PRONUNCIATION MODELING FOR SYNTHESIS OF LOW RESOURCE LANGUAGES

SUNAYANA SITARAM

CMU-LTI-15-017

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

ABSTRACT

Natural and intelligible Text to Speech (TTS) systems exist for a number of languages in the world today. However, there are many languages of the world, for which building TTS systems is still prohibitive, due to the lack of linguistic resources and data. Some of these languages are spoken by a large population of the world. Others are primarily spoken languages, or languages with large non-literate populations, which could benefit from speech-based systems.

One of the bottlenecks in creating TTS systems in new languages is designing a frontend, which includes creating a phone set, lexicon and letter to sound rules, which contribute to the pronunciation of the system.

In this thesis, we use acoustics and cross-lingual models and techniques using higher resource languages to improve the pronunciation of TTS systems in low resource languages. First, we present a grapheme-based framework that can be used to build TTS systems for most languages of the world that have a written form. Such systems either treat graphemes as phonemes or assign a single pronunciation to each grapheme, which may not be completely accurate for languages with ambiguities in their written forms.

We improve the pronunciation of grapheme-based voices implicitly by using better modeling techniques. We automatically discover letter-to-sound rules such as schwa deletion using related higher resource languages. We also disambiguate homographs in lexicons in dialects of Arabic to improve the pronunciation of TTS systems. We show that phoneme-like features derived using Articulatory Features may be useful for improving grapheme-based voices.

We present a preliminary framework addressing the problem of synthesizing Code Mixed text found often in Social Media. Lastly, we use acoustics and cross-lingual techniques to automatically derive written forms for building TTS systems for languages without a standardized orthography.

ACKNOWLEDGMENTS

I was very lucky to have Alan Black as my PhD advisor for the last three years. I met Alan in 2007 when I visited CMU for the OurCS conference and he showed me a demo of the Let's Go bus system and a speech-to-speech translation system, which sparked my interest in speech processing. Ever since then, I wanted to work with him and am very grateful to have got the opportunity to have him as my advisor. Alan is an inspiration for everyone around, with his passion for his work and enthusiasm and I've really enjoyed all the discussions I've had with him. I hope to continue working with him and learning from him for a long time.

Thank you to Richard Sproat, Lori Levin and Florian Metze for being excellent committee members. I really appreciate all the detailed feedback I got about my work and thesis document. Special thanks to them also for making scheduling the proposal and defense a breeze by being very flexible about dates and times, despite my tight time constraints for the defense.

I joined CMU in 2010 as a Masters student, and my Masters advisor was Jack Mostow. I'm very thankful to him for teaching me how to do research, and also helping me improve my writing and presentation style. I really enjoyed working with him and the entire Project LISTEN team.

Thanks also to Carol Freize and Mary Widom for organizing OurCS and giving me the chance to visit CMU as an undergrad. I really appreciate all their encouragement over the last few years and their efforts to create a community for women in the School of Computer Science.

I had the chance to intern with Microsoft Research India in 2012. Thank you to my mentors at MSRI - Kalika Bali and Ed Cutrell, and the rest of the folks there who made my internship memorable. I look forward to working with you soon.

Thank you also to Prof. Hemant Patil for having been a great mentor since 2008. I have enjoyed visiting DAIICT over the last few years and interacting with his students, and hope to continue doing so in the future.

Thank you the the staff at the LTI, especially Stacey Young for making sure everything runs smoothly.

I was very fortunate to meet many people at CMU who went on to become very good friends and colleagues, many of whom I'm sure I will forget to name here. A big thank you to Alok and Aasish for being a lot of fun to hang out with, and also for all the advice they have given me over the last few years. Prasanna and Pallavi have been wonderful friends and research group mates. Thank you also to Gopala Anumanchipalli, Serena Jeblee, Justin Chiu, Sukhada Palkar, Vivian Chen, Yulia Tsvetkov, Ming Sun, Yuanpeng Li and Joe Valeri for being great colleagues and co-authors.

Thank you to the Hobart gang Meghana, Bhavana and Ruta for all the fun times we have had and for being there for me through all these years (and all the great food!). Special thanks to Anjali for being an excellent roommate - I will really miss watching bad movies with you. Thank you to my dear friends over the years Derry, Priyanka, Athula, Kriti, Hyeju, Rose, Archna and Anuva, for all the great get-togethers and conversations. Thank you also to my friends back home - Mansi, Sandip, Harit, Vaishnavi, Grishma and Riddhi for all your love. Thank you to my partner in crime Ravi Mohan for doing daily work logs with me and helping me be productive. I hope to see you all very soon!

I'd like to thank my parents for always encouraging me for pursuing my dreams, no matter how unconventional the path I took was. My father played an important role in getting me interested in research and my mother was always there to talk to me about everything. I'd like to thank the rest of my family for their love and encouragement, particularly my grandmother, who is a source of inspiration for everyone, Shruti and Mohan for providing me with a home in New York that I could always go to when I felt homesick, and my niece and nephew Siri and Rishi, who were both born after the start of my PhD, for giving me a sense of perspective on time. Special thanks also to Sharat and Meghana for their love and thoughtfulness. My fiancé's parents and family were also very supportive of my PhD, which I am very grateful for.

My PhD would not have been possible without the love, support and constant encouragement from my fiancé Sandeep. Thank you for being with me every step of the way over the last seven years, from across continents and time zones. You have played such an important role in making sure I stay sane and happy over the last few years. This thesis is as much yours as it is mine.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Intelligible and natural sounding Text-to-Speech (TTS) systems exist for a number of languages of the world today, and there are automatic techniques to build good quality systems quickly for languages with sufficient data and linguistic resources.

By using a few hours of well recorded speech data, corresponding transcripts and a lexicon or grapheme-to-phoneme (g2p) rules, an understandable TTS system can be built using freely available software resources. However, many languages of the world do not have such data and lexical resources available. Some of these languages are the ones where communication using speech may be extremely beneficial, since in many cases these languages may primarily be spoken languages or have a large non-literate user population.

One of the most straightforward ways to build a TTS system in a language without many lexical resources is to build grapheme-based systems [1] in which very few assumptions may be made about the letter-to-sound rules in the language. In the past, there have also been attempts to use unsupervised techniques to be able to build linguistic resources automatically. For languages that have low resources, it has been shown that we can train high level components like parsers, POS taggers etc in an unsupervised manner using textual cues [2]. Most of these techniques assume that the orthography of the language is unambiguous and that the speech transcript is available. All of these techniques assume that the language has a standardized written form.

In some languages of the word such as Arabic and Hebrew, missing diacritics in the written form create ambiguity in pronunciation for speech synthesizers, both during TTS system training or labeling time and during synthesis. In higher resource dialects of such languages, such as Modern Standard Arabic, diacritizers exist which solve this problem to a certain extent by recovering the diacritics of a word automatically. However, in many of the lower resource dialects, such systems do not exist. The problem of underspecified written forms can be thought of as a general issue that happens to a certain extent in all languages, such as missing stress markers in written English, schwa deletion markers in Hindi, tone-markers in tonal languages and homographs. When this happens very frequently in languages and is not dealt with by a TTS system, intelligibility may be affected.

Instead of looking for specific pronunciation nuances in languages, it would be useful to use general features to improve the pronunciation of low resource or grapheme-based systems. In addition, instead of going from a written representation to an approximate phonetic form, it may be useful to use features that can be derived from the acoustics in a bottom-up manner. Such features may require no resources in the language itself other than acoustics, and can use models trained cross-lingually on higher resource languages.

Most Text to Speech systems assume that input text is in a single language, which is the target language. However, this is not always true. Code Switching or Code Mixing occurs when multiple languages are used in the same sentence. Code Switched or Code Mixed text is found in Social Media used by bilingual or multilingual societies. Synthesizing such text can be challenging due to multiple factors, including the scripts used for writing it and non-standard spellings.

Finally, many languages of the world do not have their own standardized written forms. Building speech systems, such as Speech to Speech Translation and Spoken Dialog Systems may be extremely useful in such languages. However, due to the absence of a writing system, traditional approaches to building such systems may not work. Coming up with a written form that is sufficiently discriminative to be able to produce understandable synthesis, so that it can be used internally in such a system, may be a useful goal.

This thesis aims to discover techniques to be able to build intelligible and natural sounding TTS systems for low-resource languages, by modeling pronunciation in those languages without having to resort to a lot of manually created resources.

1.1 THESIS ORGANIZATION

The thesis is organized as follows. Chapter 2 introduces the problem of grapheme-to-phoneme conversion in Text to Speech Systems and also describes the resources needed to build a Text to Speech system in a new language. In addition, we describe a universal grapheme-based framework to build systems in new languages.

Chapter 3 describes techniques to improve upon the grapheme-based baseline, by using better modeling techniques to make soft decisions about pronunciation and by using cross-lingual models from very related languages to learn specific letter to sound rules.

Chapter 4 describes general techniques to improve grapheme-based and low rescource voices using cross-lingual articulatory features in a bottom-up manner.

Chapter 5 describes techniques to disambiguate homographs in languages with ambiguous written forms.

Chapter 6 describes how to synthesize text in code mixed languages using a TTS database trained on one language.

Chapter 7 describes how to automatically derive a written form for languages without a standardized written form in order to build TTS systems.

Chapter 8 concludes with future directions.

1.2 THESIS STATEMENT

The pronunciation of synthesized voices built for languages without many lexical resources available can be improved by leveraging cross-lingual models of related higher resource languages and the acoustics of the TTS database.

1.3 THESIS CONTRIBUTIONS

- A baseline for universal grapheme-based synthesis covering most writing systems of the world, and techniques to improve grapheme-based systems
- Techniques to disambiguate homographs in languages with ambiguous written forms such as Arabic
- An analysis of cross-lingual articulatory features to improve pronunciation of grapheme-based voices
- A framework to improve pronunciation for synthesizing code mixed text
- Discovering written forms to synthesize from, for languages with no standardized writing systems
- Speech Synthesis for a wide variety of languages from many language families and having varied written forms, such as English, Indian languages, dialects of Arabic, European languages

such as German and Russian, endangered languages such as Inupiaq and Ojibwe and languages with no standardized written form, such as Konkani.

2 GRAPHEME-BASED SYSTEMS

2.1 INTRODUCTION

Text to Speech systems in use today are usually one of two main types: Unit Selection and Statistical Parametric Synthesis. In Unit Selection synthesis [3], actual pieces of speech are chosen from a unit database and concatenated together to produce a target utterance. In Statistical Parametric Synthesis [4], each unit of speech is modeled by a parametric representation, which is then predicted at synthesis time from the target utterance.

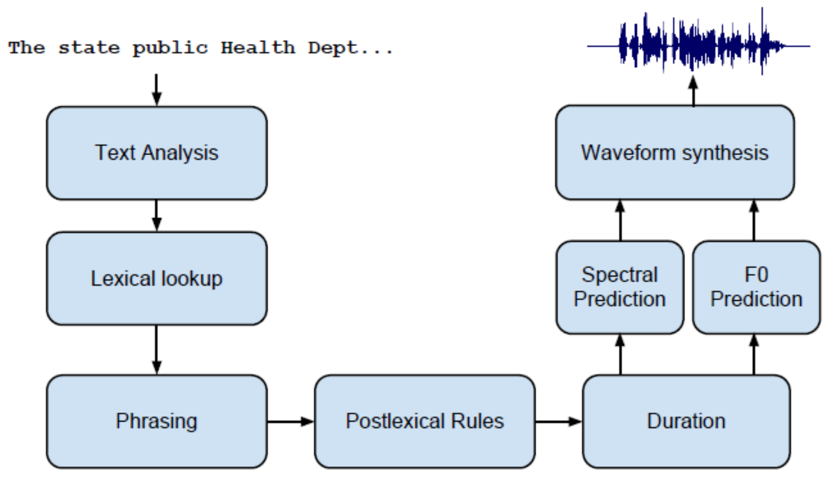

Figure 1: TTS overview [5]

In both these types of systems, at synthesis time, text input is analyzed by expanding abbreviations, dealing with numbers etc. [6] and converting the text into a sequence of tokens. A phrasing model inserts pauses in appropriate places in the utterance [7]. Then, the sequence of tokens is converted into a string of phonemes by consulting a lexicon or letter-to-sound (LTS) rules. Then, the appropriate units or parameters are chosen by the trained machine learning model based on features of the phonemes, words and phrase such as position, context, Part of Speech (POS). Lastly, units of speech are concatenated, or generated from parameters by using inverse filters [8]. Figure 1 shows the text to speech pipeline for a Statistical Parametric Synthesis system [5].

In general, the front-end of the system, that includes the text processing, lexical lookup or letter-to-sound module are the language specific parts of a TTS system. In addition, other resources such as Part of Speech (POS) taggers and parsers may be used as features in spectral or prosody models. At a minimum, to build a TTS system in a new language, the text processing, lexical lookup and LTS modules are necessary.

Text processing is a critical step in synthesizing text, particularly if the text is not being generated by the system itself, such as in a dialog system, but is found on sources such as the World Wide Web. Text processing has generally been dealt with using systems that classify text into a set of categories and expand text using rules specific to that type, such as abbreviations, numbers and dates [9]. Such rules usually need to be created for each language independently and may require input from a native speaker. Other approaches have explored the use of supervised and unsupervised techniques for text processing in new domains[6]. The challenges posed by processing text in new, low resource languages are beyond the scope of this thesis.

In this thesis, we focus on the lexical lookup and letter-to-sound blocks of the system shown above, that can be thought of as the components determining the pronunciation of words. However, pronunciation goes beyond phoneme choice in most languages. We can break up components of speech that relate to the way a word is pronounced as follows:

- phoneme selection
- Duration, realized as phoneme length
- Power, realized as lexical stress and accents
- Intonation, realized as tones in languages such as Mandarin

A high-resource language such as US English may have a freely-available lexicon such as CMUdict [10] that contains phoneme expansions and stress markers for a large vocabulary in the language. In addition, we can train letter-to-sound models from such a lexicon to predict the pronunciation of unknown words. Such resources may not be available in many languages.

2.2 TTS EVALUATION TECHNIQUES

Speech synthesis is evaluated both objectively and subjectively. In objective evaluations, synthetic speech is usually compared to reference speech, or features derived from synthetic speech are used to measure synthesis quality. In subjective evaluations, humans are asked to listen to one or multiple systems, and rate or compare them on a variety of metrics.

2.2.1 *Objective evaluation metrics*

There has been a lot of interest in coming up with better objective measures for TTS quality, since subjective tests are expensive and time consuming. The main challenge is to find objective measures that correlate with subjective metrics, since the ultimate goal of TTS systems is to sound understandable and intelligible to users. Also, some objective measures need natural speech for the reference utterance which can be an obstacle.

Perceptual Evaluation of Speech Quality (PESQ) [11] is a metric that was originally developed for the evaluation of telecommunication and has been suggested as an objective metric for TTS. It creates internal models of the target and reference speech and then calculates the difference between them. It has been shown to be correlated with subjective Mean Opinion Score (MOS) tests, which are tests in which subjects are asked to score systems based on a series of criteria. Although it can deal with target and reference speech of different lengths, it has been shown not to have a high correlation with MOS measures for a small amount of data.

Moller and Falk [12] compared objective metrics based on features such as low SNR (Signal to Noise Ratio), naturalness of the voice, robotization etc. calculated from the synthesized signal automatically similar to the PESQ framework. These metrics do not require reference natural speech and were found to have a reasonable correlation with subjective measures.

In work by Chu et al. [13], an average concatenative cost function calculated during unit selection is proposed as an objective measure of naturalness of synthetic speech. This cost function is shown to be highly correlated with a MOS of naturalness and does not require reference speech. The technique described may have large errors while estimating MOS from the concatenative cost function. Hu and Loizou [14] used linear combinations of objective measures to calculate subjective evaluation scores.

Valentini et al [15] compared four objective measures Dau, Glimpse, Speech Intelligibility Index (SII) and PESQ for correlation with perceptual tests of intelligibility for the speech in noise condition. The

Dau measure is based on the human auditory system and calculates the correlation between models of the target and reference signals. The Glimpse measure calculates the proportion of speech that has more energy than noise. The Speech Intelligibility Index calculates the weighted Signal to Noise Ratio to estimate intelligibility. The Dau and Glimpse measures were found to be the best predictors of intelligibility.

Recently, Ullman et al. [16] cast the problem of TTS objective evaluation for intelligibility as an utterance verification problem, for which there does not need to be any reference speech. This has been shown to have a high correlation with subjective intelligibility judgments.

2.2.2 *Mel Cepstral Distortion*

The Mel Cepstral Distortion[17] is calculated as a Euclidean norm between two vectors - a vector representing the target (typically the synthesized utterance) and the reference (typically natural speech). For TTS, these vectors are typically 25 dimension Mel Frequency Cepstral Coefficients (MFCC) at intervals of 5 ms, which are called frames.

This calculation assumes that the number of frames in the target and reference waveforms is the same. This constraint is applied by using the reference sentence as a template during re-synthesis, so that the durations of all phones and pauses are preserved in the synthesized utterance.

However, in case we change the durations or number of breaks, as in [18] or add, remove or substitute phones in our synthesized utterance when compared with the original, we cannot use the simple version of the MCD score any more. One alternative is to relabel the original waveform with new labels for the new pronunciations - however, we may want to compare two or more synthetic waveforms with different pronunciations with a reference, in which case relabeling the data is not the ideal solution. In such cases, we can use the DTW metric.

Dynamic Time Warping (DTW) is a technique to time-align two streams of data that are not time aligned. Xu [19] suggested the use of DTW to calculate a distance metric between reference and target utterance for TTS that may not be the same length and showed that it was correlated with subjective measures of synthesis quality. However, the limitations of this technique are that it only provides an utterance level score and depends on the alignment accuracy to calculate a reliable score.

Typically, the MCD is calculated on a frame-by-frame basis and then averaged over the entire database, which is what we have used to report the objective quality of our systems throughout this thesis. One of the limitations of using MCD as an objective metric is that it does not apply to Unit Selection systems, in which units of speech are taken from the database and joined together.

2.2.3 *Subjective evaluation*

Popular subjective metrics for TTS include A/B, AXB and Mean Opinion Score (MOS) tests which mainly test naturalness and user preference. In A/B and AXB preference tests, subjects are asked to listen to audio clips generated by two systems and asked to choose the clip they prefer, with a no difference option, while controlling for the order that the clips are systems are shown. These judgments are then averaged across all the examples played. In transcription tests, listeners may be asked to transcribe synthesized speech, and a Word Error Rate like metric can be used to evaluate the intelligibility of the system.

Tian et al. [20] describe techniques to evaluate a Mandarin TTS system subjectively in a modular fashion. They divide the TTS pipeline into text, prosody and acoustic components and carry out subjective evaluations at each stage by using the MOS metric. The *text* component contains word segmentation, POS tagging and character-to-syllable conversion which are all rated by a team of evaluators based on reference values. For the prosody component, a delexicalized waveform is played and evaluators choose from a number of options for the sentence that should correspond to it. The acoustic processing module contains both the text and prosody aspects and evaluators rate it on a scale, using MOS.

Hirst et al. [21] describe a subjective evaluation metric for evaluating prosody in which subjects are asked to highlight parts of sentences that they do not feel are satisfactory, and are asked to pay attention to the way the sentence is spoken while making judgments.

Most subjective tests in TTS deal with evaluating synthetic speech at a sentence level for intelligibility and naturalness. Although incorrect pronunciation affects intelligibility and naturalness, pronunciation differences between two systems may be subtle enough not to be perceived very easily in such generic listening tests, as described in

Chapter 4 in our listening tests for Iraqi and Modern Standard Arabic.

It is challenging to determine MCD or MOS thresholds to indicate that a system is natural sounding and usable across databases, languages and synthesis techniques. Clark et al. [22] present an analysis of multidimensional scaling of MOS for the Blizzard Challenge data, which shows that the dimensions represent discontinuous joins and 'roboticity'.

2.3 LANGUAGE RESOURCES

According to Ethnologue [23] there are around 7000 living languages in the world today. Most of these languages are only spoken languages and do not have standardized writing systems. Although there has been a lot of work over the last few decades on building speech and language resources and systems in many languages of the world, usable systems only exist for the top few languages. Many of the languages for which systems and resources do not exist are ones with high population, such as Bengali, which has 200 million speakers.

From the point of view of speech processing, specifically from a TTS system perspective, we classify languages into three types: high resource, medium resource and low resource. High resource languages like English and Modern Standard Arabic may have high quality Acoustic Models available, large clean speech databases of recordings for TTS, POS taggers, dependency parsers, hand-created lexicons and text processing front-ends. High quality, natural and understandable TTS systems may already exist for such languages.

Medium resource languages like Iraqi Arabic and Hindi may include lexicons with ambiguities, some grapheme-to-phoneme rules and a small amount of data to train Acoustic Models. It may be possible to build understandable TTS systems in such languages, although there may be issues with pronunciation and naturalness.

Low resource languages such as Bengali may have a large number of speakers, but may not have clean or labeled data to train models. In addition, in endangered languages such as Inupiaq, we may not have access to native speakers who can provide information or test systems for us.

