments are crucial for our task because they allow us to go from having information about the relationship between our components of training data at string level to the substring level. We could have a 1-1 alignment or a 1 to many (or 1-n) alignment between the word and its pronunciation. We use a special symbol (epsilon) in the alignment to express silent letters in the orthography as well as to account for the difference in string length of word and its pronunciation for the 1-1 case. An example of a 1-1 alignment for the word 'ABACK' is shown below.

graphemes	A	B	A	C	K
phonemes	ah	b	a	k	null

Table 2: Example of 1-1 grapheme-phoneme alignment

1-1 alignments do not work well for languages like English where it is common to encounter double graphemes and double phonemes as discussed in chapter 1. There are two broad categories with this respect; namely classification based and sequence modeling based.

2.2.1 Classification based methods

These methods treat the task of G2P as a multiclass classification problem, with each phoneme being a output label class. Each output label is predicted independently using the current grapheme and a context around it. These include neural networks and decision trees. Grapheme context is shown to be very crucial to most techniques discussed here. (Sejnowski & Rosenberg, 1987) proposed a three layer neural network including one hidden layer with back propagation training for English G2P. A grapheme context of length three was taken into account and they were able to predict phonemes as well as articulator and stress labels. (Chen, 2003) models the conditional distribution of the phonemes given the word by using maximum entropy criterion.

2.2.2 Generative Sequence tagging based methods

G2P can be thought of as predicting a label or tag sequence $Y = y_1 y_2 \dots y_n$for the input graphemes $X = x_1 x_2 \dots x_n$. Popular methods used for this are Hidden Markov Models (HMM) and Joint Multigram Models (JMM). (Taylor, 2005) proposes a HMM based solution with phonemes being the hidden variables and graphemes occupy emission states. The HMM uses Baum-Welch training and accomplishes both alignment and phoneme prediction. Consider the HMM formulation for a set of hidden states S an observation states O.

$$P(S|O) = \prod P(s_t|s_{t-1})P(o_t|s_t)$$

In this setup, graphemes are observations and we can see that grapheme context information is not encoded as consecutive states $O_t\ and\ O_{t-1}$ are independent of each other. The current phoneme depends only on current grapheme and some of the previous phonemes as dictated by the Markov assumption. For this reason HMM systems do not perform as well as the techniques discussed earlier. (Deligne, Yvon, & Bimbot, 1995) (Bisani, Ney, 2002)(Maximilian Bisani & Ney, 2008) posited the idea of using joint grapheme phoneme pairs as fundamental units, called 'graphones'. We can visualize a word-pronunciation pair as a sequence of segments, each containing a graphone. An optimum alignment in the form of a co-segmentation is found out using EM style algorithm and a n-gram model is applied over it to predict pronunciation given the orthography.

2.2.3 Structured Output Prediction based methods

Conditional Random Fields (CRF) is another discriminative framework, proposed by (Lafferty, McCallum, & Pereira, 2001) which is proven to be effective for sequence tagging problems.(D. Wang & King, 2011) demonstrated good results on the AMI RT05s (Hain et al., 2006) using linear chain CRF with progressively increasing context lengths. One drawback of CRF is that you need an alignment pre-computed using some other method.(Bartlett, Kondrak, & Cherry, 2008) used Support Vector Machines for performing syllabification on the CELEX and NETtalk dictionaries. They show that orthographic syllabification leads to improvement in G2P results.

2.2.4 Acoustic G2P based methods

There have been some recent attempts to incorporate speech corpus into the G2P conversion process. One of the reasons this trend is the interest in grapheme based speech recognition. (Killer, M., Stüker, S., & Schultz, 2003) did extensive study of grapheme based speech recognition across languages like English, German and Spanish. (Magimai, Rasipuram, Aradilla, & Bourlard, 2011) presented a system for speech recognition where they jointly model grapheme and phoneme information into a HMM framework using Kullback-Liebler distance. The HMM states are parameterized using multinomial distributions. (Rasipuram & Doss, 2012) extend the previous idea by using these multinomial distributions along with an ergodic HMM for decoding phonemes.

To conclude, in this chapter, we did a survey of techniques being used for G2P. In the next chapter, we will study in detail some of the techniques relevant to our thesis.

Chapter 3: State of the Art Methods for G2P

In the previous chapter we looked at some of the important literature on G2P. As mentioned in chapter 1, we want to leverage speech data to improve existing techniques. As a step in that direction, we will first study the existing techniques which currently claim to give the best results. In this section we will focus on three existing state of the art techniques, namely Conditional Random Fields, Joint Multigram Models and Phonetisaurus, a tool based on the WFST framework. We will discuss the experiments done using these methods and the results obtained. Throughout the scope of the thesis we use CMU Pronunciation dictionary (Weide, 2005). This dictionary has about 130K words in English and their pronunciation using the 39 phoneme set given by ARPAbet. It provides lexical stress markers for vowels, although for our purpose we ignore them.

3.1 Conditional Random Fields (CRF) for G2P

Conditional Random Fields (CRFs) are a popular probabilistic framework for discriminative modeling. CRFs are shown to be well suited for segmenting and labeling sequential data. As presented by (Lafferty et al., 2001), a CRF is an undirected graphical model of a target sequence of labels, which are conditioned on the observation sequence. There are two main reasons why using CRFs for G2P is an idea worth pursuing. First, CRFs , by virtue of being discriminative in nature, can directly model the posterior probability. Thus we do not need to model the joint probability distribution of observations. The other advantage is that they perform global inference over the complete label sequence. In addition the loss function for a CRF is convex, hence global convergence is a certainty.

3.1.1 Linear Chain CRF

For the purpose of our experiments, we are going to use a specific form of CRF, namely linear chain CRF. An example of a linear chain CRF is shown below.

