Pronunciation Modeling for Children's Speech Recognition

Benjamin Levin

Master of Science

Computer Science

School of Informatics

University of Edinburgh

2019

Abstract

Automatic speech recognition for adult speakers has reached a widely visible level of maturity. Performance on children's speech, however, lags behind, often due to a lack of publicly available training data and high speaker variance. This work examines a well-known, difficult children's speech corpus, PF-Star, and shows how modern neural network architectures can achieve a state-of-the-art result on PF-Star without the use of a biased language model or out-of-domain acoustic data. PF-Star contains children with a range of different ages and UK dialects. Motivated by this, a data-driven method of pronunciation modeling is investigated under the hypothesis that it will be even more effective on such a high-variance dataset than it has been in previous works. It is shown to improve accuracy by a small yet statistically significant amount, corroborating previous work, but rejecting this hypothesis.

Acknowledgements

Thank you to Steve Renals, Joachim Fainberg, Ondrej Klejch, and Nick McKenna. Without their advice, this would have been impossible. Or at least taken more than ten weeks.

Table of Contents

1 Introduction **1**

1.1 Speech Recognition 1

1.2 Motivation 1

1.3 Research Hypothesis 2

1.4 Work Undertaken 2

2 Background **3**

2.1 Brief History of Speech Recognition 3

2.1.1 Early Efforts 3

2.1.2 Hidden Markov Models 3

2.1.3 MMI 5

2.1.4 WFSTs 6

2.1.5 Deep Learning 7

2.1.6 TDNNs 8

2.2 Children's Speech 10

2.2.1 Analysis 10

2.2.2 Previous Work 11

2.3 Pronunciation Modeling 12

2.3.1 Implicit vs. Explicit Models 12

3 Methods **14**

3.1 Kaldi Toolkit 14

3.2 The PF-Star Dataset 14

3.3 Model Design and Baseline 16

3.3.1 Data Preparation 16

3.3.2 Features 17

3.3.3 Alignment 17

3.3.4 Lattice Free-MMI 18
3.3.5 TDNN-F 19
3.3.6 Acoustic Model 20
3.3.7 Pronunciation Modeling Experiment 21
3.4 Results 22
3.4.1 Baseline System 22
3.4.2 Accuracy 23
3.4.3 Pronunciation Model 23
3.4.4 Secondary Experiment: Segmentation 23

4 Discussion **25**
4.1 Analysis 25
4.1.1 Baseline Model 25
4.1.2 Interpretation of Pronunciation Modeling Experiment 26

5 Conclusions **28**
5.1 Research Hypothesis 28
5.2 Limitations of This Research 28
5.2.1 Data Normalization 28
5.2.2 Pronunciation Modeling Implementation 29
5.3 Future Directions 30
5.4 Final Words 31

Bibliography **32**

Chapter 1

Introduction

1.1 Speech Recognition

Automatic speech recognition (ASR) is one of a glut of technologies in the past decade that have broken through from academic curiosity and niche private application to global commercial exploitation and consumer familiarity. ASR systems can now be found everywhere, from customer service hotlines to educational software. This growth has been prompted by a sharp increase in the availability of performant hardware and robust neural network architectures. These have allowed researchers to leverage larger and larger datasets to train highly accurate speech recognizers.

1.2 Motivation

This large improvement in the viability of ASR systems comes courtesy of large datasets of adult speech, frequently in excess of 500 hours of transcribed audio. The Librispeech dataset, a well-known modern corpus for training recognizers, contains over 900 hours of speech data. Datasets of this size of children's speech are simply not available for academics. Most academically available children's speech corpora are 8 to 15 hours. Large corporations with public-facing data collection infrastructure may be able to assemble datasets of young consumers' speech with millions of utterances [27]. But for those outside of Google and its peers, these datasets are inaccessible. Classroom logistics, legal issues, and the extra labor necessary make formal data collection processes more difficult and expensive with children than adults, so they simply are not produced.

This situation of data scarcity is very familiar along the frontier of ASR progress.

Recognizing standard dialect adult speech of a single speaker in a quiet room with a good microphone is widely considered a "solved" problem. It is a probabilistic venture, so "solved" may not be the correct word, but the software recognizers perform very similarly in accuracy to humans listening to the same audio. What is not "solved" are the non-standard dialects, low-resource languages, noisy backgrounds, low-quality hardware, overlapping speakers, and children. These domains have severe data scarcity issues and routinely exhibit high variance between speakers and individual data collection sessions.

1.3 Research Hypothesis

This paper will address children's speech and will focus on the use of pronunciation modeling (see Sec. 2.3) to improve the accuracy of a recognizer focusing on children's speech. My hypothesis is that given such small amounts of widely varying data, the meaningful injection of human knowledge that explicit pronunciation modeling represents can help an ASR system to better define its individual phones at train-time and better accommodate children's often misarticulated speech at decode time.

1.4 Work Undertaken

The rest of this document will lay out the work undertaken. First, I will explore some of the previous research in general ASR before narrowing in on the problems inherent to children's speech recognition, and some previous work done around my proposed method, explicit pronunciation modeling. Then, I will outline the specific experiments performed: the tools, datasets and acoustic modeling techniques used to construct the baseline system, as well as the implementation of the pronunciation model itself. Afterwards, I will discuss the results of these experiments. Ultimately, I found that the pronunciation model helped improve accuracy, but not by a large amount, suggesting that my core hypothesis about pronunciation modeling being particularly applicable to children's speech is not realistic. Finally, I will describe what my results mean in the context of previous work and mention some future directions that similar research should consider.

Chapter 2

Background

2.1 Brief History of Speech Recognition

2.1.1 Early Efforts

ASR has its earliest roots in toy projects like Bell Labs' 'Audrey' 0-9 digit recognizer from 1952 [36] or IBM's Shoebox device from 1962, which had a vocabulary of 16 words [1]. The following decade and a half had many interesting and creative approaches to speech recognition coming from various researchers, but only a handful are truly relevant to the specifics discussed in this paper. One rightfully well-known example was Vintsyuk's 1968 paper on dynamic programming for speech recognition, which pioneered a technique strongly resembling modern Viterbi alignment [49].

The "Harpy" project at Carnegie-Mellon University in the mid-70s was a notable leap in progress [29]. It could recognize continuous speech with a vocabulary of over 1,000 words. Importantly, the system was aimed at speech *understanding*, not just recognition, so it had a system enforcing certain lexical and syntactical constraints. It also framed both its word recognition and semantic analysis tasks as optimal path searches through a network. Together, these approaches could be seen as a prelude to modern ASR techniques using weighted finite-state transducers (WFSTs) and decoding probabilistically with language models (LMs).

2.1.2 Hidden Markov Models

Around the same time, the mid-70s, came Jim and Janet Baker's "DRAGON" system [6]. Thoroughly ahead of its time, it pioneered the usage of Hidden Markov processes as a model of speech. It utilized a hierarchical framework of Hidden Markov Models

(HMMs) for phone recognition, lexicon modeling, and a joint syntactical/semantical grammar for word prior probabilities.