Figure 2 highlights the resources that are typically used to build a TTS system in a new language. As we can see, the back end of the TTS system is mostly-language independent. Models for the spectrum, duration and prosody are typically trained automatically using data.

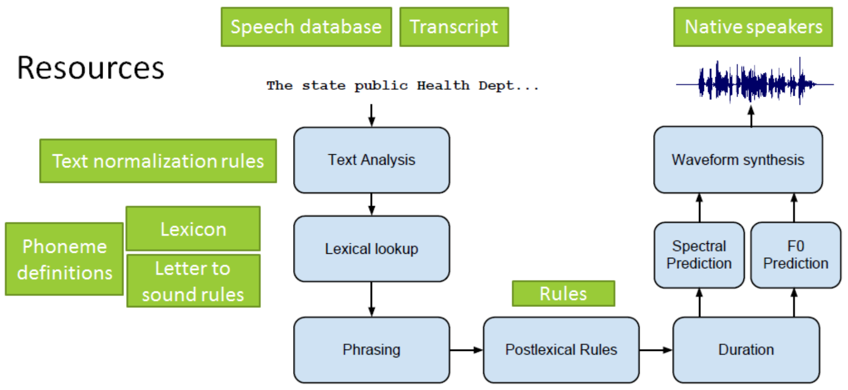

Figure 2: Resources used to build TTS systems

This thesis aims at improving synthesis quality of medium to low resource languages while relying on minimal amounts of manually created resources. Specifically, we focus on improving the pronunciation of TTS systems built in such languages, which in turn may influence the intelligibility and naturalness of these systems. To test whether our techniques work, we perform some experiments on high resource languages as well, however, we do provide results wherever possible for actual low resource languages as well.

Grapheme-based systems can be used to build baseline TTS systems in languages that have standardized written forms, and are especially good for languages where the relationship between graphemes (or letters) and phonemes follows well defined rules[1]. In the rest of this chapter, we describe our framework for building grapheme-based systems.

2.4 GRAPHEME-BASED SYSTEMS

2.4.1 *Introduction*

In very low-resource languages where there may no information about phonemes in the language, we can make the simplifying assumption that each grapheme is a phoneme. Black et al. [24] introduced this technique to build systems for low-resource languages without a phone set. While this technique requires no language specific resources, we also lose out on information about knowing what the phonemes mean, in the form of phonetic features.

In the Festvox voice building tools [25], which we use as our voice building framework for this thesis, phone sets are defined with phonetic features like vowel or consonant, consonant type, consonant place, vowel length, aspiration etc. These features are used by Classification and Regression Trees to model the spectrum and prosody.

We refer to the technique of treating each grapheme as a phoneme and ignoring the values of these phonetic features as the 'Raw Graphemes' method.

Grapheme-based systems have been used to build speech recognizers by using graphemes as units instead of phonemes for English, German and Spanish [26]. It was found that grapheme-based recognizers performed better for languages that had a close grapheme-phoneme relationship.

Another way of building grapheme-based voices is to exploit resources that transliterate graphemes into known phonemes. The Unicode specification [27] provides a standardized digital representation of scripts in most languages of the world. Qian et. al [28] have developed a toolkit that converts UTF-8 encoded text into a guessed phonetic transcription in the WorldBet[29] or X-Sampa. It supports about 40 different character code pages in the Unicode specification, excluding Latin, which we added support for, and Chinese characters. UniTran is distributed with the Natural Language Toolkit[30] for Python. We make use of this tool for grapheme-to-phoneme mappings and call it the 'UniTran' method.

Intuitively, languages with consistent and one-to-one mappings between graphemes and phonemes should be easier to model using grapheme-based methods. To test this hypothesis, we built grapheme-based voices for twelve languages from disparate language families, scripts, with varying amounts of resources available in the language and varying database sizes.

In some cases when we had manually created resources available (lexicons and letter to sound rules) , we also built 'knowledge-based' voices.

We performed all the experiments in this thesis in the context of the Festival[31] speech synthesis engine. Unless stated otherwise, we built statistical parametric synthesis models using the CLUSTERGEN [32] framework and the Festvox [25] suite of tools throughout this thesis.

2.4.2 *Data and Resources*

Table 1 shows the amount of speech data in minutes that we used to build synthetic voices for each of these languages. In the European family of languages, we did experiments with English and German. For English, we used the ARCTIC [33] recordings of the speaker SLT and for German, we used a similar corpus that we recorded locally. We built knowledge-based voices for English and German by using the standard Festival front-end for them, which included lexicons and Letter to Sound (LTS) rules for unknown words.

Hindi and Konkani Indian languages from the Indo-Aryan language family, and Tamil is an Indian language from the Dravidian family. To build our synthetic voices in Hindi and Tamil, we used the IIIT-H Indic Databases [34]. For our Hindi and Tamil knowledge-based voices, we used grapheme-to-phoneme (g2p) mappings for all the characters and added post-lexical rules for nasalized vowels. For Hindi, we added rules for terminal and medial schwa deletion. For Tamil, we added rules for contextual voicing of consonants.

Konkani is the official language of the Indian state of Goa, and is a minority language in a few other states. It has around 8 million native speakers and uses scripts such as Latin, Kannada, Malayalam and even Arabic. We used a corpus of Konkani from the CMU SPICE project[35] that used the Latin script.

Iraqi Arabic, Dari and Pashto are languages that all use the Arabic script in their written forms. The corpora we used for Iraqi, Dari and Pashto were used for building synthesizers as part of the DARPA TRANSTAC project. For Iraqi, we used a lexicon that had multiple pronunciation variants for most words, since the diacritics that correspond to short vowels are not written in the script. We used the first variant of the word in the lexicon to build our knowledge based voice.

We collected our Russian and Thai speech corpora from the SPICE[35] dataset. Inupiaq is a Inuit language spoken by about 2100 people in northern and north western Alaska, and it uses the Latin+ script as its orthography. Ojibwe is an indigenous language of the Algonquian family and is native to Canada and the United States. It has about 56000 native speakers and uses the Latin script in its written form. Our corpus for Inupiaq and Ojibwe was collected as part of the Endangered Languages project at Carnegie Mellon University.

As we can see in Table 1, the data for our languages varied from as little as 5 minutes of recorded speech for languages like Konkani and Inupiaq to over an hour of speech for English and Dari. In addition,

Table 1: *Data available for different languages*

Language	Duration (minutes)	Script
English	66	Latin
Dari	63	Arabic
Hindi	56	Devanagari
Iraqi	61	Arabic
Pashto	56	Arabic
German	53	Latin
Tamil	41	Tamil
Thai	25	Thai
Ojibwe	12	Latin
Russian	6	Cyrillic
Konkani	5	Latin
Inupiaq	5	Latin

the quality of recordings was also not uniform across the databases, with Russian being significantly worse than the other databases. We did this in order to mimic real-world scenarios of data availability for low-resource languages and to measure the gains our techniques would provide across different data sizes.

2.4.3 *Evaluation*

In order to compare the different grapheme to phoneme conversion strategies, we built full synthetic voices out of them. We held out 10% of the data during voice building. On this data, we compared the synthetic speech with reference recorded speech by looking at the Mean Mel Cepstral Distortion[36] (MCD) of the predicted cepstra. Since this is a distance measure, a lower value suggests better synthesis. Kominek [37] has suggested that MCD is linked to perceptual improvement in the intelligibility of synthesis, and that an improvement of about 0.08 is perceptually significant and an improvement of 0.12 is equivalent to doubling the data. The values in bold in the table indicate the lowest MCD for that language.

We see that in most languages, using UniTran, which gives us an approximate phonetic mapping and phonetic features, leads to an improvement in MCD. Dari and Pashto UniTran voices are worse than raw graphemes, probably because UniTran provides a single mapping for the Arabic script, which may not be appropriate for these

Table 2: *MCD for languages built with raw graphemes, UniTran and knowledge (when available)*

Language	Raw	UniTran	Knowledge Based
English	5.23	5.11	**4.79**
Dari	**4.78**	4.95	
Hindi	5.10	5.05	**4.94**
Iraqi	4.77	4.72	**4.63**
Pashto	**4.91**	4.96	
German	4.72	4.30	**4.15**
Tamil	5.10	5.04	**4.90**
Thai	**4.82**	4.98	
Ojibwe	6.72	6.71	
Russian	5.13	**4.78**	
Konkani	5.99	**5.87**	
Inupiaq	4.79	**4.68**	

languages.

The advantage with the raw grapheme method is that it can be used when we have absolutely no information about the phonetics of a language - all we need is speech data and corresponding transcripts in the orthography of the language. The disadvantage with this method is that the models cannot use phonetic feature information while clustering similar phones together, since we have no information about what the phones actually are. Another disadvantage is that multiple characters that may actually map to a single phoneme in the language (like vowel markers and vowels) now map to different phonemes, which may lead to less data and context for each phoneme.

We see that across all five languages for which we have knowledge-based front ends, the voice with knowledge is much better than the UniTran based voice. The difference is much larger (0.32) on English, and relatively smaller (0.09) on Iraqi. This may be because English has many ambiguities that simple LTS rules cannot capture, and the amount of linguistic knowledge that went into the English voice was much larger than that in the Iraqi voice.

2.5 CHAPTER SUMMARY

In this chapter, we situated our work in the general Speech Synthesis pipeline and described the resources needed to build a TTS system in a new language. We also introduced our baseline framework for building grapheme-based voices for any language, which includes the raw grapheme and UniTran based techniques. We built voices using these techniques and compared them to voices built with manually created resources, using an objective metric of TTS system quality.

Next, we will look at how to improve grapheme-based voices for low-resource languages by using better modeling techniques and features.

3 IMPROVING GRAPHEME-BASED SYSTEMS

In Chapter 2, we introduced a framework to build grapheme-based voices for low resource languages either by treating each grapheme as a phoneme or using UniTran, a resource that provides a mapping from Unicode characters to phonemes in the X-SAMPA phoneset. The main difference between these two techniques is that in the first method (raw graphemes), we do not have any information about what the phonemes mean, in terms of phonetic features. In the UniTran technique, we map Unicode characters to a known set of phonemes. Although the UniTran mapping does not cover all the scripts in the world, it covers a majority of them, and we use it as our grapheme baseline.

In this chapter, we explore two techniques to improve upon the UniTran baseline. First, we explore how using a better modeling technique may improve voices in general, including grapheme-based voices. We also present subjective and objective evaluation results comparing grapheme and knowledge based voices, with the final goal of finding out whether our best grapheme-based voices are usable.

Grapheme-based systems are limited when it comes to ambiguities in Letter to Sound rules, due to their reliance on a single phonetic representation for each letter. In the second part of this chapter, we use acoustics and cross-lingual techniques to automatically derive letter-to-sound rules to improve synthesis for Hindi.

3.1 USING BETTER MODELING TECHNIQUES

In our standard speech synthesizers, we use individual Classification and Regression Trees (CART) to model the spectrum, pitch and duration. These CART models take features extracted from the data, at various granularities such as phonemes, syllables, words, phrases and surrounding context to predict the spectrum, duration and fundamental frequency (f0) of the synthesized speech. Some of the features include Part of Speech of the word, stress, position of the word in the phrase, pause information etc. Typically, rich feature sets and a large amount of training data lead to better models.

In low-resource scenarios, we typically do not have much data to train these trees on, and we may not be able to extract certain features such as Part of Speech. Using Random Forests [38], which is an ensemble learning technique that uses multiple CART trees may provide advantages in such cases by splitting the data in different ways and making better use of features. There are multiple configurations possible with Random Forests - either we can randomly vary the features used in each tree and build multiple trees, or we can randomly choose the parameters to predict. Another variation is by using random parts of the data to train models on. In the experiments described ahead, we predict a random set of features using multiple trees. The Random Forests build is now part of the standard Festvox distribution for building models of the spectrum and duration [39] and has been shown to improve knowledge based voices built with multiple databases of various sizes in different languages.

Our motivation for exploring better modeling techniques was to see how far we could get with an imperfect g2p mapping by using Machine Learning techniques that make soft decisions on choosing units based on context, rather than modifying the pronunciation of voices by replacing phonemes, which is a harder decision. This would be especially useful if we have a lot of data to train models on, or have good models to make use of small amounts of data.

3.1.1 *Synthesis with Random Forests*

For our Random Forests (RF) voices, we started with the UniTran mapping for the grapheme to phoneme conversion. For these voices, we build 20 decision trees to predict spectral features, where each tree is randomly restricted to 70% of the standard prediction features.

Each tree individually will typically give worse results than a tree built with all features, but the combination of multiple trees built with different features will typically give a substantial improvement. Although this technique is not specifically designed for grapheme based voice builds, the combination of predictions from different trees allows better use of features and avoids over splitting the data, which we felt may be helpful in this low resource scenario.

We used the same databases as described in Chapter 2 to conduct our experiments.

3.1.2 *Objective Evaluation*

We compare Mel Cepstral Distortion (MCD) for the UniTran and RF voices shown in Table 2. We see that in every single case, we get a significant improvement in MCD. In case of Hindi, Iraqi, German and Tamil we are able to perform better than the knowledge-based front end without Random Forests. We expect that building a Random Forest voice using the knowledge based front end would lead to even better improvements, but for our low-resource scenario where such a front end will not be available, this result is very encouraging.

Table 3: *MCD for languages built with UniTran vs. Random Forests*

Language	UniTran	Random Forests
English	5.11	4.91
Dari	4.95	4.80
Hindi	5.05	4.88
Iraqi	4.72	4.56
Pashto	4.96	4.80
German	4.30	4.10
Tamil	5.04	4.85
Thai	4.98	4.74
Ojibwe	6.71	6.19
Russian	4.78	4.64
Konkani	5.87	5.59
Inupiaq	4.68	4.56

3.1.3 *Subjective Evaluation*

So far, we saw improvements in objective metrics both while going from raw graphemes to UniTran and from single CART trees to Random Forests. We compared these conditions in subjective evaluations for English, German, Russian, Hindi and Tamil. Our choice of languages was based on the availability of subjects for listening tests.

We used Testvox [40] to carry out all our subjective tests both locally and on Amazon Mechanical Turk. In order to ensure that native speakers took the test, we translated the instructions to the respective languages. Each participant listened to audio clips in random order and was asked to pick the one they preferred, with the option of picking "no difference". Each participant listened to either 5 or 10 pairs of clips, with most participants listening to 10 pairs.

Table 4 shows the results of the subjective listening comparison between raw graphemes and the UniTran-based systems. We can see that in all five languages, subjects preferred the UniTran voices to the

Raw Graphemes voices. In some languages, this difference was higher while in languages like Hindi and Tamil where the UniTran method does not provide a substantial gain over raw graphemes due to the nature of the writing system, the difference wasn't as high. The quality of the Russian speech data being lower probably made it harder to differentiate between the two voices.

Table 4: *Raw graphemes vs. UniTran preference*

Language	Participants	Prefer RG	Prefer UniTran	No difference
English	13	22%	61%	17%
German	12	29%	54%	17%
Russian	11	32%	43%	25%
Hindi	12	38%	51%	11%
Tamil	12	28%	58%	14%

Table 5 shows the comparison between the UniTran-based systems with and without and Random Forests. There was a preference for the RF voices over the UniTran voices in all cases. Once again, the Russian speech seemed harder to differentiate.

Table 5: *UniTran vs. Random Forests preference*

Language	Participants	Prefer UniTran	Prefer RF	No difference
English	12	31%	61%	8%
German	9	36%	51%	13%
Russian	12	29%	47%	24%
Hindi	12	31%	51%	18%
Tamil	12	37%	53%	10%

Since our main goal was to see how far we could get with our best grapheme voices when compared to voices built with knowledge, we carried out transcription tests in English, German and Hindi. We asked 10 subjects to transcribe 10 sentences each in English, German and Hindi for the Knowledge Based and Random Forests grapheme conditions. Table 6 shows the percentage of words transcribed correctly for all the languages.

Here we see that for English, the UniTran+RF voice is still significantly worse than the knowledge based one. This may be due to two reasons - the knowledge that went into the English voice is substantially more than the other languages and the nature of the English

Table 6: *Words transcribed correctly for Knowledge Based and RF grapheme systems*

Language	Knowledge Based	Random Forests
English	87.14%	66.52%
German	90.85%	89.89%
Hindi	88.19%	86.34%

script makes a grapheme-based technique quite inappropriate for it. However, we see that for both German and Hindi, the difference between the usability of the RF and knowledge based voices is very low, and all the voices are almost at 90% transcription accuracy.

Although our best Hindi voice is reasonably understandable, there are some Letter to Sound rules that the grapheme-based system does not capture, such as schwa deletion, which may not affect intelligibility as much as it affects naturalness and the ability to sound like a native speaker. In the next section, we explore how we can automatically derive such rules to improve grapheme-based voices.

3.2 USING ACOUSTICS TO IMPROVE GRAPHEME-BASED VOICES

3.2.1 *Introduction*

Many languages in the world have a fairly unambiguous relationship between the letters and sounds. In such languages, it may not be necessary to specify a lexicon containing all the words in a language. It may be enough to specify a set of rules that maps letters to sounds, and have a few additional rules for special cases.

Many major Indian languages have a fairly good relationship between letters and sounds, with a few exceptions. Indo-Aryan languages such as Hindi, Bengali, Gujarati etc. exhibit a phenomenon known as *schwa deletion*, in which a final or medial schwa is deleted from a word in certain cases. For example, in Hindi, the final schwa (realized as the sound ə) in the word कमल pronounced 'kamal' is deleted. None of the consonants (क, म and ल) have attached vowels, and hence have inherent schwas, and the schwa on the last consonant gets deleted. The word लगभग pronounced 'lagbhag' has consonants ल ग भ ग, from which medial as well as final schwas attached to the consonant ग (g) are deleted. If schwa deletion does not take place, these words would erroneously be pronounced as 'kamala' and 'lagabhaga' respectively. In both these cases, the orthography does not indicate where the schwas should be deleted. Similarly, there are voicing ambiguities in languages such as Tamil, which may follow a complicated set of rules that may be language specific.

3.2.2 *Schwa deletion in Indo-Aryan languages*

There are well defined linguistic rules to predict when a schwa is deleted and when it is not deleted. However, there are exceptions to these rules that are reported as being around 11% of the vocabularly [41]. With the addition of new words and foreign words, one would expect this number to be higher. Also, Hindi and other Indian languages being low resource, there are no large lexicons available that can be used to automatically train these rules from.

Previous work on schwa deletion includes approaches that take into account morphology [41] to preserve schwas that may otherwise be deleted. Other approaches have used syllable structure and stress assignment to assign schwa deletion rules [42]. Choudhury et al. [43] use ease of articulation, acoustic distinctiveness and ease of learning to build a constrained optimization framework for schwa deletion.

As per our knowledge, there has not been much work done in the area of automatically deriving schwa deletion rules by making use of acoustics.

Schwa deletion for Hindi occurs in both word final and medial positions, while for languages like Bengali it occurs only in word final positions. The schwa deletion rules currently implemented in Festvox based on a simpler version of [41] are as follows:

- Delete the schwa at the end of a word
- Process input from right to left
- If a schwa is found in a VC_CV context, delete it

Taking the examples of कमल ('kamal') and लगभग ('lagbhag') mentioned earlier, we now see how these rules apply. In case of कमल, the three consonants क, म and ल all have inherent schwas, and the last schwa is deleted according to the first rule. Since none of the other schwas are in a VC_CV context, they remain. In case of लगभग, the consonants ल ग भ ग also have inherent schwas, and once again the final schwa gets deleted. However, the schwa attached to the second ग ('g') is in a VC_CV context, and hence gets deleted. This rule gives us the correct pronunciation in both these cases.

The rules stated above have exceptions which are not implemented in the current version of Festvox. So, there is scope for improving the schwa deletion rules even with the hand-written rules that the current Hindi system has, by making use of other information to automatically come up with these rules.

Our approach to this problem involves using the acoustics, or the way the voice talent pronounced words in order to learn more about the pronunciation of words, and try to generalize them into rules that can be used for new contexts.

First, we wanted to see the impact schwa deletion has on the overall quality of the database. Our baseline system for Hindi used the UniTran front end, which automatically assigned schwas to all consonants that did not have a vowel following them. We also built a system using the Indic front end, which had hand-written rules for schwa deletion.

We found that the difference between the MCD of the two voices was 0.05, which is not very significant - a difference of 0.08 is generally considered to be perceptually significant. In informal listening tests, we found that the difference between these voices was easily perceivable by native speakers. This indicates that MCD averaged across the entire database may not be the most appropriate metric to capture this phenomenon, considering that it occurs in around 40% of the words in the Blizzard Hindi corpus.

3.2.3 *Data and Resources*

Our approach to automatically discovering the rules for schwa deletion was to use cross-lingual models and acoustic information.

We set up the problem as follows: we considered Hindi to be our low resource language with no data or resources available other than the TTS audio, corresponding transcripts in Unicode and the UniTran mappings for the Hindi Unicode characters. The UniTran mappings assigned a schwa to every consonant character which meant that our task was to delete schwas wherever appropriate. Our Hindi data came from the 2014 Blizzard Challenge [44] and consisted of around an hour of speech from a professional male speaker.