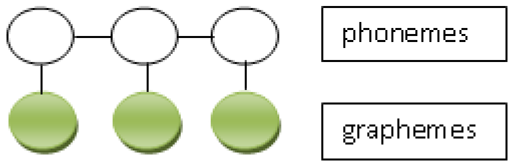

Figure 1: Linear chain CRF

As shown in the figure above, each node represents a random variable. Assuming we make the 1st order Markov assumption, all the nodes in the graph form a linear chain. Using the definition from (Lafferty et al., 2001), a linear chain CRF applied to G2P is specified by the following conditional probability.

$$P(Y|X) = \frac{1}{Z(X)} \exp\left\{\sum_{k=1}^{K} \alpha_k F_k(Y,X)\right\}$$

where X is the grapheme sequence of a word, Y is a candidate pronunciation, F_k is the k^{th} aggregated feature and α_k is a weight of the feature. Z(X) is a normalization quantity given by

$$Z(X) = \sum_{X} \exp\left\{\sum_{k=1}^{K} \alpha_k F_k(Y,X)\right\}$$

Thus the graph is separated into cliques, each of which constitutes two consecutive phonemes and the entire grapheme sequence. Thus $F_k(Y, X)$ can be expressed in terms of features of cliques, given by

$$F_k(Y, X) = \sum_{j=1}^{m-1} \{f_k(Y_j, Y_{j-1}, X, j)\}$$

We will use features that are binary functions that look for presence of graphemes and phonemes at various positions in the clique.

3.1.2 Experiments and Results

To set up the experiment we need two things. First, we need a tool for aligning the training examples. Second, we need a tool to perform training on the aligned training data. We use the Giza++ toolkit (Casacuberta & Vidal, 2007) to get 1-1 alignments. Giza++ treats the set of words as a source language and the set of pronunciations as a target language. Then learning the mapping between these two languages is modeled as a statistical translation problem.

To do the actual CRF training and testing, we use the CRF++ toolkit. This tool is developed by the NTT Communication Science Laboratories in Japan (Kudo, 2005). This open source tool, written in C++, uses the limited memory BFGS algorithm for training CRFs. This speeds up execution while making sure the memory requirements do not escalate to unmanageable proportions. In addition , this allows us to specify fairly large number of feature functions. It also gives us an option to use L1 or L2 regularization.

This toolkit allows the user to specify feature templates in advance. A macro %x[row, column] is used to specify the location of a token in the input file corresponding to the current token. It expands these macros using the training data to generate the appropriate binary indicator functions. There are two type of feature functions, unigram and bigram. The unigram feature involves only the current output token, while bigram features, if specified, contain a combination of previous and current output token. Consider the following example.

A:ah B:b A:ae C:k K:null

With this as reference the template 'T' is %x[0,0] would expand to generate functions of the following form.

func1 = if (output = k and feature="T:C") return 1 else return 0

func2 = if (output = b and feature="T:B") return 1 else return 0 ...

Thus there would be a feature function for each combination of grapheme token and label phone. We incorporate information about the grapheme context by using a n-gram of input tokens. So a macro of the form %x[-1,0]/%x[0,0] would consider both the current and previous grapheme token along with the current label phone.

We consider experiments with grapheme context windows of size 2 and 3 on CMUDict and report phone error rates (PER) using 1 best scoring and L2 regularization. We found out that L2 regularization works better than L1 to prevent over fitting.

The results are shown in the table below.

CRF window	PER (%)
CRF(+2,-2)	14.0
CRF(+3,-3)	12.0

Table 3: CRF result for CMUDict

We observe that as we capture longer context , we get an improvement in the performance. This is in line with our intuition as longer context allows learning the mapping between grapheme clusters and phonemes, which are common for a language like English. However longer context also means more feature functions. We encountered memory limitations for contexts greater than three, when we used CRF++ on CMUDict.

3.2 Joint Multigram Models for G2P

This section describes our efforts to develop G2P systems for English using Joint Multigram Models (JMM). We will follow the JMM formulation proposed by authors Bisani and H. Ney (Maximilian Bisani & Ney, 2008) and use **Sequitur G2P**, a joint Multigram based tool from RWTH Aachen university developed by the same authors. First we will introduce JMM as a joint sequence model for G2P. Then we will present our work using Sequitur for English.

3.2.1 Joint Sequence Modeling

The key idea in joint sequence approach is instead of considering the words and pronunciations being derived separately using different units, we treat the data as being generated by a common set of joint units. We stick with the term coined by Bisani and Ney, namely graphones or grapheme–phoneme joint multigrams to denote these units. We represent a letter sequence by the symbol $\boldsymbol{g}$ and a phoneme sequence by $\boldsymbol{y}$. A graphone would then be the pair $\boldsymbol{q} = (\boldsymbol{g}, \boldsymbol{y})$**.** Thus word-pronunciation pairs are now expressed as a sequence of graphones. However as the individual units making up a graphone segment can be of varying lengths, we can have many possible co-segmentations for the same pair. Consider the example $GUILTY \rightarrow g\ ih\ l\ t\ iy$. A couple of possible segmentations are shown in the table below.

	Example Co-Segmentation				
GUILTY	G	UI	L	T	Y
g ih l t iy	g	ih	l	t	iy

Example Co-Segmentation				
G	UI	L	T	Y
g_ih	-	l	t_iy	-

Table 4: Example co-segmentations for the word GUILTY

We can immediately see that some segmentations better than others. The joint probability of a word pronunciation pair is expressed by summing over all matching co-segmentations of the pair, as shown below.

$$p(g, y) = \sum_{q \epsilon\, S(g,y)} p(q)$$

where S denotes the set of all possible co-segmentations. S is represented formally as

$$S(g, y) = \left\{ q \in Q^* \middle| \frac{g_{q_1} \cup \ldots \cup g_{q_j} = g}{y_{q_j} \cup \ldots \cup y_{q_j} = y} \right\}$$

where j is the length of the sequence and ∪ is the concatenation operation.