HMMs allow for generative modeling of speech that accommodates the ever-changing time scale of a speech signal by viewing it as a series of probabilistic transitions through hidden states that produce observations (see Fig. 2.1). In the simplest case, each state transition generates observations from a single Gaussian, though more complicated models can be used as well. By convention, HMMs are represented by the parameters (A, B, π) where A is a transition matrix specifying the probabilities of moving from one state to any other, B is an observation matrix specifying the distribution of witnessed outputs given a state, and π is the array of probabilities of starting in any particular state.

Since HMMs are a generative model, they cannot evaluate acoustic input directly at inference time. A brute force search across all possible underlying phone sequences could find the state sequence that maximizes the likelihood of the acoustic observations, but this is computationally intractable. Instead, the Viterbi algorithm is used, which leverages dynamic programming to build the optimal path [50]. At training time, the Baum-Welch algorithm (a special case of the forward-backward algorithm) is used to iteratively derive a local optimum for the parameters (A, B, π) that maximize the likelihood of the training examples [8].

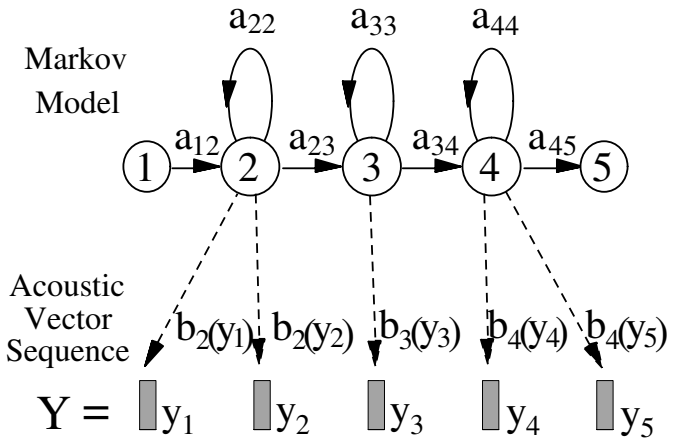


Figure 2.1: Example of an HMM phone model, from [17]

HMMs are necessarily a flawed model. The Markov assumption (that the history of states is irrelevant and only the current state influences output) does not hold for human speech, and their generative nature means training to an optimum that is contingent on the model being correct, which it is not. Nevertheless, they have proven to be a highly

effective and resilient tool, still used in many state-of-the-art systems [17].

2.1.3 MMI

1986 brought an important step in IBM's early experiments with using Maximum Mutual Information (MMI) as a training objective for HMMs [5]. Traditional maximum likelihood training updates the HMM parameters to maximize the probability that model would generate the acoustic training data given the textual (or phonetic) training data. It doesn't necessarily maximize the probability that the phonetic transcript is the optimum path through the states given that sequence of acoustic observations, which is the metric actually sought during decoding. Using MMI to train on ameliorates this. The HMM remains a generative model, but MMI allows it to be trained on a objective that better discriminates the training data. MMI is a powerful objective function to utilize even today, though for more complicated models it can become computationally difficult (see sec. 3.3.4).

The BBN Byblos system published in 1989 was another notable advancement, introducing the notion of context-dependent phones to model coarticulation effects [45]. The BBN approach included methods for backoff and interpolation of phone models with progressively less context-dependency to mitigate training data sparsity of contexts. This would later be superseded by tied-state models with decision trees, but the method was clever enough to establish a state-of-the-art for continuous speech recognition at the time [17].

The 1990s brought a great deal of practical progress to speech processing. HMMs became substantially more widespread, and using Gaussian Mixture Models (GMMs) to allow for more nuanced output distributions on HMM states was investigated [17]. This led to the proliferation of more robust, speaker-independent, large-vocabulary ASR systems like Carnegie Mellon's SPHINX [25]. The SPHINX system is particularly relevant to this paper due to the relatively early use of explicit pronunciation modeling. Other systems of similar robustness were built, though. By 1990, AT&T had already deployed automated, operator-free collect calling and service management, and they were not the only telecommunications corporation to do so [48].

One of the most important shifts the 90s brought was increased interest in collecting large, annotated speech corpora for training the increasingly sophisticated models being developed. Significant amounts of DARPA funding spurred the creation of the famous TIMIT corpus [19] and the spoken Wall Street Journal (WSJ) corpus [38], as

well as the Switchboard corpus [20], which has garnered the most interest in recent years. All of these corpora have been used to develop and evaluate hundreds, if not thousands, of speech processing systems over the years.

2.1.4 WFSTs

In the mid 1990s, Mohri et al. published a slew of papers that collectively established the practice of using weighted finite state transducers (WFSTs)for speech recognition [33][32]. They outlined the fundamental transducer algorithms necessary to assemble a system and justified their applicability. This allowed many of the advancements of the previous decades (context-dependent phone recognition, pronunciation dictionaries, HMMs, n-gram language models) to be encoded in one unified computational object.

Under their proposed framework, each of the aforementioned models is encoded as a WFST, then composed into one transducer [34]. A simple algorithm can translate an n-gram language model into a WFST (named G by convention) by encoding contexts as states, words on both the input and output of transitions, and likelihoods as weights on transitions as in figure 2.2. The pronunciation lexicon can be encoded with phones as inputs, words as outputs, and relative likelihoods of pronunciations as weights, as in figure 2.3, named L by convention. A context-dependency handling transducer C can be constructed to have base phones on its inputs with context-dependent triphones on the outputs, as in figure 2.4. Lastly, a trained HMM-based acoustic model, whether pure HMM or HMM-DNN hybrid, can be readily encoded as a WFST H with acoustic observations on inputs and context-dependent phones on outputs. This is fairly straightforward since the HMM is already a finite state machine, as depicted in figure 2.1. To build the final graph, the C transducer must first be inverted (in order to transduce the context-dependent triphones from the HMM into base phones for the lexicon FST) and the H C and L transducers need to have disambiguation symbols added to handle homophones. These are now denoted by $\tilde{H}$, $\tilde{C}$, and $\tilde{L}$. Then, the transducers are iteratively composed, determinized, and minimized, with a final operation π_ε replacing disambiguation symbols with the empty string ε: $\pi_\varepsilon(\min(\det(\tilde{H} \circ \det(\tilde{C} \circ \det(\tilde{L} \circ G)))))$. This can involve a large amount of shuffling or pruning states, pushing and recomputing weights, and rewriting disambiguation symbols, but once it is finished, the resulting finite-state machine represents knowledge at all levels of the linguistic hierarchy, whilst having a number of states less than double its largest constituent (usually G) [34]. Mohri et al. proved that this guarantees the

existence of an unambiguous optimal path through the transducer, and has extremely favorable efficiency at decode time, allowing more sophisticated and memory-hungry models to be built.