Since we treated Hindi as the language with low resources, we built an acoustic model with data from other Indic languages. We created a corpus of an hour each of Bengali, Telugu, Tamil, Telugu and Rajasthani TTS data from the Blizzard Challenge and treated all these languages as higher resource languages. We used Sphinx [45] for building an Indic Acoustic Model with this data.

In order to learn the rules from acoustics cross-lingually, we used Assamese as our higher resource language. The Festvox Indic front end uses word-final schwa deletion rules for Assamese. Our Assamese TTS data came from the Blizzard Challenge data and consisted of an hour of speech from a professional male speaker.

It may seem counter intuitive to treat Hindi as a lower resource language than Assamese, but we set up the problem in this way so that we could conduct subjective evaluations easily on the Hindi voice.

3.2.4 *Experiments*

We used the Festvox Indic front end to correctly delete schwas for Assamese, which has word-final schwa deletion, and treated this as the correct phonetic expansion. We also used the UniTran baseline on Assamese which had spurious schwas, and used this as the wrong phonetic expansion in the places where the schwa should have been deleted. We force aligned the speech data using both these phonetic transcripts using the Indic Acoustic Model we built earlier.

We used the Indic acoustic models to force align the Assamese data using Sphinx. Since the phone set for all the languages was the same, we created a lexicon using the pronunciations of the Assamese words in the data. During force alignment, the Acoustic Model produces the timestamps and an acoustic spectral match score for each phoneme. Force alignment has been used to score pronunciations in pronunciation training systems, which we will discuss in detail in Chapter 5.

Next, we used Assamese data to build two classes for the CART tree: a positive class with the correct schwa labels from the knowledge based Indic front end, and a negative class with spurious schwas from the UniTran baseline. We used the score that Sphinx assigns for each phoneme during forced alignment, that we will refer to as the Acoustic Score and its right and left contexts, and the duration of the phoneme and context phonemes as features in our model.

The main idea was to look at these features in the synthesized speech from the UniTran baseline for Hindi after force aligning it with the Indic acoustic model, and predict if a schwa belonged to a negative or positive class, that is, whether it should be deleted or not. It should be noted that Assamese only contains word-final schwa deletion, but we hoped to capture word-medial schwa deletion in Hindi as well with this model.

After running the model, we got predictions about whether schwas should be deleted or not, in the Hindi data. We manually labeled 400 words to calculate the precision of these predictions and found that the precision was slightly better than chance. However, we found that many words were labeled with the correct schwa deletion rules more often than they were labeled wrong. So, we took the most frequent label of a word and created a new lexicon for Hindi with it. We used this lexicon while building the Hindi voice. Some examples of correctly identified lexical entries with word final and medial schwa deletion include उनकी('unki' instead of 'unaki'), राजनीति('rajneeti' instead of 'rajaneeti') and इस ('is' instead of 'isa').

3.2.5 *Evaluation*

Since we had seen that MCD was not sensitive to schwa deletion, we carried out a subjective evaluation comparing the voices we built for Hindi.

We synthesized 10 sentences for Hindi and asked 10 native speakers of Hindi to choose between the systems built with the UniTran baseline and our predicted schwa deletion lexicon. We asked them to pick the system they felt had better pronunciation, with the option to pick "no difference". Table 7 shows the results of the subjective evaluation.

Table 7: *Subjective evaluation of schwa deletion*

Prefer Baseline	Prefer Predicted	No difference
30%	59%	11%

We can see that there was a preference for the system with the predicted schwa deletion rules, when compared to the baseline. Here, we used the UniTran mappings as the baseline, which does not delete the schwa at all, but we could also have used a random baseline, which randomly deletes schwas.

In this experiment, we only used an Acoustic Model phoneme-level score and duration as features in our CART trees. We could use other features, such as articulatory features and Inferred Phonemes [46] that are also acoustically derived. Further improvements can be made to this model by using a larger database size for training, better Acoustic Models for force alignment and a richer set of linguistic features during prediction. In addition, such techniques can be applied iteratively as explained in the following chapter to improve systems.

3.3 CHAPTER SUMMARY

In this chapter, we explored techniques to see how far we could improve grapheme-based voices that used Letter to Sound rules or mappings, in low-resource scenarios.

We showed that using Random Forests as a modeling technique over single Classification and Regression Trees improves the UniTran based voices and in most cases, is better than the knowledge based voice modeled with single trees. Using better modeling techniques like Random Forests may help in low resource scenarios to make the best use of the available data. This shows how far we can go by making soft decisions about the pronunciation of such systems, without making changes to the phonemes in the LTS rules that we already have, which may be useful in cases where we have a large TTS database without any other resources in the target language and very few resources in related languages. We saw that our grapheme-based systems built with Random Forests were fairly intelligible in transcription tasks, with the exception of English, which is very difficult to model using grapheme-based techniques.

To improve naturalness, we may want to go beyond soft decisions, and start making changes to the Letter to Sound rules automatically. In this direction, we described a technique with which we could automatically come up with schwa deletion rules for Hindi using acoustics and cross-lingual data. The voice built with this technique had better pronunciation than the UniTran baseline, without using additional resources from Hindi. Although features derived from force-alignment have been used in the past to detect mispronunciation, this part of the work uses them in a novel, cross-lingual way to detect specific linguistic phenomena that are common between the two languages, but not identical. Other letter to sound rules can potentially be discovered by using cross-lingual models of related languages by using the technique described for discovering schwa deletion rules.

Many low-to-medium resource languages may have lexicons, but the lexicons may not be appropriate for the particular dialect of the language or may have ambiguities. In the next chapter, we describe techniques to disambiguate pronunciations in lexicons with homographs.

4 DISAMBIGUATING HOMOGRAPHS IN LEXICONS

4.1 INTRODUCTION

Text to Speech systems typically either make use of a lexicon to look up the pronunciation of words or use letter-to-sound rules trained on a lexicon or written by hand. So far, we looked at TTS systems that used a g2p (grapheme to phoneme) lookup table or a set of hand written rules.

Pronunciation lexicons that map words to strings of phoneme take a lot of human effort to create, and are available for only a few languages of the world. For some languages, these lexicons may be ambiguous or have incorrect entries and may not be enough to provide a completely accurate pronunciation of a word. This can be problematic for both speech recognition and synthesis.

In consonantal writing systems like Arabic and Hebrew, the diacritics that indicate short vowels are usually omitted. This creates ambiguity when it comes to pronunciation, which native speakers resolve by looking at the context of the word. This also creates semantic ambiguity if the word is looked at in isolation.

For example, words derived from the root "?lm" in Arabic can have different pronunciations 'alima', 'allama',' ilm', 'alam', that mean 'to know', 'to teach', 'science' and 'flag' [47]. Without knowing the diacritics, it is not possible to disambiguate the pronunciation of this word in isolation. Homographs in English like *lives* (verb, 'the cat lives') and *lives* (noun, 'nine lives') share the same orthography but have different pronunciations that can be disambiguated by Part of Speech (POS).

This can be a challenge for TTS systems particularly in the case of low resource languages for which tools like POS taggers may not be available or be very accurate. Also, some of these variants may not be predictable by context or POS but may need a deeper understanding of the semantics of the utterance.

We describe a technique to use Acoustic Models to disambiguate pronunciations by making use of the TTS database speaker's pronunciations of words. We build Text to Speech systems using the pronunciations that the Acoustic Model selects iteratively until our objective measure of TTS system quality converges. At synthesis time, we predict the pronunciation of the word by looking at linguistic features and context using only text.

In this setup, we assume that we have a lexicon for the language and our task is to pick between pronunciation variants in the lexicon. We describe a technique to disambiguate pronunciations in lexicons of dialects of Arabic, which typically have multiple lexical entries for most words due to the absence of diacritics.

4.2 RELATED WORK

The problem of having to choose from multiple variants of a homograph in certain languages also appears to some degree in all languages. Earlier work has considered how speaker (and style) lexical choices affect synthesis using both acoustic models to label them[48] [49] and statistical models at synthesis time to choose the right variant. It is targeted at very localized choices, such as vowel reduction.

This work has some similarity to the early work of Yarowsky [50] on homograph disambiguation. However here we do not require any human labeling of initial examples, but rely on the acoustic models to find these variants in the data. But then, like Yarowsky, we predict which distinct homographic instance to use as both training time (to improve our models) and at test time when doing novel synthesis.

Davel et. al [51] describe techniques to bootstrap pronunciation dictionaries for low-resource South African languages by using a human in the loop. They describe algorithms to incorporate phonemic variants in the lexicon and automatically extract rules for grapheme-to-phoneme conversion. However, they only make use of a previously available or bootstrapped lexicon to generate and predict these variants, in isolation.

SALAAM [52] is a technique in which an existing high quality ASR system is used to automatically generate pronunciations in a target language through cross-lingual phonetic decoding. These pronunciations are then used as the lexicon by the ASR system to decode speech in the target language. The main application of this method is to build an ASR system for the low resource target language, and the pronunciations are created to maximize ASR discrimination between them [53]. These pronunciations, however, may not be suitable for a TTS system to synthesize from. Also, the SALAAM method has been used for low vocabulary scenarios and needs multiple instances of training data for each word which are not necessarily available in a

TTS database.

Anumanchipalli et al. [54] improve the pronunciation of Out of Vocabulary (OOV) words for ASR by creating a list of candidate pronunciations of the word using its orthography and then using a single acoustic realization to rerank the n-best list of pronunciations using features such as the acoustic score, phone transition cost etc.

Prahallad et al. [55] use Hidden Markov Model (HMM) topologies that have insertions, deletions and substitutions for sub-phonetic states to capture pronunciation variations for conversational TTS and show differences in pronunciation between first and second mentions of content words.

Hain [56] argued that adding multiple pronunciation variants in the ASR lexicon was not necessary for modeling pronunciation variability. He described a method to utilize single pronunciation variants for words in the lexicon and implicitly model pronunciation variability which resulted in improvements in Word Error Rate for the Wall Street Journal and Switchboard data. It is worth investigating if this finding extends to other languages that do not have good lexicons or letter-to-sound rules to begin with.

Although a low-resource language may not have its own ASR system, in this work, we make an assumption that an Acoustic Model exists in the language or in a language very close to it, or we assume that the language has enough speech data and corresponding transcripts that we may be able to build our own Acoustic Model using them. We do not however assume that additional tools like POS taggers and diacritizers exist in the language.

4.2.1 *Data and Resources*

In our experiments, we applied our techniques to three databases in two languages - Modern Standard Arabic and Iraqi Arabic. Though Modern Standard Arabic can be thought of as a high-resource language, we treated it as a low-resource language to see how well our techniques perform against the gold standard.

4.2.2 *TTS databases*

For Modern Standard Arabic (MSA), we used data from the SASSC [57] corpus. The SASSC database contains single male speaker data spoken with different styles such as normal, funny, sad, questions etc. We used 50 minutes of data from the 'normal' speech part of the database as our TTS data. The corresponding transcript was fully diacritized. Since most written Arabic is not diacritized, we ran a script to remove the diacritics from the transcript for the rest of the experiments.

We used data from BBN created for the BOLT project and Iraqi Arabic TTS data from the TRANSTAC project [58] for the Iraqi Arabic TTS systems. The BBN data had 62 minutes of speech from a male speaker. The Transtac data had 74 minutes of speech from a male speaker. In both cases, the corresponding transcripts did not have any diacritics.

4.2.3 *Acoustic Models*

For building the MSA Acoustic Model, we used the rest of the *normal* speech in the SASSC corpus, leaving out the utterances labeled *traditional* which were in Classical Arabic. This came to around 5 hours of speech data. We removed the diacritics from the transcripts used for training the Acoustic Model.

For building an Iraqi Arabic Acoustic Model, we used 2-way dialogs between native Iraqi Arabic speakers, interpreters and native English speakers from the Transtac project. We extracted 20 hours of Iraqi Arabic utterances spoken by native speakers from the dialogs using manually annotated timestamps and transcripts. The transcripts did not contain any diacritics.

We used the CMU Sphinx speech recognition toolkit [45] to train Acoustic Models and force-align speech, as described in Chapter 3.

4.2.4 *Lexicons*

For MSA, we did not have a standard lexicon. However, we had the transcripts and labels for the audio data from the SASSC corpus. We aligned the phoneme labels with the words in the transcript and created a lexicon specific to our corpus. However, this alignment was not completely accurate. In Arabic, the determiner 'al' is often blended with the end of the previous word to create fluid speech. This makes determining the word boundaries difficult because the final vowel is combined with the initial 'a' of the 'al', and the determiner sounds like it is attached to the previous word instead of the correct word. This creates many words with extra phonemes at the end, and definite words that do not have the phonemes for the determiner. This resulted in lexicon entries that are not entirely correct.

We used an Iraqi Arabic lexicon from LDC which contained words with and without diacritics in Iraqi script, Buckwalter transliteration [59] for each word, syllable boundaries and POS tags. We created a phone set by mapping the Buckwalter characters to individual phonemes with the appropriate phonetic features. The LDC lexicon contains around 88k words, out of which 11k words have multiple pronunciations, with some words having as many as eight different pronunciation variants. We created a new lexicon using Iraqi Arabic surface forms without diacritics, phonemes with syllable boundaries and POS tags. We numbered the pronunciation variants in the lexicon as word, word(2), word(3) etc.

4.2.5 *Other resources*

In order to better process the MSA data, we removed all diacritics and normalized certain consonants that have many common variations. To get part of speech tags for MSA, we used the Stanford Tagger [60] with their standard Arabic model. For Iraqi we used CALIMA [61], a morphological analysis tool for Arabic dialects developed at Columbia University.

4.2.6 *Learning Pronunciations from Acoustics*

Our data consisted of transcripts without diacritics and corresponding speech for building TTS and Acoustic Models. We also had a lexicon for Iraqi Arabic and a derived lexicon for MSA. However, each of these lexicons had multiple lexical entries for the same word, since all the diacritics had been removed and words with different short vowels were now treated as being variants of the same base word.

Given a pronunciation dictionary with multiple pronunciations for ambiguous words, audio recordings for TTS and the corresponding transcript, our task was to choose the correct pronunciation from the lexicon for the words in the transcript.

Figure 3 illustrates our approach to learning pronunciations from audio. First, we use an Acoustic Model from our target language to force align the original TTS transcript with the TTS audio. The Acoustic Model is trained on a large amount of data in the language, if available. During the process of forced alignment, the model chooses a pronunciation from the lexicon for a particular word based on the best scoring phoneme string in the lattice. So at the end of force-alignment we get a transcript that has different pronunciation variants compared to the original transcript.

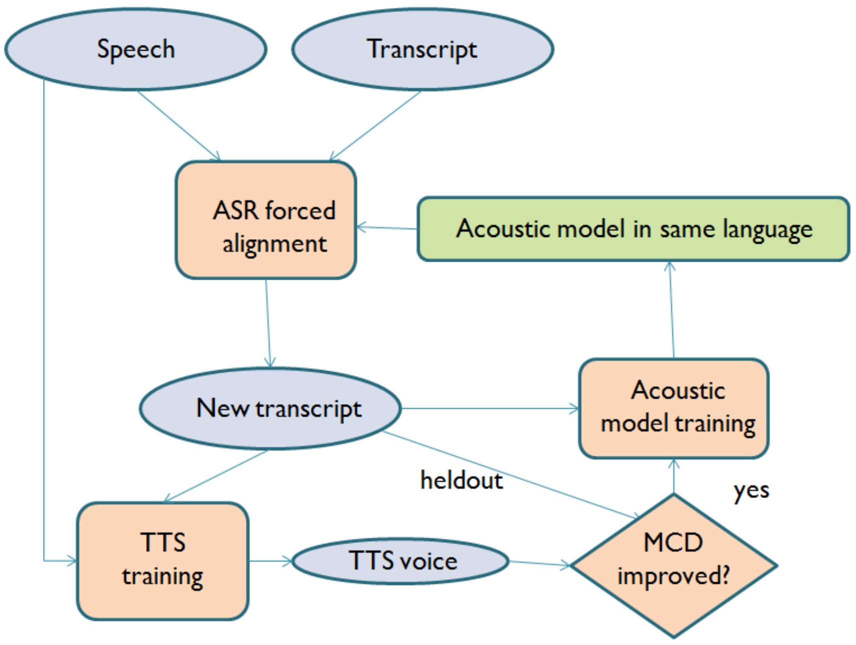

Figure 3: Iterative process to choose better pronunciations

Next, we use the new transcript and the speech from our TTS database to rebuild an Acoustic Model. This targeted Acoustic Model is then used to force align the transcript used for training the data. We repeat this process iteratively and at each stage build a CLUSTERGEN voice. We use the same held out set of sentences for all the iterations to test the TTS system and measure the MCD, thus measuring the improvement in labeling with each iteration. We stop the iterations when the MCD no longer improves. At each stage, an alternative to building new targeted acoustic models is to do some kind of model adaptation, particularly in cases when the TTS database size is very small.

4.3 OBJECTIVE RESULTS

Table 8 shows the MCD of the baseline system and the best iteration. An improvement of 0.08 in MCD is considered to be perceptually significant and an improvement of 0.12 is equivalent to doubling the training data [17]. In all three cases, we get a significant improvement in MCD compared to the baseline.

In previous work, we have seen that repeating this process iteratively while building cross lingual phonetic TTS systems for languages without a written form has given us gains, typically in the third or fourth iterations [62]. In these experiments, we observed a sharp decrease in MCD in Iteration 1 for all three TTS databases. In case of the Iraqi BBN database, there was a slight improvement in the MCD in Iteration 2.

Table 8: *Baseline and best iteration MCD scores*

Database	Baseline	Best Iteration
Iraqi BBN	4.67	4.21
Iraqi Transtac	4.88	4.35
MSA	6.64	6.34

The MSA database originally consisted of diacritized text, so we built a voice using this text which we treated as the gold standard. The MCD of the gold standard MSA voice was 6.44, which is higher than the best iteration MCD. It is important to note however that the lexicon created for MSA was not completely accurate even for the gold standard voice due to the issues in processing it mentioned earlier. However, this result shows that the iterative technique probably managed to recover from some of those errors. Since diacritized text was not available for the Iraqi databases, building a gold standard voice for them was not possible.

4.4 PREDICTING PRONUNCIATIONS FROM TEXT

At synthesis time, we need to be able to predict pronunciations from text. The Iraqi Arabic LDC lexicon contained Part of Speech tags for all the words, which we used to build Classification and Regression Trees (CART) to predict the pronunciation of a particular word, given its POS and the POS of the previous two and next two words. We used the Edinburgh Speech Tools [63] CART tree building program to build and test our trees. We built individual CART trees for each word and tested our trees on held-out data.

A survey of the incorrectly predicted words showed that most of the failed disambiguations were due to homographs with different pronunciations where contextual information is needed in order to choose the correct one, and only using POS was not enough. No reliable dependency parsers exist for Iraqi Arabic, so we used lexical features from the surrounding words to help with disambiguation. We also used induced POS [64] for MSA, but since we had a small amount of training data, we did not get reliable tags. We use the Iraqi version of CALIMA for morphological analysis. The morphological analysis includes a stem for each word, and we use this to extract prefixes, a stem, and suffixes for each word. The lexical feature vector consists of the stem and affixes for the target word as well as the next and previous word.

We found that the accuracy of the CART trees by performing 10-fold cross validation for Iraqi Arabic was very high at 93%, while for MSA it was much lower at 76%, which can be explained by the problems with the accuracy of the lexicon mentioned earlier and less training data for the trees. Using a larger amount of training data and better features trained in an unsupervised manner from the text may help in improving the prediction models.

4.4.1 *Subjective evaluation*

We saw that our iterative method resulted in better labeling for the three databases and hence better MCD, which resulted in much better quality that should be perceptually significant. However, we wanted to test how good our predictions were and whether subtle variations in pronunciation could be perceived by native listeners.

We used the Testvox [40] tool for creating AB preference tests with a 'no difference' option for all our subjective tests. In our first set of listening tests, we conducted preference tests with four native Arabic speakers outside our research group for Iraqi Arabic. Subjects were asked to listen to two synthesized sentences from the test set that had one word that our model predicted a different pronunciation for than the default pronunciation in the lexicon. We found that there was a slight preference for the utterances with our predictions compared to the baseline. In many cases, the difference in pronunciation was subtle and was not perceived in the listening test.

We thought it would be useful to have the subjects explicitly focus on the word that was different in the two utterances so that we could judge whether the pronunciations we predicted were different or not. We conducted subjective tests for MSA, and synthesized sentences similarly as we did for Iraqi. In the listening tests, we showed subjects the transcript with the ambiguous word highlighted and asked them to pick the synthesized utterance in which the word sounded correct, or choose a third option if they could not tell a difference. Table 9 shows the results of the listening test with 9 native Arabic subjects, with each subject listening to 10 pairs of sentences.

Our results for MSA show a significant preference for the predicted pronunciations compared to the baseline. The important point to note

Table 9: *Subjective evaluation of pronunciation choice prediction for MSA*

Prefer Baseline	Prefer Predicted	No difference
4.44%	64.44%	31.11%

here is that the rest of the sentence except the highlighted word was identical, and all sentences were synthesized with the same system using our best training labels. This result is very encouraging as it shows that the difference in pronunciation can be perceived and that we are making the right predictions.