Thus the joint probability can be expressed in terms of a n-gram model over the graphone history as shown in the equation below.

$$p(q_1^M) = \prod_{j=1}^{M+1} p\left(q_j \middle| q_1, \dots, q_{j-M+1}\right)$$

3.2.2 Training

Training is done in two phases. First, a unigram model is inferred from the training corpus. This model is used to segment the training corpus into uni-graphone chunks. Then a n-graphone model is trained over this segmented corpus.

The set of unigraphones can be inferred by Expectation Maximization (EM) algorithm.

The parameter of the model is the probability of the unigraphone q $v_q := p(q; v)$

The equations for Expectation step are shown below.

$$p(q; v) = \prod_{j=1}^{|q|} v_{q_j}$$

$$e(q; v) := \sum_{j=1}^{M} \sum_{q\epsilon\, S(g_j, y_j)} p\left(q \middle| g_j, y_j; v\right) n_q(q)$$

$e(q; v)$ is called the evidence of unigraphone q. It is the expected number of occurrences of q in the training sample and can be efficiently computed using the forward backward procedure. Then we update the parameter value as a part of the Maximization step as shown below.

$$v'_q = \frac{e(q;v)}{\sum_{q'} e(q';v)}$$

3.2.3 Testing

The phoneme prediction step uses maximum approximation approach, which means given the test word it looks for most likely graphone sequence using the standard A* algorithm and project them onto phonemes.

3.2.4 Experiments and Results

The results we report in table below are for training and test set generated by us by selecting random examples from CMUDict. Our train/test split was 90/10 percent and our test data size was 13K.

	Unigram	Bigram	Trigram	4-gram	5gram
PER(%)	45.6	23.8	14.6	11.1	10.4
WER(%)	99.5	77.8	54.7	44.4	42.5

Table 5: JMM results on CMUDict

As we can see, the system accuracy shows steady improvement as we incorporate longer histories. Also we can state that JMM performs better than CRF. JMM has an advantage of the alignment process being inherent to its working, while CRF requires alignment to be computed beforehand. Also JMM beats CRF in time taken to train the system as well as provides superior results.

3.3 Phonetisaurus based G2P.

In this section, we describe our experience with using Phonetisaurus(Novak, Minematsu, & Hirose, 2012) , a WFST based tool for G2P. Given the CMUDict training and test data consisting of word-pronunciation pairs, a typical Phonetisaurus pipeline would consist of the following stages.

1. Sequence Alignment: This module produces EM based multiple to multiple alignments. As described in (Novak, Dixon, et al., 2012), the authors improve on the approach proposed by Jiampojamarn (Jiampojamarn, Kondrak, & Sherif, 2007) .

2. Building Joint sequence model: The aligned corpus is the input to this module. This module builds an N gram model over sequences of aligned grapheme-phoneme symbols. This ARPA style LM is then converted into a Weighted Finite State Transducer (WFST) using OpenFst for decoding .

3 Decoding: The WFST generated in the previous step has grapheme chunks and phoneme chunks as its input and output alphabet respectively. The decoding unit creates a Finite State Acceptor (FSA) out of the test word (w) and composes it with the FST (C) obtained from step 2. Then the shortest path (P) computation is done on the composition to find the one best pronunciation.

$$P = BestPath\left(Project_O(w^\circ C)\right)$$

We can see that Phonetisaurus uses the Viterbi approximation to predict phonemes, as opposed to Sequitur, which uses the summation approach. We could also generate n-best lists if required.

3.3.1 Experiments and Results

We decided to generate a baseline system using Phonetisaurus and CMUDict as the dictionary. We set the maximum allowed subsequence length during alignment for graphemes and phonemes to be two. We built a 7 gram model from our aligned data which provides enough history to perform well. A Weighted Finite State Transducer (WFST) is built from the LM with the log of the conditional probability being the weights.

Baseline results for our test data are shown in the table below.

Test set size	Word Error (%)	Phoneme Error (%)	Substitutions (%)	Insertions (%)	Deletions (%)
12890	36.7	7.4	5.4	1.0	1.0

Table 6: Phonetisaurus baseline results on CMUDict

As we can see, we are getting very good results for the system with subsequence length of two and a 7gram LM. Also the Viterbi approach is proving effective in increasing accuracy as compared to JMM. A table summarizing the results of all the methods on CMUDict is shown below.

Technique	Phoneme Error Rate (%)
CRF(+2,-2)	14.0
CRF(+3,-3)	12.0
JMM-unigram	45.6
JMM-5gram	10.4
Phonetisaurus-7gram	7.4

Table 7: Summary table of results on CMUDict for all techniques

The aim of this chapter was to study the existing G2P systems before attempting to bring in speech data. We looked at three techniques for G2P and applied them to CMUDict to evaluate their performance. We find that Phonetisaurus is the best of the three in terms of results obtained. Also its modular approach makes is ideal candidate for us to introduce our own module based on a speech corpus. We will now move to the crux of this thesis, namely building acoustic G2P systems.

Chapter 4: Acoustic data driven G2P

4.1 Introduction

This chapter presents our attempts to build G2P models for English by using audio data to augment the Phonetisaurus tool. We will introduce HMM as models for representing graphemes and phonemes. Then we will detail our approach which uses a acoustic score computed using distances between HMM models of phonemes and graphemes to modify the Phonetisaurus pipeline. As mentioned before, we will work on CMUDict, an American English pronunciation dictionary provided by CMU. Our training and test set contain ~116K and ~13K randomly chosen examples from CMUDict.

Recall that Phonetisaurus pipeline consists of the following stages: Dictionary alignment, constructing a N-gram LM over the aligned G-P pairs and finally shortest path decoding to predict phonemes. These stages are shown in the figure below.