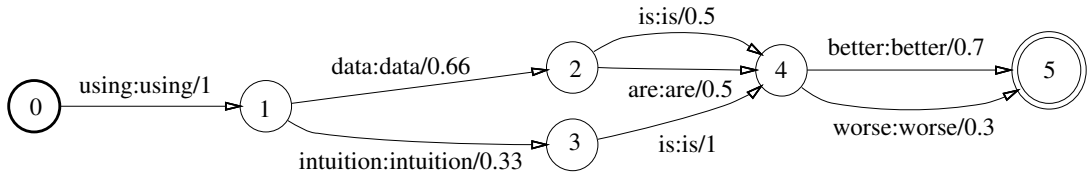


Figure 2.2: Partial n-gram LM encoded as WFST, from [34]

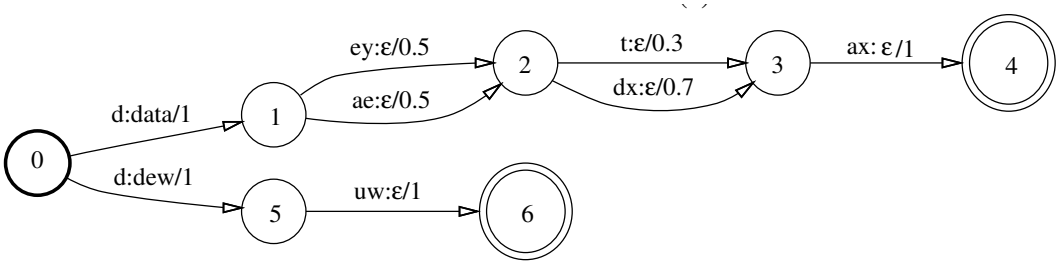


Figure 2.3: Lexicon of the words “data” and “dew” encoded as WFST, from [34]

Figure 2.4: Context dependency WFST of triphones, non-deterministic form and deterministic form, respectively, from [34]

2.1.5 Deep Learning

A key story in the advancement of any modern machine learning task is that of deep neural networks (DNNs). There were early examples of using perceptrons in the 60s, but these floundered when it was shown they could not model many types of functions [30]. Deeper networks were experimented with in the 70s to some success, but remained more a concept experiment for cognitive scientists than a tool for machine learning engineers due to a lack of computing power and practices for training [52]. The broader applicability of backpropagation was realized in the 80s and sparked another wave of interest in neural nets [28]. Yann LeCun’s well-known experiment pioneering convolutional neural networks (CNNs) in 1989 was a notable step forward for

the approach [24]. This experiment took the better part of a week to run, however, and neural networks still remained a niche tool until more performant hardware became available in the 1990s.

The biggest growth in the application of DNNs to speech came after Morgan and Bourlard showed how using a feedforward neural net with a context window of frames as its input and HMM state probabilities as its outputs could improve on GMMs while retaining the effectiveness of HMMs [35]. They argued that the shallow modeling and structural assumptions of GMMs had became too onerous. GMMs cannot handle correlated features without computationally expensive full covariance matrices, and they cannot be trained discriminatively. The same team would go on to achieve multiple state-of-the-art results in the early 90s with their HMM-DNN hybrid method, and the fundamental concept remains popular today [11].

When DNNs for speech really took off was the mid-2000s, when Hinton et al. repopularized them in the form of deep belief networks with Boltzmann pretraining [31]. These were much more effective at modeling speech patterns than previous deep learning efforts. This represented the beginning of an explosion of interest in deep learning approaches to speech recognition. Many techniques that had been invented several decades prior were yanked out of obscurity in the few years following. Recurrent neural networks (RNNs) quickly became popular in ASR with the introduction of the long short-term memory (LSTM) [15]. Connectionist temporal classification (CTC) also became popular around this time, using large RNNs to generate transcripts directly, skipping the intermediate steps required by hybrid HMM systems [21].

Another significant advancement was the introduction of GPU computation to training DNNs by Raina et al. in 2009 [43]. Using GPUs for the large matrix operations at the core of DNN evaluation and backpropagation can speed up training by two to three orders of magnitude in some cases.

2.1.6 TDNNs

Most relevant to this work, however, is the introduction of the time-delay neural network (TDNN). Like most innovations in deep learning, it was first documented in the 1980s by a team including Geoffrey Hinton [51]. Their model was simple, small, and only performed phoneme recognition with a vocabulary size of three, but it already identified the most important features of the architecture: the ability to parallelize training, the automatic identification of hierarchical feature representations, and time-

shift invariance accomplished through tied weights. TDNNs were repopularized in 2015 by Peddinti, Povey, and Khudanpur, who showed they could model acoustic context essentially as well as recurrent architectures, but with much greater parallelism possible in training [39]. This performance advantage, along with their feedforward nature, makes them a good choice for large datasets and easier to integrate into an HMM-oriented framework. Since 2015, virtually every Kaldi recipe on large training sets has used TDNNs for acoustic modeling.

Fundamentally, the TDNN is similar to a 1-dimensional convolution over time with a stride of 1 acoustic frame. The lowest layer units have a small window of features as their input, 5 frames of MFCCs in the common case. The next layer up has a window of hidden activations of the lower layer as its input, possibly of a different size than the layer below. This pattern is repeated upwards, each layer condensing a larger amount of acoustic context until the final layer outputs the inference for the current frame given the entire spanning context, typically between 5 and 15 frames on either side of the current frame. Units only have one set of weights, however, and are trained with a gradient accumulated over the entire temporal context. This enforces time shift invariance of the patterns learned and, along with the structure of the model, encourages higher layers in the network to recognize greater levels of abstraction.

The blue boxes and lines in the background of figure 2.5 show a full TDNN with asymmetrical context windows at the upper layers. This is actually inefficient computationally, since the context windows of lower layers significantly overlap. This leads to the optimization known as subsampling. For all the layers above the lowest, only the two inputs at the ends of the context window are considered. Any other inputs are not just discarded, they are never evaluated at all. This leads to the red structure in the foreground of figure 2.5. A full inference is still executed for every audio frame, but many redundant computations are avoided, and the number of parameters is significantly lessened. Moreover, the only hyperparameters to tune above a vanilla feedforward network are the choices of context windows at each layer, generally integers in $[-10, 10]$, which single-handedly determine the structure of the network and the total amount of acoustic context it observes.