An obvious question at this stage is whether it is worthwhile trying to improve the pronunciation of systems when listeners need to be told to focus on a word to discern differences in pronunciation. Our intuition is that pronunciation variations will be easier to perceive and pronunciations will be expected to be more accurate as systems improve.

4.5 CHAPTER SUMMARY

In this chapter we described our technique to disambiguate pronunciations in lexicons containing homographs and applied it to the problem of picking pronunciation variants from lexicons of dialects of Arabic. We carried out experiments on two databases of Iraqi Arabic and one database of Modern Standard Arabic and evaluated the systems built using objective and subjective metrics of TTS system quality.

Our techniques showed a significant improvement in objective measures of TTS quality for all three databases for dialects of Arabic. We built a model to predict pronunciations at synthesis time from text using POS, lexical and context features, and found significant preference for pronunciations selected by the model in subjective evaluations for MSA.

Such techniques have been used in the past to disambiguate homographs in English but scaling them to languages like Arabic where homographs occur very frequently can lead to large improvements in TTS system quality. In addition, there is some evidence that performing multiple iterations and building targeted Acoustic Models may create some improvements.

The problem of missing diacritics in languages written in the Arabic script extends to languages like Urdu, Farsi, Hebrew etc. One can also imagine applying the same techniques to European languages with missing accents in the transcript, or languages that contain many homographs. We can also apply such techniques to pick between multiple pronunciation variants in noisy or automatically created lexicons.

5 IMPROVING PRONUNCIATION USING ACOUSTIC FEATURES

5.1 INTRODUCTION

In earlier chapters, we introduced a baseline system for grapheme to phoneme conversion in any language, and showed how we could improve pronunciations with the help of acoustics and cross-lingual techniques for specific ambiguities that existed due to the written forms for some languages. In all these cases, we were already aware of the pronunciation nuances in the languages ie. the ambiguity in short vowels in Arabic because of the written form, and the schwa deletion issue in Hindi.

However, for very low resource languages, we may not know anything about the language and may not have access to a native speaker with linguistic knowledge. We may not be able to search for specific pronunciation issues and solve them. In such cases, it would be useful to have a general technique that could improve the pronunciation of low-resource grapheme-based systems. Further, it would be useful to have a general technique to improve pronunciation that can be used for higher resource languages as well.

For Hindi schwa deletion and Arabic homograph disambiguation, we used an approximate or incorrect mapping from graphemes to phonemes and then used acoustics and cross-lingual techniques to refine these mappings by substituting phonemes. In this chapter, we explore using features derived entirely from the acoustics, in a bottom-up manner.

Ideally, our features should be able to capture pronunciation variations that agree with linguistic theories. However, our aim is to improve the intelligibility and naturalness of TTS system, so we will continue to use measures of TTS system quality as our evaluation criteria.

Since our focus is on improving pronunciation of low resource languages, our technique should make use of minimal resources in the target language - ideally only the TTS database and cross-lingual resources from related higher resource languages. First, we identify some potentially useful features that satisfy these conditions.

5.2 ACOUSTICALLY-DERIVED FEATURES

There has been work on using acoustically-derived features for improving various Speech Processing and Language Technologies applications.

5.2.1 *Duration in TTS*

Kominek et al [65] suggest the use of phone durations to remove outliers to prune unit selection databases. Units with extreme durations can be thought of as bad units and can be removed from the database. In case of pronunciation, the mismatch between a predicted duration and actual duration as detected by a labeling algorithm can suggest that there is a case of phoneme insertion, deletion or substitution. Consistent patterns for particular phonemes across the database may suggest pronunciation errors that the system is making.

5.2.2 *Computer Aided Language Learning*

There has been considerable work in the field of Computer Aided Language Learning (CALL) and Computer Aided Pronunciation Training (CAPT) on trying to identify pronunciation errors made by second language (L2) learners and providing feedback on pronunciation. Explicit pronunciation training has been treated as an important issue because it has been shown that "pronunciation quality below a certain level of proficiency places additional stress on the listener and seriously degrades the ability of native speakers to understand what is being said" [66].

The Fluency project, one of the earliest examples of a CALL system that dealt with pronunciation training, initially contained a duration trainer module that scored the duration of phones of L2 learners based on native speech [67]. Later, the Fluency system also pinpointed specific phonetic errors made by non-native speakers of English, such as IH as in 'bit', S as in 'sit', T as in 'tie', V as in 'vine' [68]. In addition to pinpointing pronunciation errors by making use of the CMU Sphinx II Speech Recognition system, it also gave feedback to users on how to correct those errors.

More recently, [69], [70], [71] describe using pairwise confusion models to classify commonly made segmental errors. They use an ensemble of Support Vector Machines (SVM) to classify mispronounced phonemes and provide feedback about the general type of pronunciation errors that a student makes at the end of the session, instead of pointing out every single pronunciation error, which may be demotivating for the student. To provide this session-level feedback, only those mispronounced phonemes are considered that have a error rate significantly higher than that of native speech.

Several other studies on CAPT describe features such as duration, confidence, posterior probabilities, log-likelihood produced by HMMs [72] [73] combined with scores from Gaussian Mixture Models [74] to score segmental pronunciations. [75] takes a different approach by modeling sounds in the frequency space relative to the other sounds in the language. [76] use syllable durations and F0 contours to study differences in the generation of tones between native speakers of Mandarin and German learners of Mandarin.

Peabody [66] describes an anchoring method to project MFCC feature vectors into a feature space that improves mispronunciation detection for CAPT systems. A decision tree classifier is used with parallel native and non native models to score mispronunciations and crowd-sourcing is used for obtaining phonetic labels.

We can view the problem of improving pronunciation in synthetic speech as being similar in many ways to the problem of pronunciation errors made by second language learners. However, most of the work in CAPT has focused on scoring and giving feedback on specific pronunciation nuances, that may be specific to the L1 or L2 languages or the L1-L2 pair.

5.2.3 *Articulatory features*

Articulatory Features (AFs), which are features that describe various properties of phonemes such as place of articulation, vowel height, voicing etc., have been used in a wide variety of applications. Articulatory features can have discrete or continuous values, but in most cases they are represented by a vector of binary values that indicate the presence of these features. These features are typically labeled at a phoneme level, although in reality they are much more continuous in nature.

Deng and Sun [77] describe an early feature-based Automatic Speech Recognition (ASR) architecture that only makes use of articulatory features and is evaluated on a phone recognition task. AFs have also been used to improve ASR robustness in case of noise and reverberation, particularly in case of high noise conditions [78].

Metze [79] describes a multi-stream setup for recognition of conversational speech using Articulatory Features along with phone-based recognizers, in which stream feature weights are learned discriminatively. Articulatory features are more robust to speaking styles and conversational speech, and integrating these features with conventional recognizers improves performance.

Stuker et al. [80] [81] describe a technique to use multilingual and cross-lingual articulatory features in an HMM-based ASR framework. They show that by using multilingual AF detectors with learned weights, they can outperform monolingual detectors in terms of ASR Word Error Rate.

Schultz and Waibel [82] use AF classifiers to improve the performance of ASR for low resource conditions on the GlobalPhone corpus [83] by using a global unit set that can be shared across languages. They used a small amount of adaptation data in the target language for Large Vocabulary Continuous Speech Recognition in 10 languages.

Articulatory features have also been used for expressive speech synthesis[84]. In this work, Articulatory Features have been extracted for a variety of emotion and personality databases. Models are built to predict the Articulatory Features from text, and they are mapped to MCEPs which are then used for synthesis. The method of extracting AFs is language and dialect independent, and hence can be used cross lingually.

Articulatory features have been used recently in Text to Speech in unwritten languages with some success. Muthukumar et. al [46] have used Articulatory Features to come up with Inferred Phonemes (IPs) for languages without a standardized orthography by training a Neural Network on a corpus of labeled English speech. Phonemes are inferred by clustering them so that they best predict the MCEPs in synthesized speech. In this work, the authors experimented with the number of inferred phonemes to find the number that produces the smallest MCD and also came up with continuous phonetic features for them. Adding the automatically derived phonetic features to voices in place of traditional phonetic features got from a knowledge-based front end improved voices.

Recently, we used Articulatory Features and Inferred Phonemes for the task of minimal pair discrimination in the Zero Resource Speech Challenge in Interspeech 2015 [85]. We used both the raw Articulatory Features and the Articulatory Features of the inferred units as frame based representations of speech. The evaluation metric was minimal pair ABX discrimination within and across speakers and we found that the AFs performed well across-speakers, which may suggest that they capture speaker-independent information.

Our task in this thesis is quite similar to this in that we want to find a representative phoneme set for a language that has an underspecified written form in most cases. However, unlike this work, we do have a set of baseline phonemes that have a correspondence with the real hidden phonemes that we are trying to discover, which we should exploit. Since these AF-based phoneme like units have been used successfully in the low resource scenario for speech synthesis, we decided to explore using them to improve the pronunciation of our systems. We now describe some experiments with our grapheme based systems using the IP features.

5.2.4 *Inferred Phonemes*

First, we wanted to explore how much additional information these acoustically-derived Inferred Phonemes (IPs) gave voices that already had written forms. This is in contrast to [46], who did not use the written form of the text, but assumed that the languages did not have written forms and used cross-lingual phonetic transcripts. They used the IPs as a derived written form to synthesize speech from. They also assumed that there would be some other component that generates this written form, such as a Statistical Machine Translation system inside a Speech to Speech Translation system. More details about this framework can be found in Chapter 7.

Since we already had written forms for the grapheme voices (which may not be ideal but are probably superior to ASR-derived transcripts), we explored using the IPs as features within our system. The motivation behind this was that the IPs, being completely acoustically derived may be able to capture some pronunciation variation information which can then be used to model pronunciation implicitly within our Machine Learning models.

To extract IPs, first, the speech from the TTS database was labeled with Articulatory Features. The AF classifiers were trained on labeled English TIMIT data [86] using a 3 Hidden Layer Neural Network. This was the only language resource used to derive the IPs for all the different experiments that we conducted. The disadvantage with using only English to train AF classifiers is that articulatory features present in other languages may not be detected by such classifiers. Once the AFs were extracted for each utterance, they were clustered into IPs based on their AFs and the number of IPs specified by the stop value of the tree. Instead of using cross-lingual phonetic transcripts to fix the time stamps of the IPs as was done by [46], we used the phonemes from the TTS front ends for each language.

This gave us an IP for each phoneme in our TTS database, which we then used as features in our Classification and Regression Tree (CART) to predict the spectrum. We also used the IPs of the previous and next phones as features.

We conducted experiments on two databases - the Hindi Blizzard database and the CMU Arctic RMS US English database. For each of these, we used both our knowledge based front ends (the Indic front end and US English front end respectively) and the UniTran grapheme front ends. We also varied the number of IPs generated for each voice by changing the stop value of the Classification and Regression Tree. The exact number of IPs predicted for each database with each of these stop values varied, but in general IP1, IP2 and IP3 had 40, 60 and more than 100 IPs. We extracted IPs for the entire database, including the test set, which means that there was an IP feature value for each phoneme in the test sentences as well.

Our hypothesis was that using IPs as features would lead to some improvements in the voices. Table 10 lists the MCD of voices built with these configurations, for a total of 12 voices.

Table 10: *MCDs of voices using IPs as features*

Database	Front End	IP1	IP2	IP3
RMS English	Knowledge Based	ip1	ip2	ip3
	4.75	4.71	4.70	4.72
RMS English	UniTran	ip1	ip2	ip3
	5.13	4.97	4.97	5.00
Blizzard Hindi	Knowledge Based	ip1	ip2	ip3
	3.92	3.84	3.88	3.86
Blizzard Hindi	UniTran	ip1	ip2	ip3
	4.18	3.92	3.91	3.93

First, we see that for the RMS knowledge based voice, the decrease in MCD from the knowledge based voice to the IP voices was not significant (a decrease of 0.08 is considered perceptually significant, and 0.12 is equivalent to doubling the database). However, for the English UniTran voice, we see that there is a significant decrease in MCD for all the IP-based voices, with no significant differences between them. For the Hindi knowledge-based voice, we see a significant decrease in MCD in two of the three IP voices. For the Hindi UniTran voice, we see a very large decrease in MCD, suggesting that the IPs provide a lot of information in this case.

From these results it seems that the grapheme-based voices benefit more from the IPs than the knowledge based ones. Also, the lower resource language (Hindi) seems to benefit more than the higher resource language English.

It is important to note that in these experiments and the experiments described in [46], the IPs were extracted from the audio across the entire database, including the test set. This means that by predicting IPs perfectly for our test set, we could hope to achieve these results. However, IPs are extracted from acoustics, which are not available at synthesis time. So, the IP features needed to be predicted from text, using low level features like phonetic features, high level features of the word and phrase and context.

We built CART models to predict IPs from text using our standard text-based features. However, we found that in most cases, we did not get very high accuracy in prediction, which led to an increase in MCD making the gains provided by the IPs overall not significant. However, the results above suggested that the IPs were capturing some additional information from the acoustics that improved all the voices significantly except the very high resource US English knowledge-based voice, so it was worthwhile trying to analyze and predict them better.

Muthukumar et al. [46] used the IP transcript like a phonetic transcript, and varied the number of IPs from 30-200 to match the number of phonemes or allophones expected in a language. We decided to experiment with a very small number of IPs to be able to analyze and predict them better.

5.2.4.1 *Using fewer IPs*

We build voices for RMS and Hindi using the grapheme based front ends with 12 and 21 IPs, respectively. Surprisingly, we found that there was no significant difference between the MCD gains that we got (without IP prediction) for both these voices with a lower number of IPs. Table 11 shows the MCD of voices built using fewer IPs than before, with and without IP prediction from text, compared to the baseline with no IP features.

We see that even though we got significant gains even with using fewer IPs in the no prediction case, we were only able to predict the

Table 11: *Using fewer IPs as features*

Database	Front End	Baseline	IPs with no prediction	IPs with Prediction
RMS English	UniTran	5.13	5.04	5.10
Hindi	UniTran	4.18	3.96	4.07

Hindi IPs reliably enough to get a significant improvement in MCD. However, this result also suggests that the IPs were capturing some higher level information that could potentially be useful to predict.

We also built our Classification and Regression Trees with the stepwise option, that greedily searches for the best feature at each stage of the tree building process for IP prediction. While we did not get significant gains in prediction, we found that the top features used by the model were the name of the phoneme and left and right phonemes, the duration of the current phoneme, consonant type and voicing.

Improvements can potentially be made in predicting IPs by using better Machine Learning techniques, or an iterative framework similar to the ones we use for disambiguating homographs in Arabic.

5.2.4.2 *Visualizing IPs*

To visualize what kind of information IPs may be providing, Figure 4 shows the distribution of the 12 IP features for each phoneme in the RMS US English database using the UniTran front end, with a darker color indicating higher frequency. The y axis in the heat map shows the phonemes from the SAMPA phone set and the x-axis shows the IPs for each phoneme across the entire database.

We can see that certain phonemes, such as the short vowels 'A', 'e' and 'i' seem to share some high-frequency IPs which indicates that those particular IPs may be capturing vowel-specific information. In some cases, certain phonemes like 's' and 'r' have a single IP with high frequency while others such as 'd' seem to have multiple IPs. We should keep in mind that these phonemes have come from a grapheme-based front end, so these IPs may be able to point out where splits should have occurred due to context in the g2p mapping that did not occur due to a single g2p mapping from UniTran. Such visualizations may be helpful in eliminating IPs that are not discriminative or that are very low frequency for better prediction.

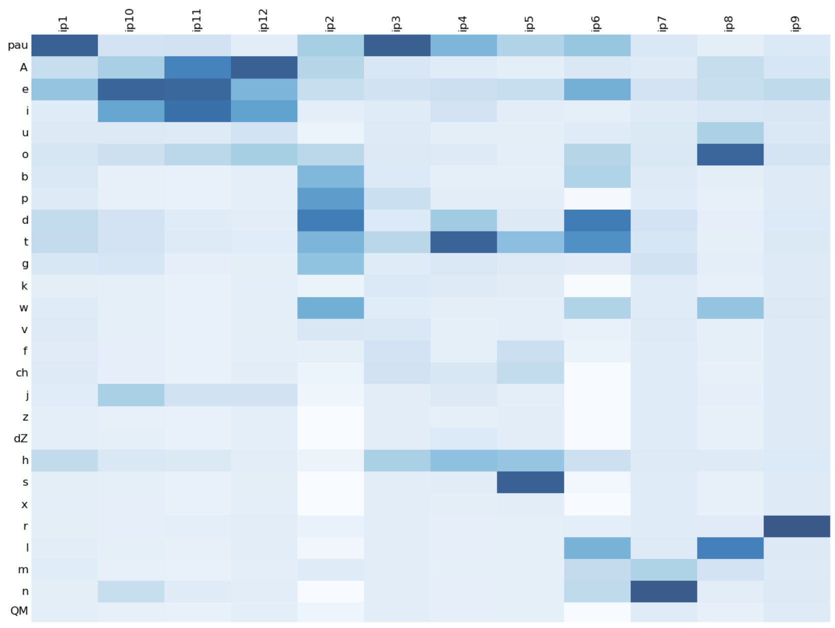

Figure 4: IP distribution for RMS US English voice

5.3 CHAPTER SUMMARY

In this chapter, we tried to improve the performance of low-resource grapheme-based TTS systems by using acoustically-derived features. We used Inferred Phonemes clustered using Articulatory Features extracted from TTS databases with the help of cross-lingual Articulatory Feature extractors. We presented a novel technique of improving systems in a generalized manner by adding these features into the standard set of features used for modeling the spectrum and showed results on two databases in knowledge-based and grapheme settings. By adding IPs as features, we got significant gains in MCD, particularly when compared to voices built with knowledge-based front ends and for higher resource languages. This suggests that acoustically derived features may be useful for improving low resource grapheme-based voices. Also, such features may be used for improving knowledge-based voices as well by using such features, particularly if the lexicon being used is not completely suitable for the voice, such as dialects of a language or speaker-specific pronunciation variations.

However, predicting IPs from text at synthesis time proved to be a challenge. Even with a low number of IPs that seemed to capture higher level phonetic information, we were unable to predict IPs from text reliably except in the case of the Hindi grapheme database. We tried to visualize what these IPs may be capturing by looking at the distribution of IPs and corresponding phonemes. Our bottom-up approach of using features derived from acoustics was in contrast to previous chapters, where we went from a text representation to phonemes.

One limitation of using a single language (in this case English) to train AF classifiers is that the classifiers may not be able to capture AFs that occur in certain target languages. Using multiple or related languages for training may improve such models. The difficulty of predicting IPs suggests that we need better techniques to jointly model acoustics and text in order to be able to take advantage of the information captured by such features. With better models, we may be able to exploit cross-lingual and acoustic information for improving the pronunciation of languages that we do not know anything about, automatically.

6 CODE MIXING

6.1 INTRODUCTION

In Chapter 3 and Chapter 4, we improved the pronunciation of TTS systems for low resource languages by improving the Letter to Sound rules, or disambiguating lexical entries. In both these cases, we assumed that the orthography of the language was standardized and that it was fairly suitable for the language at hand, even if it was borrowed from a related language.

However, in many cases, a language is written using the script of another language, even if it has its own standardized written form. A common phenomenon that occurs in text in Social Media, Instant Messaging and email in bilingual and multilingual societies is Code Switching, in which users type multiple languages in the same script, or sometimes retain the original script of the languages that are being mixed in the sentence. This occurs very frequently in speech as well.

Code Switching is defined as switching between different languages in speech or text, in which the grammatical forms of the languages are preserved. Typically, one can identify one, or a few points in a sentence where Code Switching occurs. Code mixing refers to the mixing of phrases, words and morphemes of one language into another language [87]. Lexical borrowing occurs when a word from one language is borrowed and used in another language while following the syntactic rules of the borrowing language.

Bali et al. [88] have given examples shown in Table 12 to illustrate the difference between Code Mixing and Code Switching. Code Switching can be thought of as more of an an inter-sentential phenomenon while Code Mixing can be thought of as an intra-sentential phenomenon, although the techniques we describe later in this chapter are agnostic to the difference between the two. The example in Table 12 does not show intra-word Code Mixing, which is also observed in some mixed languages.

In this thesis, we focus on pronunciation modeling, so we make a simplifying assumption to treat all these as the same, even though making these distinctions may affect pronunciation. Also from this point onwards, we will use the term Code Mixing for Code Mixing, Code Switching and borrowing to be consistent with recent relevant

Table 12: *Code Switching vs Code Mixing*

Code Switching
I was going for a movie yesterday. raaste men mujhe Sudha mil gayi.
Gloss: [I was going for a movie yesterday.] way in I Sudha meet went.
Translation: I was going for a movie yesterday; I met Sudha on the way.
Code Mixing
Main kal movie dekhne jaa rahi thi and raaste me I met Sudha.
Gloss: I yesterday [movie] to-see go Continuous-marker was [and] way in [I met] Sudha.
Translation: I was going for a movie yesterday and on the way I met Sudha.

literature. Our task is to be able to synthesize speech that sounds as natural and intelligible as possible, given such mixed text, with an emphasis on figuring out the best pronunciation rules to apply to the text.