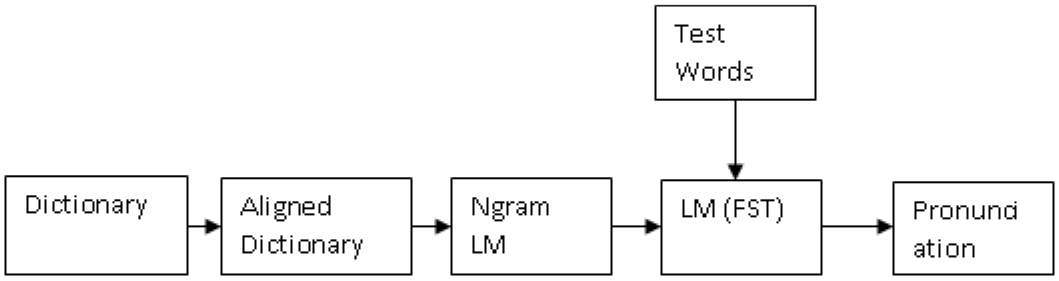

Figure 2: Block diagram of the Phonetisaurus pipeline.

We will denote the combined set of 26 graphemes and graphemes clusters obtained from the alignment as grapheme chunks. Similarly the phoneme set along with clusters are henceforth called phoneme chunks. The conditional probability of a phoneme chunk given a grapheme chunk obtained from our LM can be thought of as a linguistic score of how closely the two are related. Recently, there has been growing interest in the idea of combining information from audio sources along with the linguistic information which we have already seen how to use.(Rasipuram & Doss, 2012)(Lu, 2013) show that this combination can be particularly useful for languages with limited data. We hypothesize that we could enhance the G2P performance of existing systems if we can somehow use the audio data as a connection between graphemes and phonemes. That means we need to come up with a useful acoustic score between each grapheme-phoneme chunk pair for which we have a LM score. Figure 3 shows how this addition will fit in Phonetisaurus.

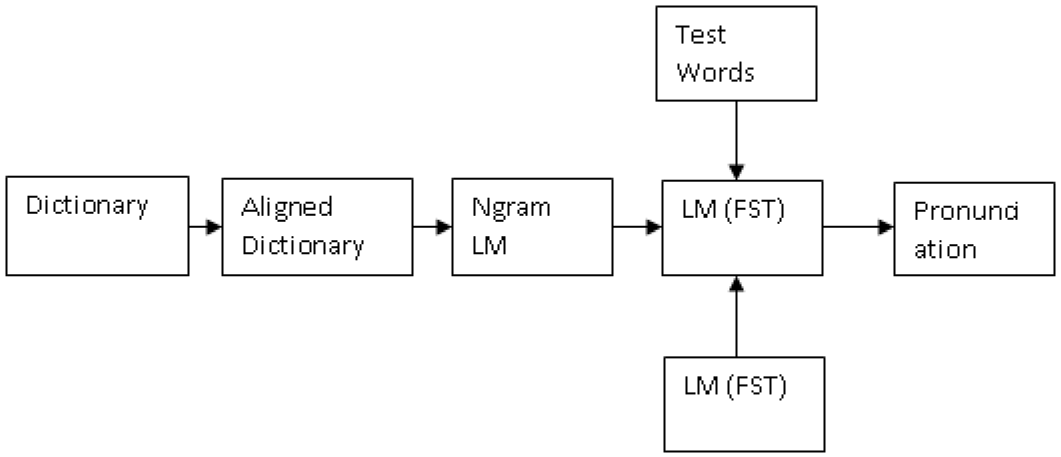

Figure 3: Block diagram of the proposed acoustic G2P system

In the following section we will discuss our method of training grapheme and phoneme acoustic models to augment phonetisaurus in order to improve its performance.

4.2 Modeling Speech Data

The speech recognition task can be described as: Given an acoustic signal X we would like to predict a corresponding word sequence W which is most likely to be generated by that signal. This is more formally expressed as follows:

$$W^{*} = argmax_{w}\, P(W|X)$$

Applying Bayes' rule

$$W^{*} = argmax_{W} \frac{P(X|W)P(W)}{P(X)}$$

As all the acoustic data is observed P(X) is a constant.

$$W^{*} = argmax_{w}\, P(X|W)P(W)$$

P(X|W) or the probability of the acoustic signal given the word is generally computed by the acoustic model and P(W) or the language model is thought of as a prior over the words. Usually we do not directly go from acoustics to words. We use sub word units to better handle factors like pronunciation variance and speaker adaptation. The formal expansion of $P(X|W)$ is as shown

$$W^{*} = argmax_{w}\, P(X|Q)P(Q|W)P(W)$$

where Q is the intermediate representation, usually in the form of phonemes.

Most speech recognition systems use phones as sub word units. Alternatively there has been interest in using the orthography itself as a basis instead of phones. (Killer, M., Stüker, S., & Schultz, 2003) demonstrates that graphemes too, can be good candidates as sub word units. Grapheme based ASR has been experimented with in the past, with phone systems in general outperforming grapheme systems for recognition task for English. This is understandable as orthography is designed for writing and does not necessarily conform with the information in the speech signal. However grapheme based systems make developing dictionaries simple, as the constituent graphemes of the pronunciation are already available. Being able to develop dictionaries relatively quickly is particularly handy as we venture into new languages and want quick development times.

4.3 Acoustic G2P

4.3.1. Building Acoustic Models

To build any acoustic model we need 4 things, a speech corpus, a base sub word unit set, a pronunciation dictionary and transcription for training and test data from the corpus. The pronunciations and transcription have to expressed in terms of the sub word units. For our experiments, we use the Wall Street Journal (WSJ) dataset(Paul & Baker, 1994), a corpus of read speech built by DARPA. Wall street journal news text were used as source material. WSJ was developed in two stages, namely WSJ0 and WSJ1. We use the training and test material from WSJ0 which contains 7139 utterances. We use CMUDict as the dictionary.