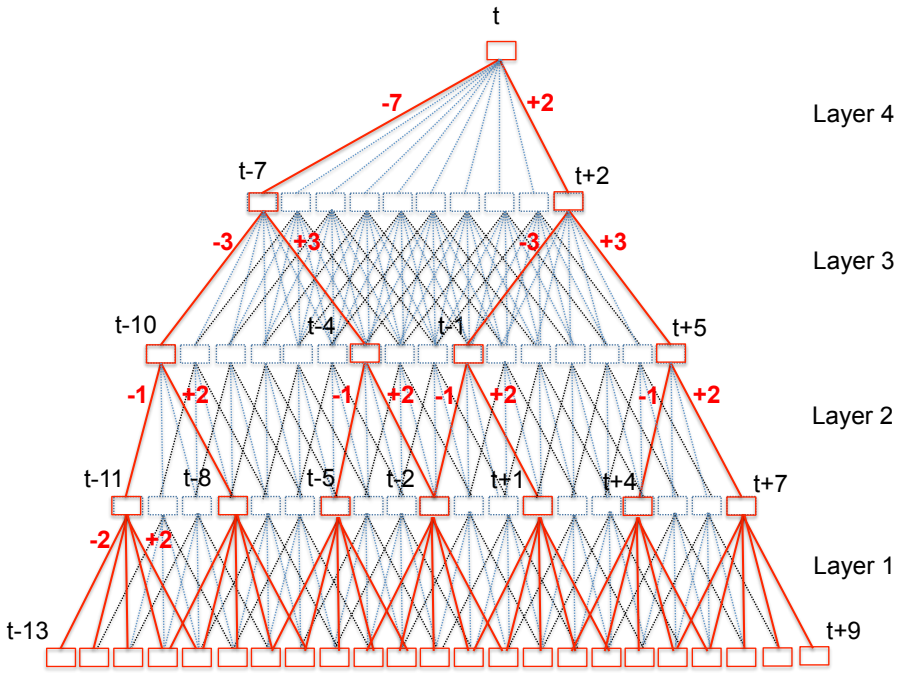


Figure 2.5: Example TDNN architecture with subsampled pathways in red. From Povey et al. [39]

2.2 Children's Speech

2.2.1 Analysis

It is well-documented that ASR systems trained on adult speech corpora perform very poorly on children's speech [44]. Even when highly tuned adaptation techniques specifically designed for this purpose are deployed, children's speech recognition remains less accurate. In fact, even native adult speakers perform significantly worse on children's speech than they do in general, with accuracy sharply decreasing below the age of 10 [13]. This strongly suggests that there are qualities inherent to children's speech that cannot be modeled simply and must be learned from children's speech.

This is problematic because the speech corpora publicly available for academic study are very small. Collecting larger amounts of data of children is proportionally more expensive, and labor-intensive than it is for adults, and there is not as much demand. The problem is further exacerbated by the observation that even when systems are trained on child and adult datasets of equivalent size, the adult systems will still be more accurate [53]. Some researchers have suggested this is due to children have larger acoustic vowel spaces for many vowels than adults do, even when adjusting for

the higher pitch center [13]. Others have connected it to the observation that ASR accuracy per child appears to be correlated to those children's teacher's assessment of their pronunciation quality [26].

2.2.2 Previous Work

A number of previous researchers have investigated applying established techniques to address different challenges intrinsic to the problem of children's speech. D'Arcy and Russell, two recurring names in children's ASR, have documented improvements using vocal tract length normalization (VTLN) and feature-space maximum likelihood regression (fMLLR) to mitigate the large formant frequency shifts found in children's speech [44].

One popular approach is to attempt to augment a children's speech corpus with out-of-domain speech, to leverage the larger amount of adult data available. Alharbi et al. tried it for detecting stuttering [3]. Fainberg et al. successfully used it with stochastic feature mapping to reduce the out-of-domain mismatch [14]. Even Google's work with their extremely large private children's speech dataset incorporated more adult speech than children's [27].

The methods most relevant to this work, however, are those targeting the variance seen in children's speech. Shivakumar et al. demonstrated improvements on the CU Read & Prompted and Story corpora using fine-grained hyperparameter searches and a simple pronunciation modeling scheme derived from a monophone decoding pass through the training data [46]).

It should be noted that corpora of spontaneous children's speech are underrepresented in academia. Extemporaneous children's speech is fraught with non-grammatical passages, babbled speech, and non-lexical filler, all of which pose challenges to recognition in an already difficult domain. Thus, the majority of children's speech corpora are either prompted (PF-Star, CU Read, TIDIGITS) or a mixture of prompted and spontaneous (CMU Kids, CU Kids, OGI Kids). This causes generalization problems because many researchers fine-tune or completely train the language models used in their recognizers on the training data. In many of these corpora, the collectors' focus was on phonetic analysis of children's speech, so they frequently prompted children with short lists of words and sentences deemed "phonetically rich", with noticeable overlap between the training and test sets. LMs trained on the training transcripts for corpora with this structure will therefore have unrealistically low perplexity at test

time. Moreover, LMs trained on broader datasets more representative of human language will have unrealistically high perplexity since the word list utterances have no context for them to base predictions off of. PF-Star is particularly prone to this problem, as further described in section 3.2.

2.3 Pronunciation Modeling

Pronunciation modeling is difficult business. It attempts to codify the relationship between the phonemic definition of a word as found in a standard dictionary and the myriad phonetic realizations of that word in human speech. This relationship can be unpredictable. Humans are extremely accurate at inferring intended phonemes through context clues even when the actual phones spoken vary wildly from the base forms. This is one of the reasons that, while linguists believe English has somewhere between 35 and 44 phonemes [10], a typical pronunciation dictionary used by an ASR system recognizes closer to 340 context-dependent phones. This is an effort to derive an acoustic analog, the “phone”, for a psycholinguistic phenomenon, the “phoneme”. Unfortunately, because phonemes are defined contrastively, it would be virtually impossible to enumerate all feasible phonetic realizations of a given word. Pronunciation modeling is broadly defined as any attempt to address this variation through the structure of the acoustic model.

2.3.1 Implicit vs. Explicit Models

Pronunciation modeling techniques can be divided into two groups: implicit and explicit. Implicit pronunciation modeling at its simplest is just not addressing the issue at all and relying on the generalizing ability of a trained acoustic model working alongside a good language model [22]. One very common method that is slightly more deliberate without being explicit is tied-state HMMs, where a decision tree is used to cluster the Gaussians of the acoustic model according to ”questions” that maximize the likelihood of the training data [17]. This is the default technique for HMM-GMMs in Kaldi.

Explicit pronunciation modeling is any technique that attempts to specifically encode the pronunciation relationships that it witnesses. Typically, this means creating new phonetic transcriptions for words in the pronunciation dictionary and using them to expand the lexicon, or assigning probabilities to existing alternatives in the lexicon

[12]. New pronunciations and probabilities can be generated manually by a linguist, by rule-based algorithms like the popular Sequitur-G2P[9], or estimated from patterns in the training data [56].

Implicit pronunciation modeling has worked very well for many systems, arguably the vast majority of ASR systems ever developed. The context-dependent triphone model used by many modern systems is highly effective at approximating the psycholinguistic knowledge mentioned in section 2.3. Some researchers have argued that any explicit modeling, regardless of how data-driven it is, represents a specious application of human knowledge that is best left to the acoustic model [23]. Others, however, like Khudanpur, Povey, et al. continue to document improvements in system accuracy when leveraging explicit modeling [56].