From the point of view of language resources, Code Mixed languages can be considered to be low resource languages in general, because not many resources (particularly lexical) typically exist for such mixed languages. However, in many cases, one of the languages being mixed may be high resource, which can be exploited.

6.2 RELATION TO PRIOR WORK

Code Switching and Mixing have been identified as challenges for language technologies ranging from Information Retrieval to Automatic Speech Recognition.

The Code Switching shared task at EMNLP 2014 [89] consisted of data from 4 mixed languages (English-Spanish, Nepali-English, Arabic-Arabic dialect, Mandarin-English) and the task was to identify for each word which language it belonged to, or whether it was mixed, ambiguous or a named entity. Chittaranjan et al. [90] describe a CRF based approach for word level Language Identification for this task, in which they used various lexical and character-based features.

Recently, Code Mixing in text has been studied for languages of the Indian subcontinent, which exhibit a lot of mixing with English and other Indian languages owing to the large multilingual population and the number of languages.

Vyas et al. [91] created a manually annotated corpus of code-mixed social media posts in Hindi-English and used it for POS tagging. They analyzed this corpus and found that 40% of the Hindi words in the corpus were written in Romanized script. They also found that 17% of the data exhibited Code Mixing, Code Switching or both. They found that transliteration and normalization were the main challenges while dealing with such text.

Bali et al. [88] further analyzed this data to find that words fall into categories of code mixing, borrowing and ambiguous, with many borrowed words being written in English and many Hindi words being misidentified as English due to spelling. They suggest that a deeper analysis of morpho-syntactic rules and discourse as well as the sociolinguistic context is necessary to be able to process such text correctly.

Gupta et al. [92] introduce the problem of mixed-script Information Retrieval, in which queries written in mixed, native or (often) Roman script need to be matched with documents in the native script. They present an approach for modeling words across scripts using Deep Learning so that they can be compared in a low dimensional abstract space.

Another Code Switching corpus of interest is an Algerian Arabic-French corpus that contains 7000 comments from an Algerian news website [93]. The unique feature of this corpus is that it contains Algerian Arabic text written in Romanized form ('Arabizi'), and Romanized Algerian Arabic tends to use Romanizations based on French orthography. This corpus has been manually annotated at a word-level with 'Arabic', 'French' and 'Other'.

Code Switching has also been studied in the context of speech, particularly for Automatic Speech Recognition (ASR) and building multilingual TTS systems.

Modipa et al. [94] describe the implications of code-switching for ASR in Sepedi, a South African language and English, the dominant language of the region. They find that the frequency and continuum of code switching makes it a very challenging task for traditional ASR trained on only Sepedi.

Vu et al. [95] present the first ASR system for code-switched Mandarin-English speech. They use the SEAME corpus [96], which is a 64 hour conversational speech corpus of speakers from Singapore and Malaysia speaking Mandarin and English. They use Statistical Machine Translation based approaches to build code mixed Language Models, and integrate a Language ID system into the decoding process.

Ahmeda et al. [97] describe an approach to ASR for code switched English-Malay speech, in which they run parallel ASRs in both languages and then join and re-score lattices to recognize speech.

Bilingual TTS systems have been proposed by [98] for English-Mandarin code switched TTS. They use speech databases in both languages from the same speaker and build a single TTS system that shares phonetic space. Microsoft Mulan [99] is another bilingual system for English-Mandarin that uses different front ends to process text in different languages and then uses a single voice to synthesize it. Both these systems synthesize speech using native scripts, that is, each language is written using its own script.

A TTS system that is capable of reading out Social Media text or informal messages needs to be able to handle multiple languages. A Personal Digital Assistant also needs to be able to both recognize and produce natural Code Mixed speech while interacting with users who live in multilingual societies. To do this, the ASR, TTS and Machine Translation components of the system need to address challenges posed by code mixing. With this motivation, our task for this part of the work was to improve the pronunciation of a TTS system that has to read out Code Mixed text. As before, we assume that very few resources exist in the target language, though the other language that is mixed with the target language may be a higher resource language.

We divide the task of synthesizing such text into two main categories - synthesizing text which retains the script that the language is in, resulting in mixed scripts, and synthesizing text that (usually) uses the script of a single language, resulting in the use of a non-standard script for the other language.

6.3 CODE MIXING WITH MIXED SCRIPTS

In some text, people preserve the original scripts that the languages use while code mixing. In cases where the scripts that the languages being mixed are different, we may be able to identify the language by looking at the script, and then use the appropriate TTS voice or LTS rules to synthesize it.

Ideally, we should get recordings from the same bilingual or multilingual speaker for all the languages that are being mixed and then switch between databases while synthesizing the different languages. However, getting such data and maintaining recording conditions across all the TTS databases may be difficult. Also, it may be difficult to anticipate which languages are being mixed in advance.

The Blizzard Challenge [100] is an annual community-wide evaluation of TTS systems in which participants are given access to a common dataset to build synthetic voices from, which are then evaluated on a number of subjective metrics. From 2013-2015, the Blizzard Challenge included tasks on building systems for Indian languages. Since code mixing is very prevalent in many of the languages of the Indian subcontinent, the Blizzard Challenge added a multilingual task in 2014, in which English words were mixed with various Indian languages.

All the English words were written using the Latin alphabet, while the Indian languages were written in native script. The task was to synthesize test sentences containing such mixed sentences - however, the training synthesis databases contained no English in the recordings and corresponding prompts. There may have been some words written in the native script that were not native, like proper names, technical terms etc. in the data, but these were few in number.

Next, we describe our approach, the databases involved in these experiments and results from the Blizzard challenge.

6.3.1 *Techniques and Evaluation*

Since we only had data in the Indian languages to train from, we came up with a straightforward approach to deal with English words in Indian language sentences. When we detected a word in the Latin script, we used the US English text processing front end to process it, which meant that all Non-Standard Words (abbreviations, numbers etc.) that were covered by the (much higher resource) US English front end were also available to us. Then, we used a mapping between the US English phone set and the Indic phone set, which was common for all the lower resource Indian languages to convert the US English phonemes to Indic phonemes. This was a simple one-to-one mapping, which had its limitations, since some phonemes in English do not exist in the Indian languages and vice versa. Also, we could not incorporate contextual rules using this approach.

However, we found that even with this simple approach, our system performed well in the Blizzard Challenge in 2014 and 2015. The languages that were included in the 2014 version of the challenge were Hindi, Gujarati, Assamese, Rajasthani, Tamil and Telugu and in 2015, Marathi, Bengali, Malayalam were included and Gujarati, Assamese and Rajasthani were excluded. The evaluation metrics used were similarity to speaker and naturalness, evaluated by paid listeners and volunteers.

Although in general we did not perform well on similarity to speaker due to our choice of synthesis technique (Statistical Parametric Synthesis as opposed to Unit Selection), our performance on naturalness was good. In particular, we had the best systems for Gujarati, Telugu, Tamil, Rajasthani and for the naturalness metric in 2014 and Hindi in 2015. The naturalness metric is not ideal for testing pronunciation quality of multilingual systems, since many other factors may also influence naturalness. However, it seemed like our approach was viable and was not influencing the quality of the system negatively.

6.4 CODE MIXING WITH THE SAME SCRIPT

In the previous section, we described a simple technique to deal with mixed script synthesis by mapping phonemes from one language to another. This is a one-time cost especially if one of the languages already has a high-resource text processing front end.

In many cases, when languages are mixed in text, the same script is used for all the languages that are being mixed. This creates the additional complexity of having to identify which language the words belong to. In addition, people may not follow standardized ways of writing the language that is using the "wrong" script. We decided to extend the capabilities of our current system to also be able to deal with same-script code mixing for some language pairs.

6.4.1 *Data and Experiments*

Our first task was to collect Code Mixed data, for which we crawled a Hindi recipe website in which the recipes were all in native script (Devanagari). The recipe descriptions were all in Devanagari, with a few words in English, such as titles, numbers etc. Interestingly, most of the comments submitted by users were in code-mixed English-Hindi, all written in Romanized script. We collected around 70k sentences from this website to create a code-mixed corpus. Two example sentences from the corpus are shown below with their translations.

Heavy Cream kya hai Ye kaha se milegi
Translation: What is heavy cream, where can it be found?

Dear nisha mujhe hamesha kaju barfi banane mein prob hoti h plzz mujhe kaju katli ki easy receipe bataiye

Translation: Dear Nisha I always have a problem making Kaju Barfi please give me an easy recipe for Kaju Katli.

In the first sentence, there is one point at which the Code Switching occurs, and the sentence is well written with correct spellings for the English words and reasonable transliterations of the Hindi words. In the second sentence, we see multiple places in which English phrases and words are inserted ("Dear nisha", "prob" (problem), "plzz" (please), "easy receipe"). We also see that there are contractions and spelling errors. Some of the contractions ("h" for the word "hai") were common across the database and may be hard to normalize automatically in isolation even by humans.

Our goal for this part of the work was to be able to synthesize sentences from this corpus. We restricted our task to synthesizing Romanized text using our Hindi TTS system. Although this is may seem slightly artificial (synthesizing pure Romanized text using a Hindi voice), we can imagine finding the reverse case, where we have to use an English system to synthesize Romanized Hindi words that appear in Social Media posts, messages and emails. Also, as in the case of websites that have content in Devanagari but user submitted comments in Romanized Hindi and English, this setting may be more practical. We used the same Hindi database from the Blizzard Challenge data for all our experiments.

First, we wanted to see how much of a difference we could hope to make by knowing the pronunciation of the Hindi words. This involved both manually identifying the Hindi words in the Romanized sentence and replacing it with its correct (normalized) Hindi spelling, to be able to retrieve the correct LTS rules for it. This was, in a sense, "ground truth", or the best we could hope to get with our system.

We manually annotated and normalized 9 sentences by replacing the Hindi words with their Devanagari forms and synthesized them using our standard Hindi TTS system. We also synthesized the sentences using our current Indic front end without any normalization, which meant that all the words went through the English front end as described in the previous section. We asked 10 participants on Amazon Mechanical Turk to pick the system that was easier to understand, with a no difference option. Table 13 shows the results from this evaluation.

We expect that the Ground Truth system would be superior to the weak baseline in this case, but from the results we see that the preference for the Ground Truth was extremely high compared to the baseline. This is quite a large gap keeping in mind that the only difference

Table 13: *Subjective Evaluation of All-English Baseline vs Ground Truth*

Prefer Baseline	Prefer Ground Truth	No difference
18%	72%	10%

between the two systems was the pronunciation of a few Hindi words in the sentence. This indicated that it would be interesting to see how close we could get to the Ground Truth by automating this process.

Our approach was the following: given Romanized text, first identify the English words in the sentence. Then, normalize all the other words to their standard spellings and try to recover the pronunciation of the normalized words. The next few sections describe these steps in more detail and the assumptions we made.

6.4.2 *Language ID*

First, we identified the English words in the English-Hindi mixed text. Our motivation to identify English words and not Romanized Hindi words first was that it is easier to find training data or resources to identify English words automatically and also that people writing in Romanized scripts were more likely to spell English words in a standardized way than Hindi words.

We took a naive approach to solving the Language Identification problem: if a word in the sentence was present in CMUdict [10], then we considered it to be an English word. Otherwise, we treated it as a misspelling of an English word or a Romanized Hindi word. Some words are ambiguous in isolation, and if they exist in the US English lexicon, they currently get marked as English.

The Language Identification step can be improved by making use of letter language models, taking into account context, using standard LID models on trained corpora etc., as described in the related work.

6.4.3 *Spelling Normalization*

After filtering out the English words present in the US English lexicon from the sentence, we normalized the spellings of the rest of the words. To do this, we used the Soundex algorithm [101]. The Soundex algorithm encodes words such that spelling variants, which may have very similar pronunciation, are collapsed into the same code. Some characters such as those that represent vowels are ignored by Soundex.

Soundex only encodes consonants and ignores vowels. Each word is given a Soundex code, which consists of a letter followed by three digits. The letter is the first letter of the word and the digits encode the rest of the consonants in the word. Consonants that are similar share the same digit, for example, labial consonants (B, F, P, V) are encoded as the digit '1'. Soundex is used in many popular databases and is also used to index individuals and look up family names in the US census.

Taking the example of the words 'Smith' and 'Smythe', the Soundex algorithm works the following way. The first letter is retained in the code, and 'm' is mapped to 5 in both cases. The next letter, 'i' in 'Smith' and 'y' in 'Smythe' are both ignored. The letter 't' is mapped to '3'. The character 'e' is ignored in the case of 'Smythe'. In this way, both words have the same Soundex code and can be considered to be spelling variants according to this algorithm.

To normalize spellings in our data, we took the most frequent words of the code mixed recipe corpus as seeds and ran Soundex comparing each of these seeds to all the other words in the data. We formed clusters of spelling variants from these seeds by adding a word to the cluster if there was a Soundex match. Figure 5 shows a cluster formed with the seed word "recipe". Then, we replaced the low frequency members of these clusters found in the sentences we wanted to synthesize with their seeds. In case a word belonged to more than one cluster we chose the seed to replace it with randomly.

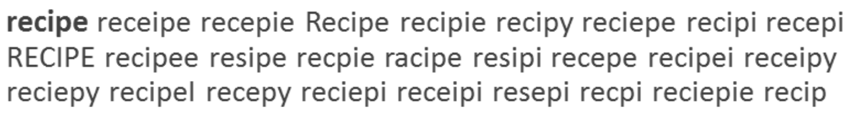
recipe receipe recepie Recipe recipie recipy reciepe recipi recepi RECIPE recipee resipe recpie racipe resipi recepe recipei receipy reciepy recipel recepy reciepi receipi resepi recpi reciepie recip

Figure 5: Spelling variants in the "recipe" cluster

In some cases (as in the case with the word "recipe") the seed words were English words that were found in the US English lexicon. In such cases, we treated these words as English words.

An extension of this could be to do a match phonetically, rather than by looking at vowels and consonants particularly for languages in which SoundEx implementations are not available. For this, we could look up the UniTran [28] expansions of words and compute a distance metric on SAMPA phoneme strings.

6.4.4 *Training the Romanized-Devanagari model*

After recovering the normalized spelling of the word in Romanized form, we then needed to recover its Hindi pronunciation. Instead of directly trying to recover the pronunciation from the Romanized form, we decided to reduce this problem into the problem described in the previous section, that is, synthesizing text that has mixed scripts, by transliterating Romanized Hindi into native Hindi script (Devanagari).

We explored transliteration standards and schemes that went from Hindi to Romanized, however, many of these standards used special marks such as diacritics to indicate vowel lengthening and most of them did not reflect how people actually type Romanized Indian languages on the web.

We trained a model on manually transliterated data from the FIRE dataset [102] that would give us a Devanagari word, given a Romanized word. We used 1000 code mixed Hindi-English sentences and extracted the words marked as Hindi, and the Devanagari forms of these words. We found 1800 unique Romanized Hindi - Devanagari pairs in the dataset. The Romanized Hindi in the FIRE dataset was clean data with very few or no spelling variations for each word, which meant that our previous step of normalizing spellings was critical.

To build a model from Romanized Hindi to English, we followed the standard procedure to build Letter-to-Sound Rules in Festvox [103]. Usually, the input to that is a string of characters and the output is a string of phonemes, but in this case, both the input and output were strings of characters.

We trained a Classification and Regression Tree for each grapheme in our Romanized Hindi word list which uses three previous and next grapheme contexts to predict Devanagari Hindi graphemes. Each Romanized Hindi grapheme aligned to none, one or two Devanagari graphemes.

6.4.5 *Evaluation and Results*

Once again, we carried out subjective evaluations on the Hindi TTS database with only the front end being changed in each condition. In each case, 10 listeners listened to 9 pairs of sentences.

Table 14 shows the preference for our technique, which we call Predicted, compared to the baseline where we treat all the words as English words, as described before. We can see that there was a significant preference for our method.

Table 14: *Subjective Evaluation of All-English Baseline vs Predicted*

Prefer Baseline	Prefer Predicted	No difference
16%	71%	13%

Next, we wanted to see how the Predicted system would compare against a system where all the Romanized words were treated as Hindi words. This is what would have happened if we had bypassed the Language ID step. Table 15 shows that there was a preference for the Predicted version, though it was not as significant as the preference over the All-English baseline.

Table 15: *Subjective Evaluation of All-Hindi vs Predicted*

Prefer All-Hindi	Prefer Predicted	No difference
30.5%	54%	15.5%

Finally, we wanted to see how close our Predicted system could get to the Ground Truth system. Table 16 shows that subjects had a preference for the Ground Truth system over the Predicted system, though the difference was not as high as the difference between the All-English baseline and the Ground Truth system.

Table 16: *Subjective Evaluation of Predicted vs Ground Truth*

Prefer Predicted	Prefer Ground Truth	No difference
28%	57%	15%

We explored using the Algerian Arabic-English corpus [93] mentioned earlier for similar experiments, however, without hand labeled examples of Algerian Arabic-English, training data was not available to build a model for transliteration. One solution may be to use a transliteration resource such as the Google Transliteration API to go from Arabizi to Arabic, however, this may not be appropriate for the Algerian Arabic dialect. Furthermore, the closest TTS voice available to us in terms of language was in Modern Standard Arabic. Preliminary experiments showed that all these factors made it difficult to replicate the Hindi-English experiments for this data. However, given a reasonable amount of code mixed text and a mapping between the letters and the phone sets of the mixed languages, it should be possible to replicate the experiments done for Romanized Hindi.

6.5 PHONEME SET MAPPING

The resources we used to make our Hindi TTS system capable of synthesizing Romanized Hindi and English words were a Language Identification System (in this case a US English lexicon), a moderate number of pairs of Romanized Hindi and Devanagari words, Soundex rules and a mapping between the US English and Hindi phoneme sets. While this mapping is a one-time cost, it requires some knowledge about the phonetics of both languages.

In addition, it may not always be possible to find good phonetic mappings between languages because some phonemes in one language may not exist in the other. In such cases, bilingual speakers may map phonemes to the closest possible phoneme, or borrow phonemes. 9 shows the mapping between English phones from the phone set used in our standard US English build and Hindi phonemes from the Indic phone set, along with the phonetic features assigned to each phoneme in both sets. In some cases, we did not map phonemes from English to very low frequency phonemes in our Hindi corpus, but chose to replace them with more frequent phonemes.

In all the experiments described above, we mapped phonemes from English to Hindi manually. However, we wanted to automate the process as much as possible. This step may not be necessary if we use a common phone set for both languages, as we do in case of UniTran and the Indic languages within Festvox.

6.5.1 *Related Work*

The approach described by Nerbonne et al. [104] uses Levenshtein distance to measure the distance between words in different dialects of Dutch, and uses this to group dialects together. A common set of words are compared across all dialects, with the distance being compared based on letters with different weights for insertions, substitutions and deletions. [105] extend this work by using phonetic features and a variety of distance metrics.

Sriram et al. [106] describe a technique for multilingual query processing, in which words in the queries are converted into a language independent 'common ground' representation, after which a weighted phonetic distance measure is used to match and rank queries. Phonetic features include vowel rounding, frontness, height, length, voicing, aspiration etc. which are the same features that are used in the standard Festival phonetic feature set. The authors also suggest weights that can be given to these phonetic features, since some of them may be more important than others while calculating phonetic similarity.

Le et al. [107] used a top-down bottom-up knowledge based approach for calculating phoneme similarity. They use models of similar phonemes cross lingually to create context dependent Acoustic Models in new languages. They used IPA rules to create a hierarchical graph of phoneme similarity, which splits phonemes into categories like Consonant-Vowel, Close-Back Vowels, Close-Front, Close-Back vowels etc. Acoustic Models were built with multiple languages and used for recognizing Vietnamese speech with a small amount of adaptation data.

Melnar et al. [108] describe an approach to measuring phoneme distance cross lingually by representing phonemes as a feature matrix, with feature weights based on lexical frequency. In addition to traditionally used phonetic features, they also include corrolary features that represent allophonic realizations of the phones. This approach was shown to perform well on cross-lingual ASR tasks.

Pucher et al. [109] describe phonetic similarity measures for ASR grammar optimization in which they compare minimum edit distance based measures, perceptually motivated measures and HMM-distance based measures and correlate them to word confusion in the ASR.

6.5.2 *Experiments*

We implemented a simple weighted distance based metric to map English and Hindi phonemes in our system based on [106]. We conducted subjective tests comparing our manually mapped phonemes to the automatically mapped phoenemes. However, we found that in subjective tests, listeners had a very significant preference for the manually mapped phonemes. This may be due to the fact that the manually assigned phonetic features may not be completely correct, and many phonetic features ended up getting the same weight in [106].

Ideally, we should be able to automatically set these weights or learn this mapping from data. Recent work in creating vector representations of acoustics [110] could be a promising direction for creating such mappings automatically.