We need a method to learn the statistical properties of speech corpus we have given its labels. The most widely used technique is the Hidden Markov Model (HMM). (Rabiner, 1989) gives a very good tutorial on fundamentals of HMM is a directed graphical model ideal for modeling speech data. For our experiments, we use HTK, which a powerful open source toolkit developed at Cambridge university for building speech recognizers using HMMs.

4.3.2. Preparing Data for HMM training

In case of phoneme models, we can directly use the WSJ0 transcriptions and CMUDict as it is. We remove the stress markers in CMUDict as for G2P as we only need the possible phonetic expressions of a words. We use the set of 39 phonemes as described by ARPAbet as the sub word units. In case of phoneme chunks we decided to concatenate the trained HMMs of individual phonemes that make up the chunk.

In case of graphemes however, we need to decide on the list of sub word units (grapheme chunks) and process CMUDict and word level transcriptions provided in WSJ0 to get a grapheme dictionary and grapheme transcription.

For grapheme based systems, the most obvious way of generating a dictionary is to have its orthographic expansion as the pronunciation for each word, For example, $THERE \rightarrow T\ H\ E\ R\ E$. Thus we would have 26 models as the basis set. However our ultimate aim is to come up with an acoustic score for each pair of symbols in the LM derived from phonetisaurus. Since that involves clusters of size up to two, thanks to the many to many aligner, we need to add those clusters to the list of sub word units and reformat the grapheme dictionary to reflect these clusters. Example aligner output:

GUILTY g ih l t iy $\rightarrow$ G|U}g I}ih L}l T}t Y}iy

We end up with base set 422 grapheme units which is significantly higher than the number of units typically used for acoustic modeling. This raises the question of whether we have sufficient speech data to characterize these units. To address this problem we prepare the grapheme transcription for WSJ0 by expanding the available word level transcription in terms of the 422 symbols. Then we only keep chunks which appear more than 100 times in the transcription. We split the remaining low frequency chunks seen in both transcription and the dictionary into individual letters. The final grapheme chunk set contains 107 symbols. Now we are ready to train HMMs.

4.3.3. HMM Training

For the phone based systems, we trained a standard monophone HMM system with 16 mixtures without considering the phone clusters discovered in Phonetisaurus. For graphemes we use the refined set of 107 graphemes, each represented by a 3 state HMM with 16 Gaussians for each state. The following discussion is applicable for both the phone and grapheme systems. For the purpose of this section, we will use the word 'symbol' as a general term instead of using grapheme/phone. The basic steps followed are

1. Create single Gaussian models for all the monosymbols and silence , trained using the transcriptions.

2. Add a single state short pause model which is tied to the centre state of the silence model

3. Use a mixture splitting procedure provided by HTK to re-train the monosymbol models by a step by step increase in the number of mixtures. In our experiments, we use a maximum of 16 mixtures.

4.3.4. HMM Distance Calculation

As mentioned in the end of section 4.2, in order to use speech data, we need to add some kind of acoustic score component which complements the LM probabilities. (Jyothi & Fosler-lussier, 2009) use the distance between phoneme HMMs to predict speech recognition errors. We use it as a guideline to compute the distance between the HMMs representing the graphemes and phonemes as the acoustic score. To represent the phoneme chunks we simply concatenate the 3 state representations of individual phones. Thus the clusters will be represented by a 6 state HMM. Similarly for the grapheme chunks in LM but not in acoustic modeling, we resort to using a 6 state HMM.

4.3.5. HMM state alignment

Notice that a grapheme-phoneme HMM pair does not have a unique state alignment, as both graphemes and phonemes could have 3 or 6 state HMMs. Even if both HMMs have same number of states, we could have more than one way to align states. Hence before calculating a distance we need to first align the HMM states.

Some of the possible alignments are shown below. In out experiments we consider only these three alignment options.

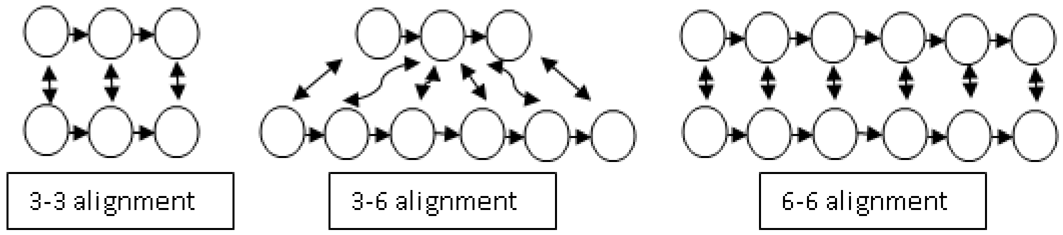

Figure 4: HMM state alignments for distance calculation

Given these configurations the distance between two HMMs is the sum of the distance between the Gaussian mixtures of aligned states, averaged by the length of the alignment. This is expressed in the equation below.

$$d_{hmm} = \frac{1}{L}\sum_{i,j} d_{GMM}(M_i, M_j)$$

Here M_i, M_j are the GMMs corresponding to the grapheme state and phoneme state respectively, while L is the length of the alignment. As the number of states for phoneme and grapheme need not be the same for 3-6 or 6-3 case, we consider the value of L to be six. d_{GMM} is calculated as the 0.5 weighted sum of the inter dispersions, normalized by the dispersion of the GMM with itself. The dispersion between two different GMMs A and B is a weighted double sum over all the distances between the monomodal Gaussian distributions (calculated using Bhattacharya distance) in both the GMMs. For more details on the calculation, we encourage the reader to refer (X. Wang & Box, 2004) and (Jyothi & Fosler-lussier, 2009).

The smaller this distance value, the more similar the HMM pair and by extension the G-P pair is. We wrote a script to calculate the distance between all pairs of grapheme-phoneme symbols observed in the LM obtained from Phonetisaurus.