Chapter 3

Methods

3.1 Kaldi Toolkit

Kaldi is an open-source toolkit for speech recognition [41]. It was first published in 2009 by a speech recognition workshop at Johns Hopkins University [2]. It was heavily inspired by and originally heavily depended on an older ASR toolkit: HTK [55]. It has grown dramatically since then under the supervision of maintainer Daniel Povey, who has also contributed to a substantial number of the state-of-the-art results Kaldi's recipes have been responsible for.

The core of Kaldi is large C++ class library implementing a litany of feature extractions, transformations, acoustic models, etc. The toolkit builds against OpenFst[4], another open source toolkit providing efficient operations on WFSTs that Kaldi uses to implement its training and decoding graphs. HMMs, pronunciation dictionaries, and n-gram language models are all rendered as WFSTs for Kaldi's purposes, allowing them to be composed together into a single computational graph roughly as described by Mohri et al. back in the 1990s [32]. Kaldi also provides its own full-featured, hardware-optimized deep learning framework, complete with a template language for specifying architectures and tools for using different objective functions.

3.2 The PF-Star Dataset

The PF-Star project was an international cooperative research effort sponsored by the European Union [7]. There were several speech corpora collected in different countries and languages as part of this project, but the one that I refer to as "PF-Star" in this paper is a corpus of prompted English children's speech. It was collected in the mid-

2000s at two primary schools and a university lab in the Midlands. I chose to focus on it primarily because previous work on it has struggled to break past 30% WER, and I believed there was possibility for improvement through pronunciation modeling. After delving deeply into it, however, it has become clear that the challenges the dataset presents are manifold.

To start, the corpus is small, with only 7.5 hours in the training set and 5 hours in the test set. Its curators attempted to mitigate its size by designing the prompted utterances to be as phonetically "rich" as possible. However, their method of doing so has unintended consequences. First, there is significant overlap in the transcripts of the training utterances and the test utterances. Of the 72 unique text prompts that were given to children to create the test set, every one also appears in the training set. In fact, there are only 10 prompts that appear in the training set but not the test set. Due to the unpredictable nature of children's speech and the presence of a data collector in the room, the actual reference transcripts for these utterances are not identical, but they are extremely close. The primary consequence of this is that using the training data transcripts for any type of language model training or fine-tuning represents a huge target leakage. Virtually every example of previous research done on PF-Star achieving a WER better than 28% has used a language model biased in this fashion [18][3].

The leakage problem aside, PF-Star still creates language modeling challenges. The utterances are a mixture of lists of digits, lists of words, complete sentences, and grammatical fragments in rapid succession. This unpredictable grammatical construction means homophones (two & to, there & they're) often hit substitution errors through no fault of the acoustic model. The usage of non-standard spellings and proper nouns ("tunafish", "trish") also creates ambiguity that might be reasonable to expect an ASR system to gracefully handle, but not with such a small amount of data to work with.

Acoustically, the challenges posed by the corpus are more in line with expectations. There is extreme variation between speakers, not only in the strength of their dialects (from Birmingham and Worcestershire), but also in the clarity and speed of their articulation. This is unsurprising for children's speech. There is also substantial background noise, which is somewhat unexpected given the use of a head-mounted microphone. A handful of speakers have such loud background noise (what sounds like construction equipment immediately outside the room) that they are disproportionately difficult to recognize. Perhaps the most significant acoustic challenge is that, unlike the

transcripts, there is zero overlap in the speakers of the test set and training set. Both sets have some speakers exhibiting variability from mild lisps, de-rhotacization, and reading errors, but the test set contains two speakers with severe speech sound disorders (022m11bh & 005m08bh) which have no equivalent in the training set. Several of the transcripts appear to be deliberately difficult to pronounce without making articulation errors ("he will allow a rare lie", "where were you while we were away"), but this does not constitute example data to train a model to account for these disorders at test time.

3.3 Model Design and Baseline

The design and training of the final acoustic model was inspired by Kaldi's most recent recipe for the WSJ corpus, though the data preparation and alignment procedures were devised just for PF-Star.

3.3.1 Data Preparation

The data preparation script was based off prior work by Fainberg et al. on PF-Star [14], but had to be rewritten from scratch to work around compute cluster file system issues and to accommodate additional experiments mentioned in sections 3.3.1 and 3.4.4.

The text files for PF-Star are in the "label" format designed for use with HTK. They contain word-level transcripts including millisecond-resolution alignment data and annotations for silences. Kaldi has its own best practices for alignment, so I discarded both of these. Then, all words were converted to uppercase and written to the text file passed to Kaldi.

The transcripts also contain brackets around words that (in the opinion of the transcribers) were poorly articulated enough by the children that their value as acoustic training data is dubious. I intended to conduct separate experiments comparing performance when these garbled words were replaced with "<UNK>" versus having their brackets removed and being treated like any other word. Unfortunately, the transcribers went a step further than the documentation described. For many words that a human could confidently infer the correct transcription of, the transcribers replaced the exact letters they felt were misspoken with asterisks. This decisively puts these words outside of any ASR system's dictionary. Theoretically, it would be possible to reconstruct their original spellings given that the text prompts used are known, but this is outside

the scope of this project. In all my experiments, these words were simply replaced with "<UNK>", which the pronunciation dictionary maps to a garbage phone.

The audio files for PF-Star are stereo WAV files encoded with 16-bit signed PCM at a sample rate of 22.05 kHz. Best practice for Kaldi (and many other ASR systems) is to use single-channel audio sampled at 16 kHz, so the audio was mixed down to one channel and downsampled to 16 kHz. There is some doubt about the interpretation of the two channels in these files (see Sec. 5.3), and early on I aimed to do contrastive experiments using one channel or the other. However, the experimental variable in this case happens early in feature extraction and would require a complete beginning-to-end retraining of the system, which takes over two days and a dozen gigabytes of drive space on an extremely crowded cluster. Thus, it had to be dropped, and all the experiments I document here were done with a simple mixdown to mono.

3.3.2 Features

The features used as input to the system are mel-frequency cepstral coefficients (MFCCs) as used in many ASR systems [54]. The settings are mostly the Kaldi defaults for feature extraction: 25ms long frames sampled every 10ms, 23 mel-spaced frequency bins, with an end result of 13 cepstral coefficients per frame. These MFCCs are extracted from both the training set and test set ahead of time, as online recognition is not a focus of this project. To generate phone alignments for training the neural acoustic model, a series of successively more accurate HMM-GMM systems are trained, each one applying a different transform to these features. For these initial rounds of alignment, the features are not augmented in any way other than the specified transform. For training the final acoustic model (see section 3.3.6), 3-fold speed perturbation is applied, replacing the acoustic data with copies of itself at 0.9x, 1.0x, and 1.1x speed.