6.6 CHAPTER SUMMARY

This chapter presents preliminary work towards solving the problem of synthesizing code mixed text. With the advent of smart phones and digital assistants in multilingual societies, we believe that this will be a very relevant problem to address in the future for speech processing. We provided a framework to synthesize code mixed text that came from Social Media in and presented experiments on Romanized Hindi-English using a corpus that we crawled from the web. Some of the ideas presented in this chapter, such as spelling normalization can also be used while synthesizing single-language text from Social Media.

We conducted subjective listening tests to compare speech synthesized with our framework to baselines that treated all the words in the sentence as English words or Hindi words, and showed that there is a preference for speech synthesized with our technique. We also compared our technique to gold standard manually labeled and transcribed sentences with subjective tests and showed that there is still a gap between the two, which can be addressed with better Language Identification, spelling normalization and cross-lingual pronunciation rules. In all cases, we used minimal resources to extend the capability of our current system to make it capable of synthesizing code mixed text. In addition, better cross-lingual phonetic mapping techniques that make use of acoustics may eliminate the need to map phonemes manually. We did not explicitly address Code Mixing that takes place at the morpheme level in our technique, and this would be an interesting future direction.

7 TEXT TO SPEECH WITHOUT TEXT

7.1 INTRODUCTION

So far, we looked at various techniques to improve pronunciation of TTS systems in low resource languages. We made the assumption that the languages have a standardized orthography, and either treated graphemes as phonemes, used lexicons or used Letter to Sound rules to find pronunciations. In case of code mixing, where non-standard spellings may be used, we introduced a framework in which the spelling of a word can be normalized in order to improve chances of it being pronounced correctly, once the language it belongs to has been identified.

Many of the languages of the world are not written, but are only spoken. Many languages do not have their own written form, but use the writing system of another language which may or may not be related to them. For many spoken languages of the world, finding large corpora or linguistic resources is difficult. Yet, some of these languages have many native speakers around the world and it would be very interesting to deploy speech technologies in them.

This part of the thesis deals with building TTS systems for languages that are purely spoken languages: they do not have a standardized writing system. These languages could be mainstream languages such as Konkani (a western Indian language with over 8 million speakers), or dialects of a major language that are phonetically quite distinct from the closest major language.

Building a TTS system usually requires training data consisting of a speech corpus with corresponding transcripts. However, for these languages that aren't written down in a standard manner, one may be able to only find speech corpora. This chapter focuses on building speech synthesis systems when our training data does not contain text.

The techniques described here can also be applied to languages where transcripts are not available for the corresponding speech data, even if the language does have a standardized written form. These techniques can be used when creating transcripts manually or having an Automatic Speech Recognizer in the language create the transcripts is not feasible. Another scenario in which these techniques can be used is when the written form of a language is not suitable for a spoken version of the language and an automatically created writing system is desired.

The obvious question that arises while trying to build speech processing systems and specifically TTS systems for languages without an orthography is whether there is any application that such a system could be used for. If there is no text at training time, there would be no text at synthesis time either. However, consider the case of having a TTS system that does not use text but uses some other representation to synthesize from.

Such a system could be part of a dialog or Speech-to-Speech translation system. Let us imagine that we have a system that translates spoken Hindi, a higher resource language with a written form to Konkani speech, with Konkani not having a standardized writing system. We can imagine using some intermediate representation that can act as a text form that the machine translation system can produce.

Going back to the block diagram introduced in Chapter 2, we see in Figure 6 additional missing resources, which are the transcript at training and synthesis time. In addition, we can assume that for such languages, there may not be other lexical resources available.

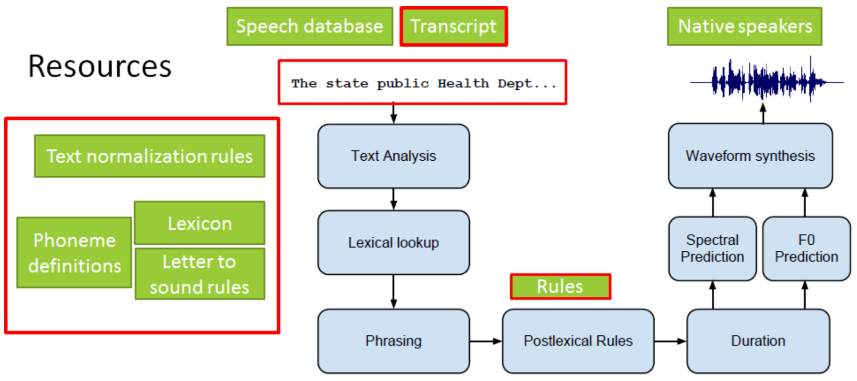

Figure 6: Missing resources for TTS without text

Our goal in this part of the work was to derive a written form that is sufficient to synthesize from, given only speech data in the target language. We relied on acoustics and cross-lingual techniques to come up with written forms and produce understandable synthesis. We built TTS systems for several languages that ranged from high to very low resource languages, with some languages not having their own standardized written form.

7.2 BASIC CROSS-LINGUAL APPROACH

In order to derive units that could be used as part of a written form, our approach uses Automatic Speech Recognition to decode speech in the TTS databases in the target languages without an orthography. However, since the language does not have an orthography and is probably a low resource language, we cannot assume that an ASR system exists in the language. So, we use an ASR system built for another language, and we perform phonetic decoding, rather than word-level decoding. Ideally, we use an ASR system in a language that is related to the target language. Figure 7 shows a block diagram of the components and flow of our approach.

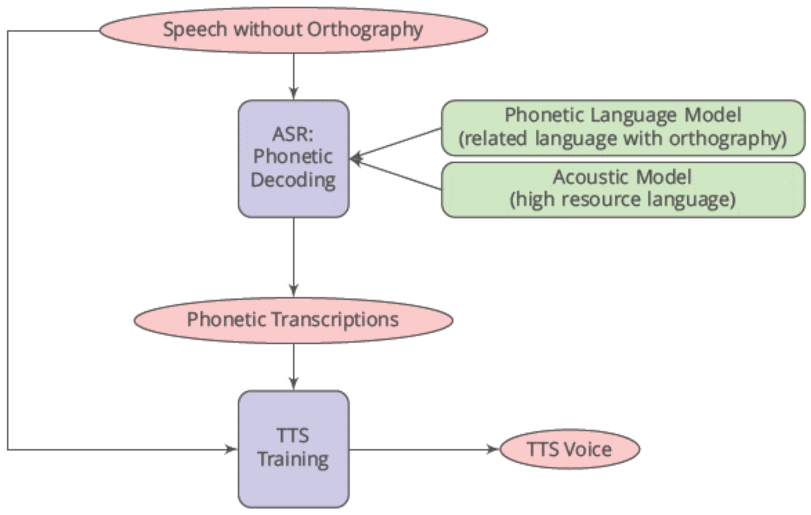
Speech without Orthography
ASR: Phonetic Decoding
Phonetic Language Model (related language with orthography)
Acoustic Model (high resource language)
Phonetic Transcriptions
TTS Training
TTS Voice

Figure 7: Cross-lingual Phonetic Decoding

First, we decode the audio in the target language with a phonetic decoder using an acoustic model and language model from another language. Then, we build a synthetic voice using the transcript obtained from the phonetic decoder. This transcript contains phonemes in a language that we know about, so we can use phonetic features and other phonetic information while building the voice. Once we build the voice, we evaluate it using the Mel-Cepstral Distance (MCD) mentioned in earlier chapters, which is an objective metric of TTS system quality.

It is easy to see that the quality of the transcript obtained by one pass of decoding using the cross-lingual decoder may not be ideal. So, using the transcripts obtained from decoding and the TTS speech corpus, we iteratively build new targeted acoustic models and use them to decode the speech again. We use the phonetic transcripts to build synthetic voices and evaluate them objectively at each stage of the iteration. Once the MCD stops improving, we stop the iterations. This process is shown in Figure 8.

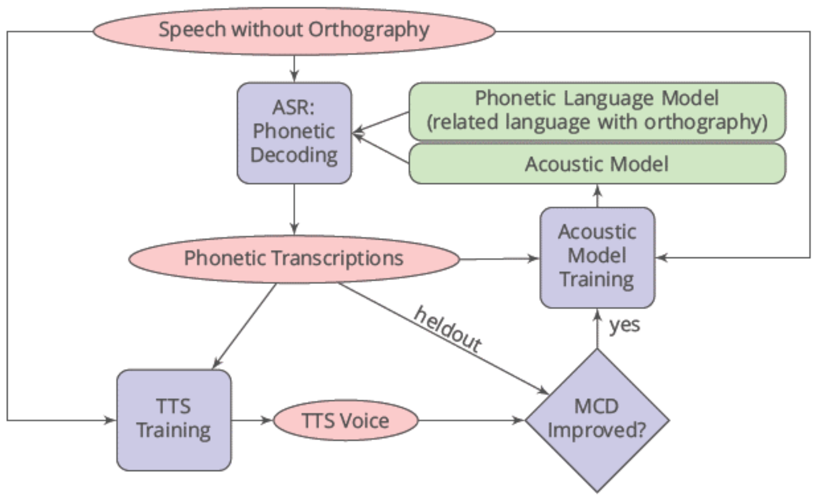

Figure 8: Iterative process to choose better pronunciations

7.3 RELATION TO PRIOR WORK

Speech to speech translation typically involves a cascade of three models: an Automatic Speech recognition System (ASR) in the source language, a Statistical Machine Translation system (SMT), and a Text to Speech (TTS) System in the target language. Generally, these three models are developed independently of each other. Recent work such as [111], [112], [113], [114] has looked into deeper integration of this pipeline, but the general assumption here is that the target language has an orthography.

If the target language of speech to speech translation does not have a written form, it has been proposed that one be defined, though training people to use it consistently is in itself very hard and prone to inconsistencies e.g. Iraqi Arabic transcription techniques in the TRANSTAC Speech to Speech Translation Project [115]. Our proposal is to use a phonetic-like representation of the target speech, derived acoustically as the orthography to use. Stuker et al. [116] have investigated such an approach.

Bacchiani et al. [117] describe a technique to derive segmental units based on acoustics for ASR, and then map them to phonemes in a lexicon. They use Dynamic Programming to derive segmental units assuming that these units follow the polynomial trajectory model. The motivation behind this work is to find segmental units that can capture word transitions that may be more suitable for conversational speech recognition.

ASR for unwritten languages has been studied very recently [118]. In this work, pronunciations for "words" in the target unwritten language are generated automatically by using cross-lingual phonetic decoding, and using a SMT model to translate words from a resource rich language into inferred words in the target language. Using multiple source languages improves performance of the ASR. In [119], grapheme-based ASR systems are used for decoding speech, and SMT is used for phrase-based translation between clustered graphemes and the n-best list output of the ASR. This is then used to automatically construct a lexicon that can be used for ASR for low resource languages without lexicons.

Changes have been proposed to SMT modeling methods [120] to specifically deal with phoneme strings in the target language. In order to induce the automatic phonetic writing form, we use an ASR system in a foreign language and adapt the acoustic model to match the target speech corpus. Speech synthesis voices are typically built from less data compared to speech recognition systems.

Although we can adapt our models instead of rebuilding targeted models at each iteration, Acoustic Model adaptation with limited resources can be challenging [121]. Zavaliagkos et al. [122] have recently proposed a rapid acoustic model adaptation technique using cross-lingual bootstrapping that showed improvements in the ASR of under-resourced languages. Our model adaptation technique is somewhat similar to that method, but we optimize the adaptation towards better speech synthesis, and have only acoustic data in the target language.

Although such representations may be difficult for a native speaker to write, an SMT system can help bridge the gap from a source language to the target phonetic representation of the language. The technique described by Elsner et al. [123] models pronunciation variability based on articulatory features and is very suited for our purpose, since the ASR transcript could be noisy.

7.4 DATA AND RESOURCES

7.4.1 *TTS databases*

We used TTS databases from eight languages, various scripts and diverse language families for this research. Our audio data ranged from almost two hours of speech to less than six minutes, as shown in Table 17.

Table 17: *Languages, Sizes and Scripts for TTS without Text*

Language	Script	Size (minutes)
English	Latin	111
Dari	Arabic	52
Iraqi	Arabic	62
Pashto	Arabic	39
Thai	Thai	25
Ojibwe	Latin+	12
Inupiaq	Latin+	5.5
Konkani	Various	5.5

Our English data was from the Blizzard Challenge [100] 2013 audio book task, recorded by a professional voice recording artist.

Dari is a dialect of Persian that is used in Afghanistan as an official language and also spoken in parts of Iran and Tajikistan. It has over 18 million native speakers. Pashto is a also an official language of Afghanistan and has over 40 million speakers. The Dari and Pashto corpora are from the DARPA TRANSTAC project. Iraqi Arabic is a dialect of Arabic spoken in Iraq and has about 15 million speakers. The Iraqi Arabic corpora were provided by BBN as part of the DARPA BOLT project.

The Thai language is spoken by over 20 million people and is the official language of Thailand. We used the Thai speech corpora from the SPICE [35] dataset.

Inupiaq is an Inuit language spoken by about 2100 people in northern and northwestern Alaska. Ojibwe is spoken in Canada and the United States and has around 56000 native speakers. Both Inupiaq and Ojibwe use the Latin script in their written forms. Our data for Inupiaq and Ojibwe came from a corpus collected as part of the Endangered Languages project at Carnegie Mellon University.

Konkani is an official language of India and is used primarily in Goa and Karnataka. It has over 8 million native speakers. Konkani does not have its own script, and native Konkani speakers use Devanagari, Latin, Kannada, Malayalam and even Arabic scripts to write it. We used a corpus of Konkani from the CMU SPICE project [35].

7.4.2 *Decoder and ASR resources*

We used the CMU Sphinx [45] speech recognition toolkit in allphone mode as our phonetic decoder and to train new acoustic models. The allphone mode allows us to get a phonetic decoding of the speech even if we use context dependent models.

Our phonetic decoder used trigram phonetic language models built from German and Marathi data. For the German language model, we used the Europarl [124] corpus and for the Marathi language model, we used a corpus created by collecting news stories from a Marathi news website, Esakal. We used a single acoustic model, the Wall Street Journal (WSJ) English acoustic model provided with CMU Sphinx.

For the experiments described here, we used the English WSJ Acoustic Model for decoding all languages, although we have experimented with using other Acoustic Models in previous work [62]. Using the WSJ acoustic model for decoding English speech is not fair, but we used it to keep the Acoustic Model consistent in all our experiments.

Ideally for phonetic decoding, an acoustic model and phoneme language model from a closely related language should be used. To simulate this in our experiments, we used Marathi and German phonetic language models, as listed in 18. In some cases, we tried to use Language Models that were related to the target language in terms of language family (for example, Marathi for Konkani and German for English) while in other cases the choice was arbitrary.

Table 18: *Language Models used during decoding*

Language	Phonetic LM
English	German
Dari	Marathi
Iraqi	German
Pashto	Marathi
Thai	Marathi
Ojibwe	German
Inupiaq	German
Konkani	Marathi

7.4.3 *Voice building*

We used the Festvox voice building tools to build CLUSTERGEN [125] voices for the Festival [126] speech synthesizer. Our method can be used with any waveform synthesis technique, since the written form only influences the front end of the system. We used the TestVox [40] tool to run listening tests online.

7.5 OBJECTIVE RESULTS

After decoding speech in the target language using the appropriate acoustic and language models, we iteratively trained new acoustic models using the decoded transcript as the text and the original audio as the speech. At each stage of the iterative process, we calculated the Mel Cepstral Distortion (MCD) of the voice built using the decoded transcript from that iteration. We will now look at the MCD of voices built using these transcripts for various languages.

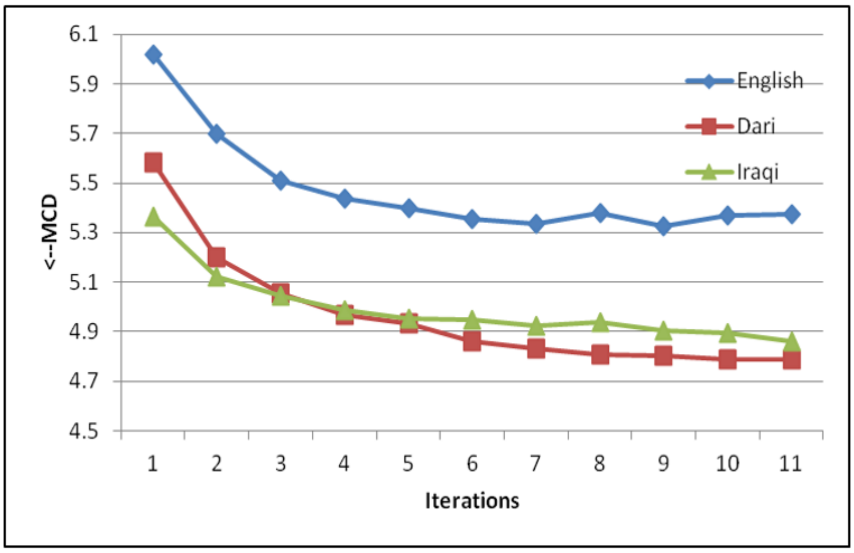

Figure 9: MCD across iterations for > 1 hour speech

Figure 9 shows the MCD for voices built for English, Dari and Iraqi Arabic. English has about two hours of speech while Dari and Iraqi Arabic have about one hour of speech. We see that there is a big drop in MCD value from the first iteration to the second, in which the targeted acoustic model is built. In the case of English, iteration 7 has the lowest MCD, after which it rises slightly. For Iraqi Arabic and Dari, the MCD continues to fall until the last iteration.

Figure 10 shows the MCD graph for Pashto and Thai, both of which have around 30 minutes of speech. We see that the MCD for Pashto in

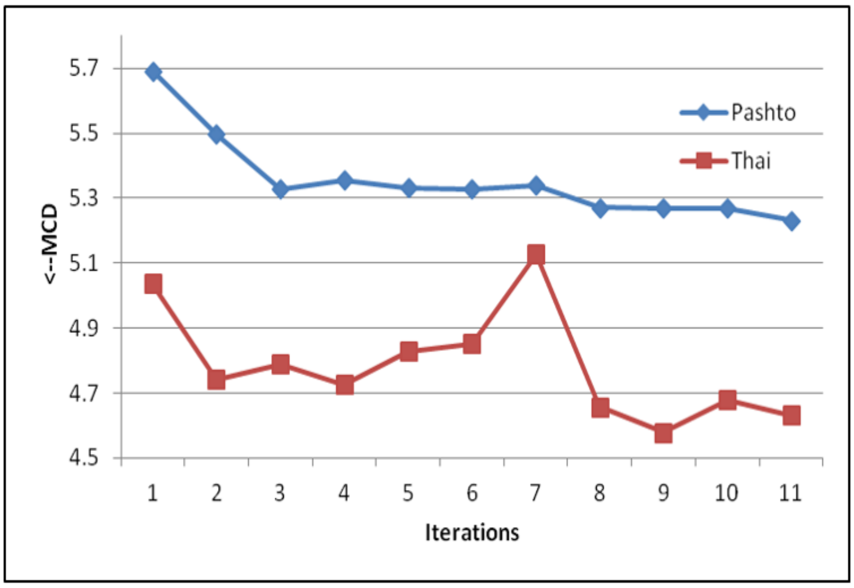

Figure 10: MCD across iterations for 30 minutes speech

the first three iterations falls rapidly and then does not change much, while for Thai, there is a big drop after the first iteration, which is consistent with the results for English, Dari and Iraqi Arabic. There is a large rise in MCD at iteration 7 for Thai, but it falls again in the next iteration. We can see that even with half an hour of speech, our iterative method produces better transcripts than the base decoding with the WSJ acoustic model.

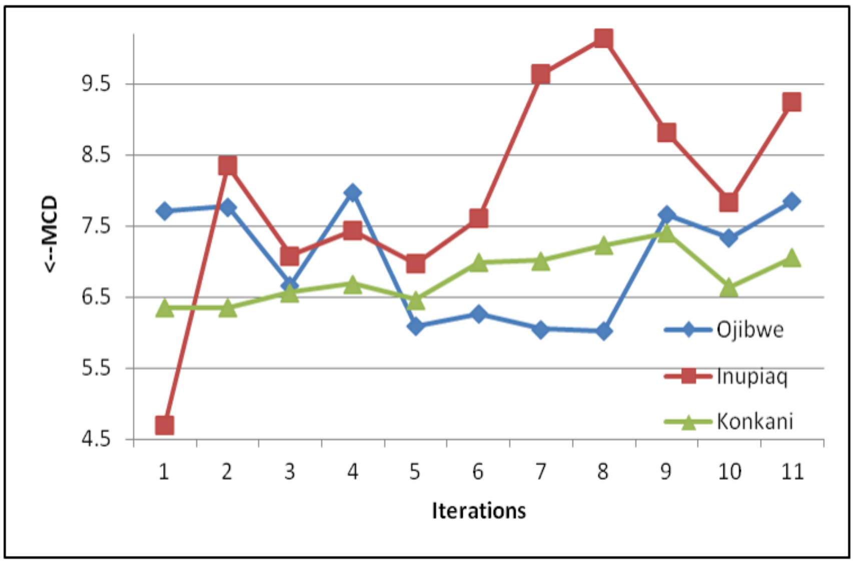

Figure 11: MCD across iterations for < 15 minutes speech

Figure 11 shows results for Ojibwe, which has 12 minutes of speech and Inupiaq and Konkani, both of which have around five minutes of speech. We see that for Ojibwe, the MCD rises slightly after the first iteration and then falls after the fifth iteration, with the difference in the MCD between the base and best iteration being 1.43. This shows that even with just 12 minutes of speech, the iterative method is able to come up with a better transcript than just the base decoding. However, for both Inupiaq and Konkani, we see that the MCD rises after the first iteration. This is probably because the amount of speech is too small to build reasonable targeted acoustic models.