4.3.6. Modifying the baseline LM

Now that we have acoustic distances, the questions remains as to exactly how we are going to integrate them in the existing LM with it. We would still need an ARPA style model file, but instead of just conditional n-gram probabilities, we want to have a mix of acoustic distance and LM log probability. This results in a new type of weight we call $w_{combined}$. For the purpose of our set of experiments we tried linear mixing of weights.

$$w_{combined} = (\propto)w_{LM} + (1-\propto)w_{Acoustic}$$

We use the SRILM toolkit to generate the new LM. We create an new 'acoustic' language model file in ARPA format with same n-gram sequences as the LM but with the negation of the acoustic distance. We need to negate the distance as the SRILM expects that field to be log of a probability, which cannot be positive. Also negating the distances does not affect our analysis in any way. The closer the G-P pair is, the distance will still approach zero (from the negative side). Once we generate the new LM with new score (lexical + acoustic), we can redo step 3 of Phonetisaurus to get new results.

4.4 Experiments and Results

To perform experiments, we curate CMUDict to remove examples which are part of the 5K dictionary used for HTK training. We want to make sure that the test data does not contain words which are a part of acoustic training. This ensures that the results reported are fair. The final training set consists of 116016 training examples an 12890 test pairs.

4.4.1 Unigram Acoustic LM Vs Unigram Pure LM

To get an idea of how the acoustic distance performs as a standalone score, we compare two systems, one built on only the unigrams of the LM used in Phonetisaurus baseline and other with a unigram LM with only acoustic scores. We did not use the full 7 gram pure LM so as to have a fair comparison between the two systems.

To get a better idea about the distribution of the scores, we plot the histogram of the acoustic scores, as shown in the figure below.

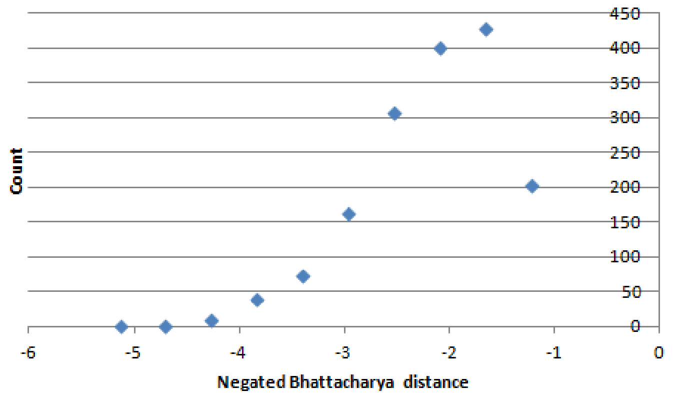

Figure 5: Histogram of acoustic scores

We also plot the histogram of LM probabilities, as shown in the figure below.

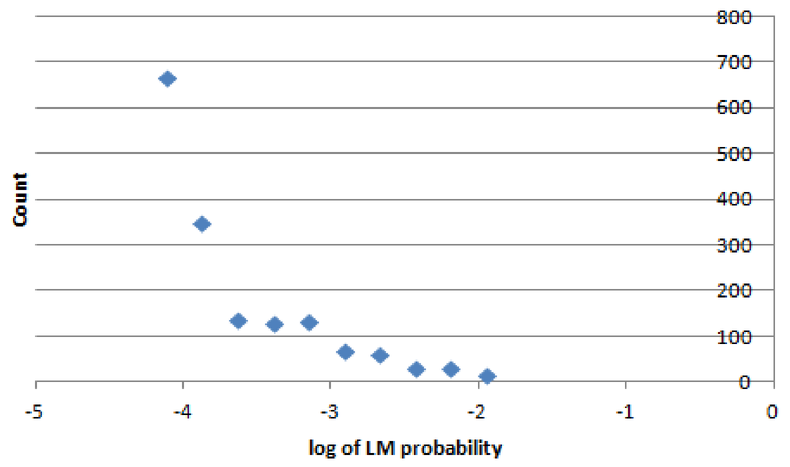

Figure 6: Histogram of LM scores

We can see that both scores have similar range of values. We can also see that both the histograms are skewed (although in different directions) to a degree and have peaks at different values. In order to better understand the relationship between the acoustic and LM scores we need to find how correlated they are.

We generated a scatter plot with the co-ordinates of a point in a plot representing the two scores for a particular G-P pair. The plot is shown in the figure below.

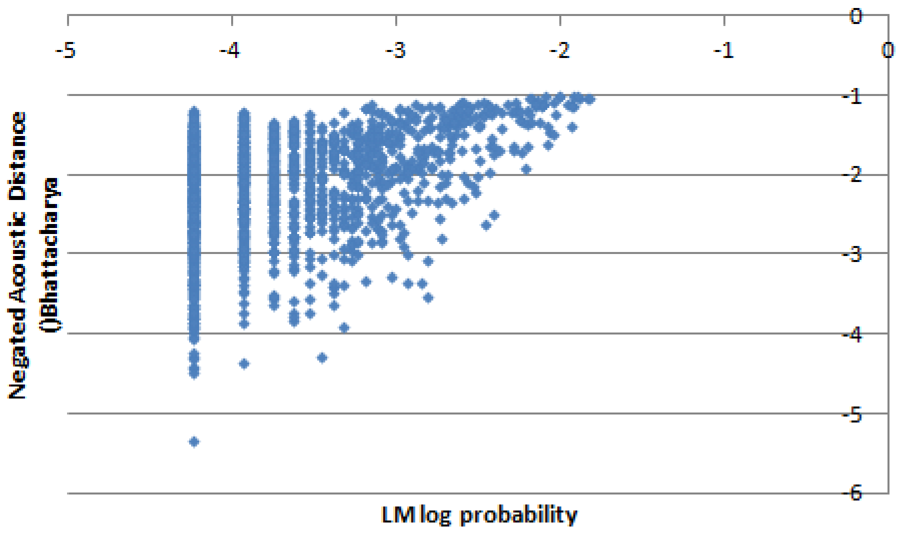

Figure 7: Scatter plot of Acoustic and LM scores

We can observe from the scatter plot that the two scores are not highly correlated. The interesting region of the scatter plot are points where we have poor LM score but strong acoustic scores. These points are indicators that the acoustic information can complement the LM scores.