3.3.3 Alignment

In this case, the series of systems for alignment is a monophone system followed by three triphone systems. The first monophone system concatenates each frame of MFCCs with their deltas computed over a sliding window of 5 frames, then does the same again with those deltas to generate second-order deltas. It also adds an extra feature representing the log-energy of the frame, for a total of 40 features per frame. These are not saved to disk, but are extracted identically for the first triphone system, which also trains on the delta-delta features. The second triphone system instead

splices features across 9 frames, uses linear discriminant analysis (LDA) to reduce the dimensionality back down to 40, then iteratively estimates a maximum-likelihood linear transformation (MLLT) matrix that maximizes the likelihood of the features given the first triphone model. The third and final triphone system uses the feature transformation from the LDA/MLLT system, then further estimates a maximumim likelihood linear regression inside the transformed feature space (fMLLR) for each speaker in the training set. These transformations will not be available at decoding time, but still help the model better generalize at training time.

The decoding accuracy of these successive models can be seen in the first four points of the chart in figure 3.1.

3.3.4 Lattice Free-MMI

The final acoustic model used in these experiments is a "chain" model. This represents a set of optimizations Kaldi uses when training and decoding DNN-HMM models on the MMI objective function described in section 2.1.3 [2]. First, the HMM transition probabilities are fixed; the neural network is relied on to produce the best possible state probabilities within this setting. Second, the output frame rate is made three times smaller to save computation at decode time. This also means the topology of the HMM must be tweaked slightly, to allow traversing most phones in a single state. All of this is mostly in service of the goal of training the neural net on the MMI objective function.

Most neural network ASR systems are trained frame-by-frame on the cross-entropy objective and their final layer is a softmax over the set of phones. This means that the training at each frame is supervised by the single phone the input alignments have at that position. Deviating from the alignments is penalized, even though there are myriad ways a neural net could allowably deviate from them without decreasing its final accuracy. At 100 inferences per second, there could be many frames that are not representative of the phones they are aligned to, and the alignments are not guaranteed to be completely correct. Training against MMI addresses this, training sequence discriminatively instead of frame-by-frame. Instead of maximizing the neural network's likelihood of selecting one exact phone, it maximizes the probability of the neural network generating the entire reference phone sequence given its role in the hybrid acoustic model. This means training the network towards exactly what the goal at decode time is, instead of training to an approximation of that goal, and it is documented

to improve word error rates, particularly on non-recurrent architectures [42].

The problem with MMI, however, is how to compute it. Broadly speaking, computing MMI in the context of training an acoustic model means dividing the probability of the correct phone sequence given the acoustic data by the probability of all other phone sequences. Computing the denominator of this fraction naively would be unbelievably computationally expensive. In earlier approaches to using MMI, the denominator was approximated using lattices. The "chain" model in Kaldi is structured around making the true, non-approximate computation tractable, including the aforementioned optimizations. Kaldi also uses a number of other shortcuts to optimize it. At training time, it compiles a full decoding WFST for the denominator at the phone-level instead of the word level. This includes an interpolated 4-gram/trigram phone-level language model trained on the training data and an HMM recognizer. Then it runs the forward-backward algorithm on this graph to estimate the posteriors of all the neural network outputs. It computes the numerator in a similar way, building a full FST for each utterance, but this time it does use lattices to represent the known alignment, including acceptable variations from it, i.e. some phone alternatives and "wiggle room" on the edges of each phone [42]. All of this is evaluated using a GPU with very granular optimizations to ensure memory locality and a host of other hacks to enable minibatches within utterance boundaries.

3.3.5 TDNN-F

The neural network used in this acoustic model is the factorized TDNN (TDNN-F) built for the WSJ recipe. The TDNN-F was inspired by the observation that many engineers reduce a trained network's size for deployment by finding the truncated singular value decomposition (truncated SVD) of the layer's weight matrix, then pruning the singular values below a certain threshold. It works as follows: Given a matrix M, proper SVD finds the factorization $M = U\Sigma V*$ where Σ is diagonal and the factors U and V are unitary. The values along the diagonal of Σ are "singular" values. Ranking and removing those below a certain threshold amounts to pruning out entire dimensions of the relationship between the factors that, according to SVD theory, have the least influence on the value of the approximated matrix [47]. The number of singular values retained is the width of the bottleneck applied to the network, a tunable hyperparameter.

Attempts to train networks under this constraint from the start (a valuable propo-

sition, if viable) have had problems with training not converging [40]. The TDNN-F design stabilizes training by enforcing the constraint that one of the factor matrices is semi-orthogonal. That is, that the product of the matrix and its transpose (or vice versa) is the identity matrix. Strictly enforcing this would be difficult, so the TDNN-F does it approximately. After every few iterations of stochastic gradient descent, the training pauses and performs an iteration of gradient descent on the factor matrix towards an objective defined by the semi-orthogonality formula instead of the training objective. Given a factor matrix is M, if it were semi-orthogonal, $MM^T = I$ would be true. Formalizing the difference Q between M and its semi-orthogonal version as $Q = MM^T - I$, then minimizing the function $f = tr(QQ^T)$ is equivalent to constraining M to being semi-orthogonal. The update the TDNN-F applies to the M after every few iterations of training is the gradient of f with respect to Q, and can be read in detail in [40].

3.3.6 Acoustic Model

To summarize, the acoustic model in my initial experiments is exactly as it was designed for the WSJ corpus. It consists of a factorized TDNN trained on Kaldi's lattice-free implementation of the MMI objective. This TDNN-F is taller and has narrower strides than the TDNN depicted in figure 2.5, with 13 hidden layers. The lowest three layers are not subsampled and have offsets of $[-1,+1]$. All the layers above are subsampled with offsets of $-3,0,+3$. The stride of 3 makes for very efficient evaluation alongside the modified frame rate the "chain" model imposes (see section 3.3.4). Each layer has 1024 units with a linear bottleneck (see section 3.3.5) of 128. The model uses the three measures mentioned in section 2.7 of [42] to prevent overfitting. The cross-entropy regularization constant is 0.1, the l_2 regularization constant is 0.005, and the leaky HMM coefficient is 0.05. The phone-level LM used to compute LF-MMI denominators is allowed 2000 4-grams, with everything else backing off to trigrams. At the start of training, the learning rate is 0.0005, decaying exponentially to 0.00005 by the 10th and final epoch. The training starts with 2 parallel jobs and linearly grows to 8 parallel jobs by the end. At the end of training, the model is decoded on the test set using a trigram language model trained on the WSJ transcripts pruned to a minimum likelihood of 10^{-7}. The lattices from decoding are then rescored using an unpruned 4-gram language model trained on a larger corpus of transcripts.