Overall, we see that with a moderate amount of speech data, the iterative targeted acoustic models produce better phoneme transcripts than just using base decoding from a cross-lingual phonetic decoder, shown here on a variety of languages.

Throughout the iterations, we kept the language model used by the ASR consistent. One extension of this approach is to adapt the language model at each iteration. However, preliminary experiments on interpolating the original language model at each iteration with the new transcript did not yield improvements in MCD. Another extension of this approach is to do Acoustic Model adaptation, instead of rebuilding new targeted Acoustic Models at each iteration.

7.6 INDUCING HIGHER LEVEL UNITS

So far, we have discussed the bootstrapping method which produces phoneme transcripts of the audio, which may be noisy. When we build TTS systems using these transcripts with the framework described above, the system treats each phoneme as a word.

Text to Speech systems typically benefit from using syllable and word level information. Some lexicons contain syllable and stress information, which can be used as features while building models. Knowing where the words are in a sentence can help with phrasing, prosody and duration. Many languages that have a standardized written form indicate words through the use of spaces between words, although sometimes what a word is may be poorly defined, especially when it comes to spoken language.

In order to test how much of a difference it makes to know where the words are, we synthesized voices in English and German using 1. Oracle Phones and 2. Oracle Words. In the Oracle Phones case, we used the actual phonemes that the knowledge-based front ends gave us, but without any word boundary information. In the Oracle Words case, we used the knowledge-based front ends as we typically do, when we build TTS systems for languages with written forms. So, the only difference was the presence of word boundary information in the Oracle Words voices. Table 19 shows the MCD of voices built with these configurations.

Table 19: *MCD of Oracle Voices*

Language	Front end	MCD
English	Oracle Phones	2.46
English	Oracle Words	2.15
German	Oracle Phones	2.12
German	Oracle Words	1.75

We can see that the Oracle Words voices are significantly better in both cases. Our task was now to find word boundary information for languages without a written form, for which we had decoded phonetic transcripts.

We took two approaches to this problem: first, we tried to group phonemes in the phonetic transcripts together into higher level units. Second, we tried to induce higher level units from acoustics.

7.6.1 *Higher-level units from transcript*

We performed experiments on inducing higher level units for two languages - English and Dari.

First, we grouped the phonemes from the best phonetic transcript (according to MCD) for English and Dari into syllable-like units. To obtain syllables, we use heuristic rules based on the maximum onset algorithm [127] built into the Festival speech synthesizer to join phonemes in the transcripts into syllables. The maximum onset algorithm assigns consonants to the syllable onset rather than the syllable coda. We treated the syllables as words and added appropriate entries in the lexicon.

Next, we induced word-like units by using cross-lingual information. We trained a Conditional Random Field (CRF) model on German word boundaries and phonemes. Our task was to predict word boundaries given a string of phonemes.

We created training data for the CRF by extracting phonemes and word boundaries from the German Europarl data. We used CRF++ [128] to train a German model that could group phoneme sequences into word-like units and ran the model on the best English and Dari transcripts. We discarded words that were rare (< 300 in frequency) and used the rest of the hypothesized words in our transcripts. We added appropriate lexical entries for these words and built voices for English and Dari.

The following example taken from the English voice show the original transcript, transcript obtained by cross-lingual decoding and the higher level units obtained by inducing syllable-like units.

Original transcript: Black Beauty the autobiography of a horse
Cross-lingual phones: JH M B L EH K D IY D IY K P V IY AH L OW B AY EH L B AH R F IY AH B AH HH AO S D
Syllables: JHMBLEHK DIYD IYK PVIY AHL OWB AY EHLB AHRF IY AHB AHHH AOSD

Table 20 shows the result of syllable and word induction. We see that both for English and Dari, grouping phonemes into syllables decreases the MCD of the new voice. Surprisingly, this difference is very large in the case of Dari. The voice built for English using CRF word induction has a slightly lower MCD than the syllable method. However, this method does not seem to make much of a difference in the case of Dari. This could be because we used a German word model, and German word rules are quite different from Dari.

Table 20: *MCD of Syllable and Word Voices*

Language	Best Iteration	Syllables	Words
English	5.32	5.26	5.25
Dari	4.78	4.16	4.76

7.6.2 *Higher-level units from acoustics*

So far, we used syllabification rules and cross-lingual word boundary detection techniques to group the ASR-derived phonemes into higher level units. Even though the notion of a syllable is fairly consistent across many languages, some languages do not have formal notions of words, have rich morphology and are agglutinative. So, we used a technique to derive higher level units by using information from acoustics.

To do so, we automatically induced units known as accent groups, which have been used recently for prosody modeling in speech synthesizers [129].

An accent group is defined as being a group of syllables with one pitch accent. Accent groups are similar to the notion of metrical feet, without having an explicit definition of how to group syllables, other than the constraint that each accent group contains only one syllable. The technique for deriving these units from speech is completely data driven. The complete description of accent groups and training strategy is provided in [129].

Accent Groups are derived by analyzing the pitch contour in tandem with the syllable sequence and approximate it with a synthetic contour described as a sequence of TILT shapes [130] over parses of syllable groups. The optimal parse on the syllables is one that minimizes the reconstruction error of the target pitch contour.

A stochastic context free grammar is trained on such parses of accent groups, so as to allow prediction of accent groups for unseen sequences of syllables. In order to uniquely identify syllables, we tag each syllable with the vowel name, the onset and coda categories as described in [130]. These categories are only a few in number and yet are language independent, allowing us to use this approach for arbitrary new languages here.

Given such parses derived acoustically from the pitch contours on all of training data, a grammar is trained to predict parses of unseen sequences of tagged syllables. This is further improved with decision trees about the positional information of each syllable, so as to reliably estimate for each syllable boundary, if there is a accent group boundary, or not.

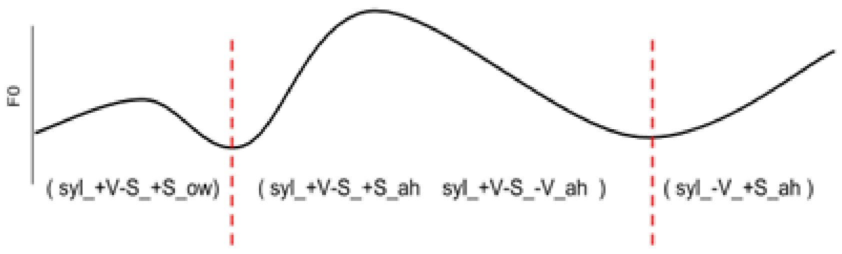

Figure 12: Accent Groups

Figure 12 shows accent groups derived automatically from parses of tagged syllables.

7.7 SUBJECTIVE EVALUATION

From our objective results mentioned earlier, we saw that the voices built using syllables were better than the voice built on the best iteration using phonemes. Word induction seemed to help in the case of English, but not Dari.

To test this subjectively, we conducted listening tests comparing the voice having the lowest MCD and the voice with syllable units for both English and Dari to see if grouping phonemes together into syllables was perceptually better. Table 21 lists the results from tests on English and Dari. In both cases, we see that participants preferred the voice with syllabified transcripts significantly more than the best iteration.

Table 21: *Subjective Preference for Syllable Induction*

Language	Participants	Prefer Best Iteration	Prefer Syllable	Can't say
English	7	4%	68%	28%
Dari	5	6%	72%	22%

Next, we carried out subjective tests in English, in which we compared the voice built with Accent Groups to the baseline best iteration, syllable and CRF word induction voices. Table 22 shows results for the listening tests in which 10 subjects listened to 10 pairs of sentences in each condition.

Table 22: *Subjective Preference for Accent Groups vs Other Voices*

Voice A	Voice B	Prefer A	Prefer B	Can't say
Best Iteration	Accent Group	12%	78%	10%
Syllable	Accent Group	47%	43%	10%
CRF words	Accent Group	22%	70%	8%

The results indicate that significant gains can be obtained by induction of the speech-derived accent group units, as opposed to word derivations through CRFs over phoneme transcriptions. While it is encouraging that the Accent Group voices perform comparably, syllable voices remain the most reliable units that can be induced in the current setting. This is perhaps due to the unavailability of sufficient data, or features that effectively capture the contextual information in building voices using higher levels of phonology.

7.8 INFERRED PHONEMES FOR TTS WITHOUT TEXT

Recent work that has built on this general framework for building TTS systems without a standardized orthography has used Inferred Phonemes derived from Articulatory Features [46], described earlier in Chapter 5. Articulatory Feature classifiers are trained cross lingually on a labeled database using a three hidden-layer network and are then used to label a database without transcripts. The Articulatory Features are then clustered into units known as Inferred Phonemes. The best phoneme transcript from the iterative technique described earlier is used to infer phoneme boundaries, so the Inferred Phonemes are similar to ASR-derived phones in duration. The Articulatory Features averaged over each Inferred Phoneme is used in place of traditional phonetic features that would be used with known phones. Adding the average AF information was found to improve the MCD of voices built for Hindi and Dari.

One of the decisions that needs to be made is the number of IPs to derive during clustering, which can be controlled by changing the stop value of the Classification and Regression Tree. In experiments conducted on Hindi, Dari and Iraqi, having 140-180 IPs yields the best MCD scores.

Recently, we have also used Articulatory Features, Inferred Phonemes and Cross-lingual phonetic decoding for the task of minimal pair discrimination in the Zero Resource Speech Challenge in Interspeech 2015 [85]. The evaluation metric was minimal pair ABX discrimination witin and across speakers, and we took the approach of inducing a discriminative set of units that would be optimized for synthesis, with promising results.

7.9 CHAPTER SUMMARY

In this chapter, we addressed the challenge of building a Text to Speech system for a language without a standardized orthography. While in the previous chapters we had access to an orthography which we could normalize or use directly to look up pronunciations, in this case, we derived an orthography by using acoustics from the TTS database.

We applied an iterative cross-lingual decoding technique and derived a phonetic written form for eight languages from various language families. We saw that with as little as half an hour of speech, we could get improvements in objective measures of TTS system quality with voices built with these transcripts over the baseline decoded transcripts.

We also used techniques to group phonemes in these transcripts into higher units, which led to higher quality voices, both objectively and subjectively. In addition, we described a method to use acoustic information to identify accent groups to create higher level phonological units. Our results indicate that inducing such units leads to a large improvement in both objective metrics and subjective preference.

Although the quality of the systems built using this technique is not as high as voices built with transcripts, the results are promising. Such systems may be usable in limited domains, for found data and in languages with very few resources. Currently, this work is being extended to build Speech Translation systems for unwritten languages [131] for the hospital domain, to help refugees who speak languages that do not typically have a written form. The techniques described in this chapter can also be applied to found data, or speech in a written language for which transcripts are not available.

8 CONCLUSION AND FUTURE DIRECTIONS

In this thesis, we presented several techniques to improve the intelligibility and naturalness of Speech Synthesizers built for low resource languages. We focused on improving the pronunciation, specifically the grapheme to phoneme mapping or lexical lookup part of Speech Synthesizers for such languages. Our basic hypothesis was that the pronunciation of systems for low resource languages can be improved by exploiting models built cross-lingually, and also utilizing the acoustics of the TTS database.

Throughout the thesis, we used standard TTS evaluation techniques to evaluate our voices objectively and subjectively. We conducted experiments on various languages of the world from different language families and having different types of writing systems.

First, we presented a grapheme-based baseline to build synthesizers for most languages of the world that have a written forms, by using a resource that maps Unicode characters to phonemes from a known phone set. This improves upon raw grapheme based techniques in which we treat all graphemes as phonemes and know nothing about the phonemes and their features.

Next, we compared our grapheme-based voices built with better modeling techniques to knowledge-based voiced built with standard techniques to see if we could implicitly model pronunciation with these models. We found that by using better modeling techniques, we could reach the performance of voices built with knowledge-based front ends, including large lexicons and hand-written letter to sound rules, for some languages.

In order to discover specific letter-to-sound rules that may be slightly different in related languages, we proposed a technique to cross-lingually discover schwa deletion rules in Hindi, by using Assamese as a higher resource language.

Some languages may have lexicons with ambiguities that are not easy to resolve without external resources, which may not be available in the low resource setting. We presented a technique to disambiguate homographs in dialects of Arabic by making use of acoustics.

We presented some results and analysis of using Inferred Phonemes derived using cross-lingual Articulatory Features to improve grapheme-based voices. There is some evidence that such features capture information from the acoustics that can significantly improve the quality of voices. However, the Inferred Phonemes are difficult to predict using only text.

A common phenomenon seen in social media and instant messaging in bilingual and multilingual communities is code switching or code mixing. Speech Processing systems need to be able to deal with such mixed language both during recognition and synthesis. In many cases, the script used to write the language being mixed is not appropriate for the language. In addition, social media text contains non-standard spellings and contractions. We presented a preliminary framework to synthesize code mixed text in Romanized Hindi and English using our standard Hindi voice.

Lastly, we described a technique to automatically discover a written form using cross-lingual phonetic decoding for languages without a standardized writing system. We built voices for a variety of languages using our discovered written form and improved them by inducing higher level units using both the transcription and acoustics. Speech Synthesizers built using this technique can potentially be used as part of applications such as Spoken Dialog Systems and Speech Translation systems for languages that do not have their own written form, or databases that do not have a transcription available.

In this thesis, we presented several techniques that improve the pronunciation, or grapheme-to-phoneme mapping of systems, or in the case of TTS without text, find a phonetic transcript that we can synthesize from. In most cases, we found improvements in subjective and objective metrics. However, it can be argued that not all the improvements we saw were due to the pronunciation or the phonemes themselves. In a sense, our techniques provided better input to downstream Machine Learning algorithms (such as labeling, predicting the spectrum, duration), that then improved the system overall.

8.1 FUTURE WORK

Recently, there has been a lot of interesting in vector-representations of words based on semantics [132]. These vector representations have been used for various applications including Statistical Machine Translation. This idea has been applied to phonemes as well, in which acoustics are used to generate embeddings of phonemes [110] and acoustics based context embeddings [133].

Recently, we have conducted preliminary experiments on adding vector representations of phonemes derived from lexicons as features available to our Machine Learning models for modeling the spectrum. Adding these automatically derived features to our system that has no knowledge of the phonetic features of phonemes seems to yield similar objective scores as a voice built with knowledge of phonetic features. This could be very useful in the raw grapheme scenario, where we do not have any phonetic feature information. We are also exploring how such vectors can be used cross-lingually in languages that have similar phoneme sets. Currently, these vectors are derived from lexicons and do not take into account any acoustic information. Future work in this direction includes integrating acoustics into such representations and using them as features in our models.

In this thesis, we used Crowdsourcing for carrying our subjective listening tests on Amazon Mechanical Turk. The crowd can potentially be used to pinpoint and correct specific pronunciation errors that the TTS system is making, particularly in cases when native speakers of the language are not easily accessible in person.

Text normalization is an important challenge when building a front end for a new language, which we did not address in this thesis. It may be possible to exploit cross-lingual resources from related languages to automatically build text normalization rules for low resource languages. Since speakers of some of these languages may be available as crowd workers, some of these rules or exceptions may potentially be crowd sourced.

In this thesis, we focused on grapheme-to-phoneme conversion and lexical lookup for pronunciation modeling. However, there are many other factors that influence pronunciation, such as lexical stress in languages like English and tones in tonal languages. Some of the techniques we have described can potentially be applied to such dimensions of pronunciation as well, if they can be reliably extracted from acoustics.

We provided a simple framework for dealing with code mixed text in this thesis. Future work includes discovering better techniques to do Language ID from the point of view of getting a better pronunciation of the word. Recent work on lexical borrowing has shown that it is possible to identify donor words in a resource rich language to obtain translations of Out of Vocabulary words in low resource borrower languages, which leads to an improvement in Machine Translation [134]. Identifying such donor words may be useful in figuring out the pronunciation of unknown words in code mixed languages.

A decision may need to be made about the degree of foreignness of the pronunciation of a word depending on the application and audience, which is an interesting research direction. This could be of particular relevance for conversational agents and Spoken Dialog Systems deployed in multilingual societies, both for Speech Recognition and Synthesis.

In this thesis, we assumed that most of the text we encountered was well written, if at all written, except in the case of Code Mixed text, where we tried dealing with non-standard spellings. A Speech Synthesizer capable of dealing with Social Media text such as Twitter would need better techniques to normalize words with misspellings and contractions.

Finally, in this thesis, we used listening tests and standard objective metrics to evaluate the pronunciation of our systems, although we found that in a few cases we had to ask people to listen to specific words during subjective listening tests. We also found that the objective metric we used, the Mel Cepstral Distortion, was not always sensitive to phenomena such as schwa deletion. Finding better objective and subjective metrics for evaluating pronunciation and speech synthesis in general is an important future direction.

9 APPENDIX A

9.1 US English-Hindi phone mappings

English phone	Hindi phone
aa	A:
ae	e
ah	A
ao	o
aw	aU
ax	A
axr	A
ay	aI
b	b
ch	c
d	dr
dh	dB
eh	e
er	E 9r
ey	ay
f	ph
g	g
hh	hv
ih	i
iy	i:
jh	J
k	k
l	l
m	m
n	nB
nx	nB
ng	nB

English phone	Hindi phone
ow	o
oy	o j
p	p
r	9r
s	s
sh	s
t	tr
th	tBh
uh	u
uw	u:
v	v
w	v
y	j
z	s
zh	sh

BIBLIOGRAPHY

[1] S. Sitaram, A. Parlikar, G. K. Anumanchipalli, and A. W. Black, "Universal grapheme-based speech synthesis," in *Interspeech*, 2015.

[2] O. S. Watts, *Unsupervised learning for text-to-speech synthesis*. PhD thesis, 2013.

[3] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *ICASSP*, vol. 1, pp. 373–376, IEEE, 1996.

[4] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1039–1064, 2009.

[5] G. K. Anumanchipalli, *Intra-lingual and cross-lingual prosody modelling*. PhD thesis, Carnegie Mellon University, 2013.

[6] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards, "Normalization of non-standard words," *Computer Speech & Language*, vol. 15, no. 3, pp. 287–333, 2001.

[7] A. Parlikar, *Style-specific phrasing in speech synthesis*. PhD thesis, Carnegie Mellon University, 2013.

[8] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *ICASSP*, vol. 3, pp. 1315–1318, IEEE, 2000.

[9] P. Taylor, *Text-to-speech synthesis*. Cambridge university press, 2009.

[10] R. Weide, "The CMU pronunciation dictionary, release 0.6," 1998.

[11] M. Cernak and M. Rusko, "An evaluation of synthetic speech using the PESQ measure," in *Proceedings of the European Congress on Acoustics*, pp. 2725–2728, 2005.

[12] S. Moller and T. H. Falk, "Quality prediction for synthesized speech: comparison of approaches," in *International Conference on Acoustics*, 2009.

[13] M. Chu and H. Peng, "An objective measure for estimating MOS of synthesized speech," in *Interspeech*, pp. 2087–2090, 2001.

[14] Y. Hu and P. C. Loizou, "Evaluation of objective quality measures for speech enhancement," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 1, pp. 229–238, 2008.

[15] C. Valentini-Botinhao, J. Yamagishi, and S. King, "Can objective measures predict the intelligibility of modified HMM-based synthetic speech in noise?," in *Interspeech*, pp. 1837–1840, 2011.

[16] R. Ullmann, R. Rasipuram, and H. Bourlard, "Objective intelligibility assessment of text-to-speech systems through utterance verification," tech. rep., 2015.

[17] J. Kominek, T. Schultz, and A. W. Black, "Synthesizer voice quality of new languages calibrated with mean Mel Cepstral Distortion," in *SLTU*, pp. 63–68, 2008.

[18] A. Parlikar and A. W. Black, "Modeling pause-duration for style-specific speech synthesis," *Interspeech*, vol. 56, no. 3.8, p. 130, 2012.

[19] J. Xu, C. Guan, and H. Li, "An objective measure for assessment of a corpus-based text-to-speech system," in *Proceedings of IEEE Workshop on Speech Synthesis*, pp. 179–182, IEEE, 2002.

[20] J. Tian, J. Nurminen, and I. Kiss, "Modular text-to-speech synthesis evaluation for Mandarin Chinese," *Proceedings of ISCSLP*, 2006.

[21] D. Hirst, A. Rilliard, and V. Aubergé, "Comparison of subjective evaluation and an objective evaluation metric for prosody in text-to-speech synthesis," in *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis*, 1998.