The comparative results of unigram models are shown below.

	Test set size	Word Error(%)	Phoneme Error(%)	Substitutions (%)	Insertions (%)	Deletions (%)
LM unigram	12890	98.6	43.2	11.8	0.2	31.3
Acoustic unigram	12890	98.3	45.4	18.8	0.7	25.9

Table 8: Unigram acoustic system Vs Unigram LM system

We can see that acoustic distances based systems perform almost as well as pure LM. In fact we get a slight improvement in the WER for our system. There is significant reduction in deletions, which indicates acoustic distances are doing a better job of capturing relations between grapheme clusters and phonemes. Although deletion accuracy is better, it is offset by increase in substitutions. This probably means we need more speech data for discriminating between competing G-P pairs (For example, $R \rightarrow r \text{ and } R \rightarrow er$) which are leading to substitutions.

4.4.2 Acoustic unigram and Pure 7gram LM combined

In this system we get a hybrid LM by linear mix of acoustic unigram and 7gram pure LM using tools from SRILM. The procedure to accomplish this is already described in the previous section. The results of this system are compared with the baseline system, as shown in the table below.

	Test set size	Word Error (%)	Phoneme Error (%)	Substitutions (%)	Insertions (%)	Deletions (%)
Acoustic + 7 gram LM	12890	48.1	11.5	6.8	0.8	3.9
Baseline	12890	36.7	7.4	5.4	1.0	1.0

Table 9: Acoustic+LM results

Error Analysis

As we can see , this result is not as good as the baseline. Thus we need to investigate why this is happening. Let us look at the top three examples in each error category, as shown in the table below.

Substitutions	Deletions	Insertions
r/er	ah	ah
ah/ih	aa	t
Ih/ah	eh	s

Table 10: Example error stats

We observed that a r/er was the most common substitution. What was interesting was that this substitution was invariably proceeded by a deletion of 'aa' or 'eh'. Consider the following test words and their pronunciations as predicted by the Acoustic unigram and Pure 7gram LM combined system.

```
ADORED                   ARAB
 ah d ao r  d            eh r  ah b
         |                  |   |
ah - -  er d             -  er ae b
```

In the first example, we see that the mapping should have been $OR \rightarrow ao\ r$. However it outputs 'er'. To see why this is happening we need to go into the decoding process.

For decoding we perform 1 best path search on the composition of input word and the FST obtained from LM. Also keep in mind that lower the weight value, more favorable the path. Now in case of OR there are two possible branches while decoding. Now the problem is every time we have a choice between the cluster as well as the individual graphemes making the cluster, we always choose the cluster as sum of weights of the branches is always worse.

The decoding graph for 'OR' is shown below.

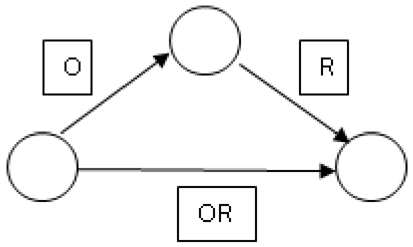

Figure 8: Decoding graph structure for 'OR'

Thus this system had a bias and it was hurting performance as substitutions and deletions were getting coupled together. We proposed a solution for this problem, which resulted in improved results. We will discuss this solution next.

4.4.3 Heuristically modified acoustic distance based system

To cancel out the bias, we need to boost the distance value for the clusters so that they will be comparable to the other branch. We created a new 'Modified' acoustic LM by applying this selective scaling heuristic. Whenever we observe a grapheme cluster during HMM distance calculation we simply multiply the distance by two.

Results for the same test set with this new heuristic is shown in the table below. We include the baseline and the Acoustic+7gram LM results again for comparison purpose.

	Test set size	Word Error (%)	Phoneme Error (%)	Substitutions (%)	Insertions (%)	Deletions (%)
Baseline	12890	36.7	7.4	5.4	1.0	1.0
Acoustic + 7 gram LM	12890	48.1	11.5	6.8	0.8	3.9
Modified Acoustic + 7gram LM	12890	46.0	10.4	6.7	1.2	3.5

Table 11: 'Modified Acoustic' system results

We can see improvements in overall accuracy over the previous system. The substitution and deletion error rate has dropped. Consider the following example of relative frequency of couple of substitution errors across all systems, as shown in the figure below.

	Phonetisaurus baseline	Acoustic + 7 gram LM	Modified Acoustic + 7gram LM
r/er substitution	219	415	266
ah/ow substitution	81	48	56

Table 12: Example error stats

The first example shows that even though the number of times r/er substitution in this system is still more than the baseline, there is a significant reduction from the previous system. This would indicate that doubling the distance is working in our favor. The second example serves to demonstrate that our system is better than baseline for some error examples. This gives us encouragement to think that with some more modifications, we could eventually beat the baseline.

Chapter 5: Conclusion

In this thesis, we investigated the feasibility of using information from speech data to possibly boost the performance of existing G2P systems. We looked at some of the existing state of art techniques for G2P like CRFs, Joint Sequence Models and Phonetisaurus. We modified Phonetisaurus to include acoustic scores and discussed the set of experiments we performed and their results in the previous section.

While our current set of results did not improve over the baseline, they came pretty close to matching the baseline performance. Also as evident from the examples in error analysis, the acoustic models show improvements over baseline in specific examples of errors. We saw further improvement when we built a new model where we tweaked the distances based on our observations of errors in test examples applied to the original model. This gives us hope that while we haven't beaten the baseline yet, this still remains an idea worth pursuing.