All these initial hyperparameter choices came from the most recent WSJ recipe

("tdnn1g" at the time of writing) which was specifically designed to work reasonably well on smaller datasets. Initial experiments with it achieved $\sim 32\%$ WER, respectable for an untuned model. Comparing the path of training loss and validation loss over epochs showed that validation loss was nearly 3 times higher, indicating overfitting, which is a known issue with LF-MMI training. I raised the l_2 regularization constant from 0.005 to 0.01, raised the leaky HMM coefficient from 0.05 to 0.1, and changed the number of parallel jobs at the end to 4, since at 8 jobs, each one was calculating training updates on less than 1 hour of data. This lowered the WER to 30.89% and made the training loss/validation loss difference much smaller.

The WSJ recipe includes the ability to rescore lattices using a recurrent language model (RNNLM), also trained on the WSJ transcripts. I believed this would be helpful in resolving some of the homophone and grammatical ambiguity mentioned in Sec. 3.2. However, it actually resulted in worse accuracy than the 4-gram rescoring. This was suspicious, and further investigation revealed that the 4-gram WSJ language model had very high perplexity on the PF-Star test set (approx. 1600) and actually scored worse than the smaller trigram model did (approx. 1000). Furthermore, the best WERs from these rescorings almost always used the minimum language model weight. I took this as an indication that the WSJ language model domain (politics and finance) was too outside the scope of PF-Star. I migrated the system to use pruned trigram and unpruned 4-gram language models from the Librispeech dataset instead, which is trained on text from audiobooks [37]. This immediately improved the score to 27.68% on the 4-gram language model, and became the final baseline system.

Order	WSJ LMs (unmodified)	WSJ LMs (tweaked hyperparameters)	Librispeech LMs
Pruned trigram	33.25	31.38	29.80
Full 4-gram	32.62	30.89	**27.68**

Table 3.1: TDNN accuracy with different hyperparameter choices and language models

3.3.7 Pronunciation Modeling Experiment

Initially, this experiment was to consist of manually creating a pronunciation confusion graph that would represent a formal linguistics-driven view of common pronunciation variations and errors made by children. This would be converted into an FST, then composed with the lexicon FST of the baseline system and the acoustic model would

be trained again from scratch. There was an alternative, however. Kaldi, in fact, has tools to estimate pronunciation probabilities from lattices and incorporate them into an existing lexicon. Using these tools meant weakening the firm link between deep learning and formal linguistics that was one of the initial goals of this project. It did not break the link entirely, however, and estimating pronunciation probabilities directly from experience is a more attractive prospect for general use than the application of expert knowledge, so I decided to use these tools.

To begin with, these tools implement the pronunciation probability derivation documented in the 2015 paper by Chen et al. [12]. They do not invent new pronunciations, they count the occurrences of pronunciation alternatives already present in the lexicon so their relative probabilities can be estimated. The core of this logic is in Kaldi's C++ function `CompactLatticeToWordProns`. This function iteratively expands the arcs of an n-best lattice, saving the resulting phone sequences as alternative pronunciations. The n-best lattice required can be generated by aligning the training data with an HMM system and using Kaldi's `linear-to-nbest` on the alignments. The pronunciation counts are then written to a new version of the lexicon and normalized to have a maximum of 1 for each word. This is to avoid the most common pronunciation of a word being penalized over pronunciations of other words simply because they have fewer alternatives. At first glance, this appears to violate the law of total probability, but since the probabilities will ultimately be encoded as costs in an FST, it does not pose an issue.

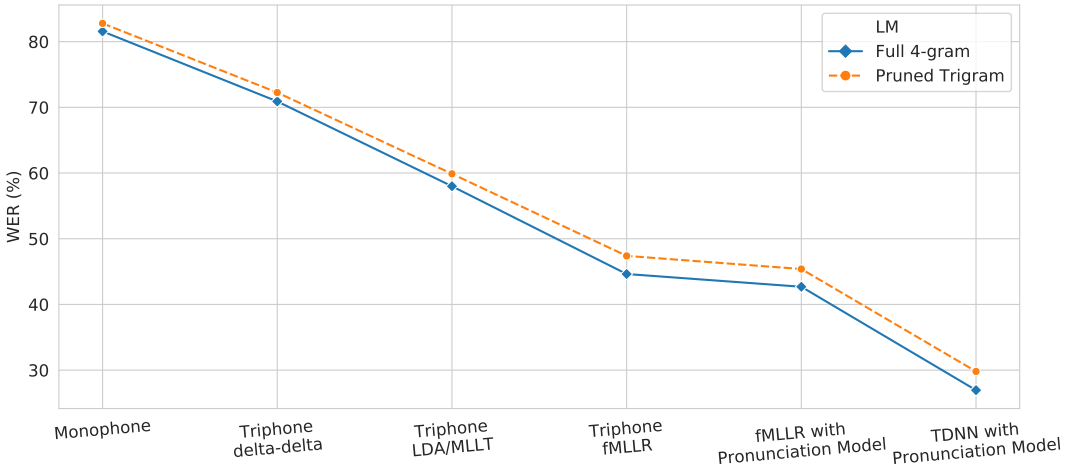
With this implemented, instead of moving on to training the neural acoustic model after the final round of alignment on fMLLR features, the system in this experiment uses the alignments to estimate pronunciation probabilities. These are added to the lexicon and a second fMLLR system is trained on them. This second system is used to generate alignments. Then, the TDNN acoustic model is trained using these alignments and the newly probabilistic lexicon, which is also used at decoding time. The decoding accuracy on the test set of the entire sequence of models can be seen in Fig. 3.1.

3.4 Results

3.4.1 Baseline System

Given the tight constraints on training data available, and the limited compute resources to do a large hyperparameter search, the baseline TDNN-F performed well.

Figure 3.1: WER of successive models



Its results with different hyperparameters are shown in table 3.1.

3.4.2 Accuracy

Ultimately, the pronunciation modeling achieved a modest improvement in accuracy, from 27.68% to 26.95%. This is not a revolutionary improvement, but after consulting a large body of previous research, it appears to be the lowest error rate achieved by any project not utilizing a biased language model or training on additional acoustic data outside of PF-Star.

3.4.3 Pronunciation Model

In total, the n-best count pronunciation modeling method generated just over 300 probabilities other than 1 to add to the lexicon. This sounds like a small number, but the transcripts of the data only contain 1239 distinct words, so it is actually a significant portion of the vocabulary having its frequency modeled.

3.4.4 Secondary Experiment: Segmentation

One small confounding issue in the PF-Star corpus is that fact that many of its utterances, primarily those with "list" in their file name, are not composed of grammatically distinct units. According to the corpus' curators, these utterances are ten "phonetically rich sentences" read back-to-back. In reality, a large minority of them are not sen-

tences, but fragments. For example, “My favourite champion”, “With a pointed nose”, and “A very good vision” all appear in these lists, frequently adjacent to other unrelated fragments. They also frequently begin with words that are likely to rely on disambiguation from the language model, “I’ve” vs. “I’m”, and “there” vs. “the”, to name a few. All of this becomes a problem because the default practice in Kaldi is to assume each audio file and corresponding transcript represents an independent utterance for language modeling purposes. This is clearly not the case for these files, and I believed this might be causing unnecessary errors at decode-time.