[22] R. A. Clark, M. Podsiadlo, M. Fraser, C. Mayo, and S. King, "Statistical analysis of the Blizzard Challenge 2007 listening test results," *In proceedings of SSW6*, 2007.

[23] R. G. Gordon and B. F. Grimes, *Ethnologue: languages of the world*, vol. 15. Sil international Dallas, TX, 2005.

[24] A. W. Black and A. F. Llitjos, "Unit selection without a phoneme set," in *Proceedings of 2002 IEEE Workshop on Speech Synthesis*, pp. 207–210, IEEE, 2002.

[25] A. W. Black and K. Lenzo, "Building voices in the Festival speech synthesis system," tech. rep., 2002.

[26] M. Killer, S. Stüker, and T. Schultz, "Grapheme based speech recognition," in *INTERSPEECH*, 2003.

[27] C. Unicode Staff, *The Unicode standard: worldwide character encoding*. Addison-Wesley Longman Publishing Co., Inc., 1991.

[28] T. Qian, K. Hollingshead, S.-y. Yoon, K.-y. Kim, R. Sproat, and M. LREC, "A Python toolkit for universal transliteration.," in *LREC*, 2010.

[29] J. L. Hieronymus, "ASCII phonetic symbols for the world's languages: Worldbet," *Journal of the International Phonetic Association*, vol. 23, 1993.

[30] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*. " O'Reilly Media, Inc.", 2009.

[31] A. W. Black and P. Taylor, "The Festival speech synthesis system: system documentation," tech. rep., Human Communication Research Centre, University of Edinburgh, January 1997.

[32] A. W. Black, "CLUSTERGEN: a statistical parametric synthesizer using trajectory modeling," in *Interspeech*, 2006.

[33] J. Kominek and A. W. Black, "CMU Arctic databases for speech synthesis," in *Proceedings of the 5th Speech Synthesis Workshop*, (Pittsburgh, Pennsylvania), pp. 223–224, June 2004.

[34] K. Prahallad, E. N. Kumar, V. Keri, S. Rajendran, and A. W. Black, "The IIIT-H Indic speech databases," in *Interspeech*, (Portland, OR, USA), September 2012.

[35] T. Schultz, A. W. Black, S. Badaskar, M. Hornyak, and J. Kominek, "SPICE: web-based tools for rapid language adaptation in speech processing systems," in *Interspeech*, pp. 2125–2128, ISCA, 2007.

[36] M. Mashimo, T. Toda, K. Shikano, and N. Campbell, "Evaluation of cross-language voice conversion based on GMM and STRAIGHT," 2001.

[37] J. Kominek, *TTS from zero: building synthetic voices for new languages*. PhD thesis, Carnegie Mellon University, 2009.

[38] L. Breiman, "Random forests," *Machine Learning*, vol. 45(1), pp. 5–32, 2001.

[39] A. W. Black and P. K. Muthukumar, "Random forests for statistical speech synthesis," in *Interspeech*, 2015.

[40] A. Parlikar, "TestVox: web-based framework for subjective evaluation of speech synthesis," *Opensource Software*, 2012.

[41] B. Narasimhan, R. Sproat, and G. Kiraz, "Schwa-deletion in Hindi text-to-speech synthesis," *International Journal of Speech Technology*, vol. 7, no. 4, pp. 319–333, 2004.

[42] T. Naim R and I. Nagar, "Prosodic rules for schwa-deletion in Hindi text-to-speech synthesis," *International Journal of Speech Technology*, vol. 12, no. 1, pp. 15–25, 2009.

[43] M. Choudhury, A. Basu, and S. Sarkar, "A diachronic approach for schwa deletion in Indo-Aryan languages," in *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, SIGMorPhon '04, (Stroudsburg, PA, USA), pp. 20–26, Association for Computational Linguistics, 2004.

[44] K. Prahallad, A. Vadapalli, S. Kesiraju, H. A. Murthy, S. Lata, T. Nagarajan, M. Prasanna, H. Patil, A. K. Sao, S. King, A. W. Black, and K. Tokuda, "The Blizzard Challenge 2014," 2014.

[45] P. Placeway, S. Chen, M. Eskenazi, U. Jain, V. Parikh, B. Raj, M. Ravishankar, R. Rosenfeld, K. Seymore, M. Siegler, *et al.*, "The 1996 hub-4 Sphinx-3 system," in *Proceedings of the DARPA speech recognition workshop*, pp. 85–89, 1997.

[46] P. K. Muthukumar and A. W. Black, "Automatic discovery of a phonetic inventory for unwritten languages for statistical speech synthesis," in *ICASSP*, pp. 2594–2598, IEEE, 2014.

[47] F. Ahmed and A. Nurnberger, "Arabic/English word translation disambiguation using parallel corpora and matching schemes," in *Proceedings of EAMT*, vol. 8, p. 28, 2008.

[48] C. L. Bennett and A. W. Black, "Using acoustic models to choose pronunciation variations for synthetic voices," in *Interspeech*, 2003.

[49] C. L. Bennett and A. W. Black, "Prediction of pronunciation variations for speech synthesis: a data-driven approach," in *ICASSP*, pp. 297–300, 2005.

[50] D. Yarowsky, "Homograph disambiguation in text-to-speech synthesis," in *Progress in speech synthesis*, pp. 157–172, Springer, 1997.

[51] M. Davel and E. Barnard, "Extracting pronunciation rules for phonemic variants," *ISCA technical and research workshop*, 2006.

[52] F. Qiao, J. Sherwani, and R. Rosenfeld, "Small-vocabulary speech recognition for resource-scarce languages," in *Proceedings of the First ACM Symposium on Computing for Development*, p. 3, ACM, 2010.

[53] H. Y. Chan and R. Rosenfeld, "Discriminative pronunciation learning for speech recognition for resource scarce languages," in *Proceedings of the 2nd ACM Symposium on Computing for Development*, p. 12, ACM, 2012.

[54] G. K. Anumanchipalli, M. Ravishankar, and R. Reddy, "Improving pronunciation inference using n-best list, acoustics and orthography," in *ICASSP*, vol. 4, pp. IV–925, IEEE, 2007.

[55] K. Prahallad, A. W. Black, and R. Mosur, "Sub-phonetic modeling for capturing pronunciation variations for conversational speech synthesis," in *ICASSP*, vol. 1, (Toulouse, France), pp. 853–856, May 2006.

[56] T. Hain, "Implicit modelling of pronunciation variation in automatic speech recognition," *Speech Communication*, vol. 46, no. 2, pp. 171–188, 2005.

[57] I. Almosallam, A. AlKhalifa, M. Alghamdi, M. Alkanhal, and A. Alkhairy, "SASSC: A standard Arabic single speaker corpus," in *Proceedings of 8th ISCA Speech Synthesis Workshop*, 2013.

[58] R. Hsiao, A. Venugopal, T. Köhler, Y. Zhang, P. Charoenpornsawat, A. Zollmann, S. Vogel, A. W. Black, T. Schultz, and A. Waibel, "Optimizing components for handheld two-way speech translation for an English-Iraqi Arabic system.," in *Interspeech*, 2006.

[59] D. Graff, T. Buckwalter, H. Jin, and M. Maamouri, "Lexicon development for varieties of spoken colloquial Arabic," in *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, pp. 999–1004, 2006.

[60] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pp. 173–180, Association for Computational Linguistics, 2003.

[61] N. Habash, R. Roth, O. Rambow, R. Eskander, and N. Tomeh, "Morphological analysis and disambiguation for dialectal Arabic," in *HLT-NAACL*, pp. 426–432, 2013.

[62] S. Sitaram, S. Palkar, Y.-N. Chen, A. Parlikar, and A. W. Black, "Bootstrapping text-to-speech for speech processing in languages without an orthography," in *ICASSP*, pp. 7992–7996, IEEE, 2013.

[63] P. Taylor, R. Caley, A. W. Black, and S. King, "Edinburgh speech tools library," *System Documentation Edition*, vol. 1, 1999.

[64] A. Clark, "Combining distributional and morphological information for part of speech induction," in *Proceedings of the 10th conference on European chapter of the Association for Computational*

Linguistics, pp. 59–66, Association for Computational Linguistics, 2003.

[65] J. Kominek and A. W. Black, "Impact of durational outlier removal from unit selection catalogs," in *Fifth ISCA Workshop on Speech Synthesis*, 2004.

[66] M. A. Peabody, *Methods for pronunciation assessment in computer aided language learning*. PhD thesis, Massachusetts Institute of Technology, 2011.

[67] M. Eskenazi and S. Hansma, "The Fluency pronunciation trainer," *Proceedings of Speech Technology in Language Learning*, pp. 77–80, 1998.

[68] M. Eskenazi and G. Pelton, "Pinpointing pronunciation errors in children's speech: examining the role of the speech recognizer," in *ISCA Tutorial and Research Workshop (ITRW) on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology*, 2002.

[69] S. Picard, G. Ananthakrishnan, P. Wik, O. Engwall, and S. Abdou, "Detection of specific mispronunciations using audiovisual features.," in *AVSP*, pp. 7–2, 2010.

[70] G. Ananthakrishnan, P. Wik, and O. Engwall, "Detecting confusable phoneme pairs for Swedish language learners depending on their first language," *TMH-QPSR*, vol. 51, no. 1, pp. 89–92, 2011.

[71] G. Ananthakrishnan, P. Wik, O. Engwall, and S. Abdou, "Using an ensemble of classifiers for mispronunciation feedback," in *SLaTE*, pp. 49–52, 2011.

[72] S. Wei, G. Hu, Y. Hu, and R.-H. Wang, "A new method for mispronunciation detection using support vector machine based on pronunciation space models," *Speech Communication*, vol. 51, no. 10, pp. 896–905, 2009.

[73] S. Witt and S. Young, "Performance measures for phone-level pronunciation teaching in CALL," in *Proc. of the Workshop on Speech Technology in Language Learning*, pp. 99–102, 1998.

[74] J.-C. Chen, J.-S. Jang, J.-Y. Li, and M.-C. Wu, "Automatic pronunciation assessment for Mandarin Chinese," in *ICME*, vol. 3, pp. 1979–1982, IEEE, 2004.

[75] N. Minematsu, "Yet another acoustic representation of speech sounds," in *ICASSP*, vol. 1, pp. I–585, IEEE, 2004.

[76] H. Hussein, H. Mixdorff, H. S. Do, S. Wei, S. Gong, H. Ding, Q. Gao, and G. Hu, "Towards a computer-aided pronunciation training system for German learners of Mandarin - prosodic analysis," *L2WS-2010, Tokyo*, 2010.

[77] L. Deng and D. X. Sun, "A statistical approach to automatic speech recognition using the atomic speech units constructed from overlapping articulatory features," *The Journal of the Acoustical Society of America*, vol. 95, no. 5, pp. 2702–2719, 1994.

[78] K. Kirchhoff, G. A. Fink, and G. Sagerer, "Combining acoustic and articulatory feature information for robust speech recognition," *Speech Communication*, vol. 37, no. 3, pp. 303–319, 2002.

[79] F. Metze, *Articulatory features for conversational speech recognition*. PhD thesis, Karlsruhe, Univ., Diss., 2005, 2005.

[80] S. Stüker, T. Schultz, F. Metze, and A. Waibel, "Multilingual articulatory features," in *ICASSP*, vol. 1, pp. I–144, IEEE, 2003.

[81] S. Stüker, F. Metze, T. Schultz, and A. Waibel, "Integrating multilingual articulatory features into speech recognition.," in *Interspeech*, 2003.

[82] T. Schultz and A. Waibel, "Language-independent and language-adaptive acoustic modeling for speech recognition," *Speech Communication*, vol. 35, no. 1, pp. 31–51, 2001.

[83] T. Schultz, "Globalphone: a multilingual speech and text database developed at Karlsruhe university.," in *Interspeech*, 2002.

[84] A. W. Black, H. T. Bunnell, Y. Dou, P. K. Muthukumar, F. Metze, D. Perry, T. Polzehl, K. Prahallad, S. Steidl, and C. Vaughn, "Articulatory features for expressive speech synthesis," in *ICASSP*, pp. 4005–4008, 2012.

[85] P. Baljekar, S. Sitaram, P. K. Muthukumar, and A. W. Black, "Using articulatory features and inferred phonological segments in zero resource speech processing," in *Interspeech*, 2015.

[86] A. Wrench, "The MOCHA-TIMIT articulatory database," 1999.

[87] C. Myers-Scotton, *Duelling languages: grammatical structure in codeswitching*. Oxford University Press, 1997.

[88] K. Bali, J. Sharma, M. Choudhury, and Y. Vyas, ""i am borrowing ya mixing?" an analysis of English-Hindi code mixing in Facebook," *EMNLP 2014*, p. 116, 2014.

[89] T. Solorio, E. Blair, S. Maharjan, S. Bethard, M. Diab, M. Gohneim, A. Hawwari, F. AlGhamdi, J. Hirschberg, A. Chang, *et al.*, "Overview for the first shared task on language identification in code-switched data," *EMNLP*, p. 62, 2014.

[90] G. Chittaranjan, Y. Vyas, and K. B. M. Choudhury, "Word-level language identification using crf: code-switching shared task report of MSR India system," *EMNLP 2014*, p. 73, 2014.

[91] Y. Vyas, S. Gella, J. Sharma, K. Bali, and M. Choudhury, "Pos tagging of English-Hindi code-mixed social media content," *Proceedings of the First Workshop on Codeswitching, EMNLP*, 2014.

[92] P. Gupta, K. Bali, R. E. Banchs, M. Choudhury, and P. Rosso, "Query expansion for mixed-script information retrieval," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pp. 677–686, ACM, 2014.

[93] R. Cotterell, A. Renduchintala, N. Saphra, and C. Callison-Burch, "An Algerian Arabic-French code-switched corpus," in *Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools Workshop Programme*, p. 34, 2014.

[94] T. I. Modipa, M. H. Davel, and F. De Wet, "Implications of Sepedi/English code switching for ASR systems," 2013.

[95] N. T. Vu, D.-C. Lyu, J. Weiner, D. Telaar, T. Schlippe, F. Blaicher, E.-S. Chng, T. Schultz, and H. Li, "A first speech recognition system for Mandarin-English code-switch conversational speech," in *ICASSP*, pp. 4889–4892, IEEE, 2012.

[96] D.-C. Lyu, T.-P. Tan, E.-S. Chng, and H. Li, "Mandarin-English code-switching speech corpus in South-East Asia: SEAME," *Language Resources and Evaluation*, pp. 1–20, 2015.

[97] B. H. Ahmed and T.-P. Tan, "Automatic speech recognition of code switching speech using 1-best rescoring," in *Asian Language Processing (IALP), 2012 International Conference on*, pp. 137–140, IEEE, 2012.

[98] H. Liang, Y. Qian, and F. K. Soong, "An HMM-based bilingual (Mandarin-English) TTS," *Proceedings of SSW6*, 2007.

[99] M. Chu, H. Peng, Y. Zhao, Z. Niu, and E. Chang, "Microsoft Mulan-a bilingual TTS system," in *ICASSP*, vol. 1, pp. I–264, IEEE, 2003.

[100] A. W. Black and K. Tokuda, "The Blizzard Challenge 2005: Evaluating corpus-based speech synthesis on common datasets," in *Interspeech*, 2005.

[101] R. Russell and M. Odell, "Soundex," *US Patent*, vol. 1, 1918.

[102] R. S. Roy, M. Choudhury, P. Majumder, and K. Agarwal, "Overview and datasets of fire 2013 track on transliterated search," in *Fifth Forum for Information Retrieval Evaluation*, 2013.

[103] A. W. Black, K. Lenzo, and V. Pagel, "Issues in building general letter to sound rules," 1998.

[104] J. Nerbonne, W. Heeringa, E. Van den Hout, P. Van der Kooi, S. Otten, W. Van de Vis, *et al.*, "Phonetic distance between Dutch dialects," in *CLIN VI: proceedings of the sixth CLIN meeting*, pp. 185–202, 1996.

[105] J. Nerbonne and W. Heeringa, "Measuring dialect distance phonetically," in *Proceedings of the Third Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON-97)*, 1997.

[106] S. Sriram, P. Talukdar, S. Badaskar, K. Bali, and A. Ramakrishnan, "Phonetic distance based crosslingual search," in *Proceedings of the International Conference on Natural Language Processing*, 2004.

[107] V. B. Le, L. Besacier, and T. Schultz, "Acoustic-phonetic unit similarities for context dependent acoustic model portability," in *ICASSP*, vol. 1, pp. I–I, IEEE, 2006.

[108] L. Melnar and C. Liu, "A combined phonetic-phonological approach to estimating cross-language phoneme similarity in an ASR environment," in *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology*, pp. 1–10, Association for Computational Linguistics, 2006.

[109] M. Pucher, A. Türk, J. Ajmera, and N. Fecher, "Phonetic distance measures for speech recognition vocabulary and grammar optimization," in *3rd Congress of the Alps Adria Acoustics Association*, pp. 2–5, 2007.

[110] O. Watts, Z. Wu, and S. King, "Sentence-level control vectors for deep neural network speech synthesis," in *Interspeech*, 2015.

[111] B. Zhou, L. Besacier, and Y. Gao, "On efficient coupling of ASR and SMT for speech translation," in *ICASSP*, vol. 4, pp. IV–101, IEEE, 2007.

[112] N. Bertoldi, R. Zens, and M. Federico, "Speech translation by confusion network decoding," in *ICASSP*, vol. 4, pp. IV–1297, IEEE, 2007.

[113] P. D. Agüero, J. Adell, and A. Bonafonte, "Prosody generation for speech-to-speech translation," in *ICASSP*, vol. 1, pp. I–I, IEEE, 2006.

[114] V. K. R. Sridhar, S. Bangalore, and S. S. Narayanan, "Factored translation models for enriching spoken language translation with prosody," in *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[115] L. Besacier, B. Zhou, and Y. Gao, "Towards speech translation of non written languages," in *Spoken Language Technology Workshop, 2006. IEEE*, pp. 222–225, IEEE, 2006.

[116] S. Stüker and A. Waibel, "Towards human translations guided language discovery for ASR systems.," in *SLTU*, pp. 76–79, 2008.

[117] M. Bacchiani, M. Ostendorf, Y. Sagisaka, and K. Paliwal, "Design of a speech recognition system based on acoustically derived segmental units," in *ICASSP*, vol. 1, pp. 443–446, IEEE, 1996.

[118] F. Stahlberg, "Towards automatic speech recognition for nonwritten languages using translations from other languages,"

[119] W. Hartmann, A. Roy, L. Lamel, and J.-L. Gauvain, "Acoustic unit discovery and pronunciation generation from a grapheme-based lexicon," in *ASRU*, pp. 380–385, IEEE, 2013.

[120] Z. Ahmed, J. Jiang, J. Carson-Berndsen, P. Cahill, and A. Way, "Hierarchical phrase-based MT for phonetic representation-based speech translation," in *Proceedings of the tenth Biennial Conference of the Association for Machine Translation in the Americas (AMTA), San Diego, CA*, 2012.

[121] F. Stahlberg, T. Schlippe, S. Vogel, and T. Schultz, "Word segmentation through cross-lingual word-to-phoneme alignment," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pp. 85–90, IEEE, 2012.

[122] G. Zavaliagkos and T. Colthurst, "Utilizing untranscribed training data to improve performance," in *DARPA Broadcast News Transcription and Understanding Workshop*, pp. 301–305, 1998.

[123] M. Elsner, S. Goldwater, and J. Eisenstein, "Bootstrapping a unified model of lexical and phonetic acquisition," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pp. 184–193, Association for Computational Linguistics, 2012.

[124] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT summit*, vol. 5, pp. 79–86, 2005.

[125] A. W. Black, "CLUSTERGEN: a statistical parametric synthesizer using trajectory modeling," in *Interspeech*, 2006.

[126] P. Taylor, A. W. Black, and R. Caley, "The architecture of the Festival speech synthesis system," 1998.

[127] E. O. Selkirk, "English compounding and the theory of word structure," *The scope of lexical rules*, pp. 229–277, 1981.

[128] T. Kudoh, "Crf++," *Software, http://crfpp. sourceforge. net*, 2007.

[129] G. K. Anumanchipalli, *Intra-lingual and cross-lingual prosody modelling*. PhD thesis, Carnegie Mellon University, 2013.

[130] P. Taylor, "Analysis and synthesis of intonation using the tilt model," *The Journal of the acoustical society of America*, vol. 107, no. 3, pp. 1697–1714, 2000.

[131] L. J. Martin, A. Wilkinson, S. S. Miryala, V. Robison, and A. W. Black, "Utterance classification in speech to speech translation for zero-resource languages in the hospital administration domain," 2015.

[132] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech*, pp. 1045–1048, 2010.

[133] T. Merritt, J. Yamagishi, Z. Wu, O. Watts, and S. King, "Deep neural network context embeddings for model selection in rich-context HMM synthesis," in *Interspeech*, 2015.

[134] Y. Tsvetkov and C. Dyer, "Cross-lingual bridges with models of lexical borrowing," *Journal for Artificial Intelligence Research*, 2015.