As we have seen across all major G2P methods, context plays an important role in getting good results. However at this point, we do not consider neighboring phoneme or grapheme chunks while calculating acoustic distances. One of the things we would like to do in future is to incorporate tri phoneme models in our system so as to have the distance between G-P pair depend on the context surrounding the phone as well. The HTK toolkit allows us to train HMM models for context dependent tri phonemes. For example, consider the grapheme-phoneme pairs (A, aa+r) and (R, aa-r), where the symbols '+' and

'-' indicate context to the right and left respectively. We believe that these distances will provide a better representation of the closeness that (A,aa) and (R, r). Consequently, we hypothesize that this could play an important role in getting better results in the future. Another potential area for improvement is in the process of calculation of the acoustic score. For example, given the HMM states between the grapheme-phoneme pair, we consider only one alignment between the states to calculate our score. We would like to extend this to a case where we calculate scores over all possible state alignments and either take their sum and average over the number of alignments, or simply take the minimum value as the final score.

It would be interesting to apply our technique to different languages and compare its performance in relation to orthographic depth of those languages.

Bibliography

Bartlett, S., Kondrak, G., & Cherry, C. (2008). Automatic Syllabification with Structured SVMs for Letter-to-Phoneme Conversion. In *Association for Computational Linguistics* (pp. 568–576).

Bisani, M, & Ney, H. (2002). Investigation on joint-multigram models for grapheme to phoneme conversion. In *INTERSPEECH*.

Bisani, Maximilian, & Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, *50*(5), 434–451.

Borgwaldt, S. R., Hellwig, F. M., & Groot, A. M. B. De. (2005). Onset Entropy Matters – Letter-to-phoneme Mappings in Seven Languages. *Reading and Writing*, *18*(3), 211–229.

Casacuberta, F., & Vidal, E. (2007). GIZA ++ : Training of statistical translation models.

Chen, S. F. (2003). Conditional and Joint Models for Grapheme-to-Phoneme Conversion. In *INTERSPEECH*.

Deligne, S., Yvon, F., & Bimbot, F. (1995). Variable-Length Sequence Matching For Phonetic Transcription Using Joint Multigrams. *Speech Communication*, *23*(2), 223–241.

Hahn, S., Vozila, P., & Bisani, M. (2012). Comparison of Grapheme-to-Phoneme Methods on Large Pronunciation Dictionaries and LVCSR Tasks. In *INTERSPEECH*.

Hain, T., Burget, L., Dines, J., Garau, G., Kara, M., Lincoln, M., … Wan, V. (2006). The AMI Meeting Transcription System : Progress and Performance. In *Machine learning for multimodal interaction* (pp. 419–433).

Jiampojamarn, S., Kondrak, G., & Sherif, T. (2007). Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion. In *HLT-NAACL* (pp. 372–379).

Jyothi, P., & Fosler-lussier, E. (2009). A Comparison of Audio-free Speech Recognition Error Prediction Methods. In *INTERSPEECH* (pp. 1211–1214).

Kaplan, R. M., & Kay, M. (1994). Regular Models of Phonological Rule Systems. *Computational linguistics*, *20*(3), 331–378.

Killer, M., Stüker, S., & Schultz, T. (2003). Grapheme Based Speech Recognition. In *INTERSPEECH*.

Kominek, J., & Black, A. W. (2006). Learning Pronunciation Dictionaries:Language Complexity and Word Selection Strategies. In *Association for Computational Linguistics* (pp. 232–239).

Kudo, T. (2005). CRF++: Yet another CRF toolkit. User's manual and implementation available at http://crfpp. googlecode. com/svn/trunk/doc/index. html.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the International Conference on Machine Learning (ICML'01)* (pp. 282–289).

Lu, L. (2013). Acoustic data driven pronunciation lexicon for speech recognition. Retrieved from http://www.natural-speech technology.org/sites/default/files/liang_nst_2013.pdf

Magimai, M., Rasipuram, R., Aradilla, G., & Bourlard, H. (2011). Grapheme Based Automatic Speech Recognition using KL-HMM. In *INTERSPEECH* (Vol. 1).

Novak, J. R., Dixon, P. R., Minematsu, N., Hirose, K., Hori, C., & Kashioka, H. (2012). Improving WFST-based G2P Conversion with Alignment Constraints and RNNLM N-best Rescoring. In *INTERSPEECH*. (pp. 1–4).

Novak, J. R., Minematsu, N., & Hirose, K. (2012). WFST-based Grapheme-to-Phoneme Conversion : Open Source Tools for Alignment , Model-Building and Decoding. In *10th International Workshop on Finite State Methods and Natural Language Processing* (pp. 45–49).

Paul, D. B., & Baker, J. M. (1994). The Design for the Wall Street Journal-based CSR Corpus. In *Proceedings of the workshop on Speech and Natural Language* (pp. 357–362).

Rabiner, L. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Rasipuram, R., & Doss, M. M. (2012). Combining Acoustic Data Driven G2P and Letter-to-Sound Rules for Under Resource Lexicon Generation. In *INTERSPEECH* (pp. 6–9).

Sejnowski, T. J., & Rosenberg, C. R. (1987). Parallel Networks that Learn to Pronounce English Text. *Complex Systems 1*, *1*(1), 145–168.

Taylor, P. (2005). Hidden Markov Models for Grapheme to Phoneme Conversion. In *INTERSPEECH* (pp. 2–5).

Toutanova, K., & Moore, R. C. (2002). Pronunciation Modeling for Improved Spelling Correction. In *Association for Computational Linguistics* (pp. 144–151).

Wang, D., & King, S. (2011). Letter-to-Sound Pronunciation Prediction Using Conditional Random Fields. *Signal Processing Letters, IEEE*, *18*(2), 122–125.

Wang, X., & Box, P. O. (2004). A GMM-Based Telephone Channel Classification For Mandarin Speech Recognition. *Signal Processing*, *IEEE*, *1*, 642–645.

Weide, R. (2005). The Carnegie mellon pronouncing dictionary [cmudict. 0.6]. Retrieved from http://www.speech.cs.cmu.edu/cgi-bin/cmudict