For these reasons, I conducted an alternative experiment attempting to address these utterances separately. I modified the data preparation scripts to make a best-effort attempt at segmenting the recordings with multiple sentences (those with “list” or “sentences” in their names, not those with “digits” or “w_list” in their names, which are just spoken digits and words without any context). I decided on a simple heuristic to determine utterance boundaries: Assume all silence annotations not at the beginning or end of the text file are utterance boundaries. Then, if more than a threshold number of boundaries were detected, assume this speaker had such an irregular flow of speech that a basic segmentation is impossible, and the entire file should be considered a single utterance, as in the baseline described in Sec. 3.3.1. After some manual experimentation, I chose 15 segments as the threshold value, since it filtered out all the egregiously stuttered files, but allowed a number of cases where speakers merely had to repeat a few utterances again at the prompting of the data collector. The segmentations derived were straightforward to implement since the label files contained timing annotations that could be parsed and reformatted into a “segments” file, a Kaldi feature that supports forming multiple independent utterances from each recording using timing information.

After deriving the segmentation timings and retraining the system from scratch, it became clear that this method provided no advantage. The WER on the baseline model increased from $\sim 30\%$ to 44%. It appears that for many of the segments, the amount of silence before and after the utterance was too short for the Kaldi HMM implementation to handle well. The experiment does demonstrate, however, that a lack of language model adaptation is not nearly as harmful as a basic acoustic effect like uncomfortably short silences.

Chapter 4

Discussion

4.1 Analysis

4.1.1 Baseline Model

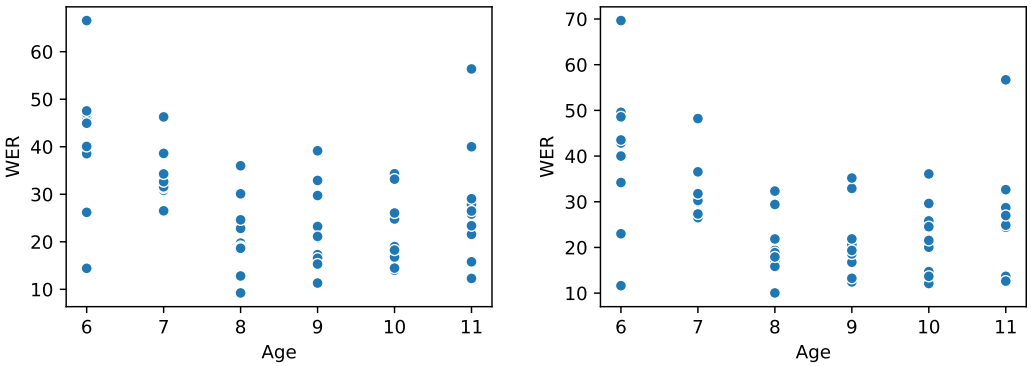
The baseline model is a strong one considering its task. I deliberately set the constraint that only the acoustic data from PF-Star would be used, and as explained in Section 3.2, this dataset is problematic on many levels. The Kaldi recipe this model was adapted from was built to work on the WSJ corpus. Like PF-Star, WSJ is prompted, uses both head-mounted and general-purpose microphones, and exhibits widely varying amounts of background noise. The biggest difference is the size. The WSJ corpus is over 300 hours of speech, while PF-Star is closer to 8 hours, 13 including the test set. Overall, however, it appears the architecture of the model has transferred reasonably well. The hyperparameter tweaks made to mitigate overfitting were not onerous, and the accuracy achieved is impressive given the circumstances.

Crawling the decoding logs made it clear, however, that the score was consistently being penalized for mistakes that were not true mistakes. Decoding "favourite" as "favorite" generated 15 substitution errors, "grey" as "gray" made 9 more. In the strictest sense, this is a language modeling concern. I was aware at implementation-time that the corpus was of British English, and ensured that the British English spellings of words existed in the dictionary used. But the language models were trained on multi-dialectical text, not British English exclusively, and thus would be unable to distinguish the correct spelling to use unless there was an explicit hint within their n-gram history. This is discussed more in section 5.2

Plotting the speakers' ages against the WER achieved by the baseline (Fig. 4.1a)

shows fairly clearly that the model struggles more with the younger speakers, which is expected. The speaker age and WER appear to be inversely correlated. The Pearson correlation coefficient observed is -0.42, with a p-value of 0.00086. This is extremely unlikely to be a coincidence, and it corroborates previous research on the difficulties of children's speech [26].

Figure 4.1: Per-speaker error rates by age



(a) Baseline model

(b) With pronunciation modeling

4.1.2 Interpretation of Pronunciation Modeling Experiment

Kaldi's implementation of counting pronunciations appears to have struggled somewhat in this experiment. While it generated a reasonable number of alternative pronunciation probabilities, and it helped recognition accuracy, the numbers could have been stronger.

I focus on the deletions made by the model here because, intuitively, adding pronunciation probabilities to the model is likely to have the biggest effect on the words that it initially failed to map to anything in the dictionary at all. In total, the baseline made 909 deletion errors, while the pronunciation model made 768, an improvement of over 15%. The counts for insertions and deletions were not significantly different between the models. Figures 4.2a and 4.2b show the distributions of word deletions made by the baseline and pronunciation model respectively. It is clear that the pronunciation model decreases the number of words that experience high rates of deletion. Words like "bob" and "draw" that have unusual pronunciation in a Birmingham accent both have their deletion counts strongly improved, from 7 to 4 and 11 to 7, respectively. Words like "a" and "it's" that are prone to elision or severe coarticulation by children

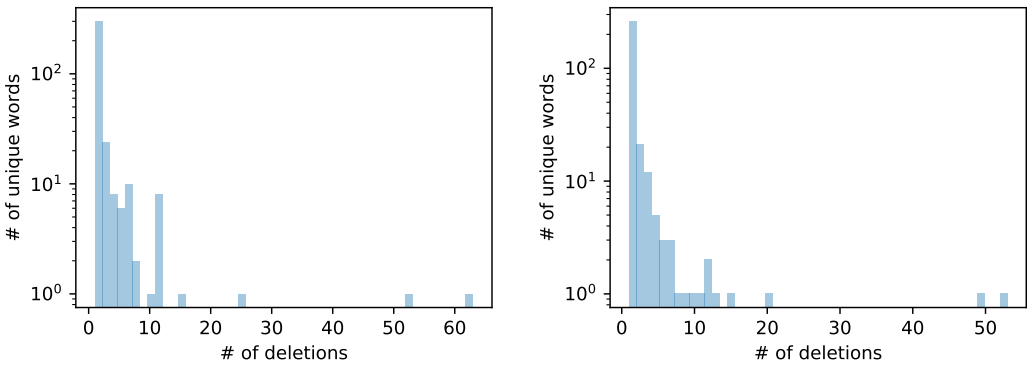
are also improved, from 63 to 53 and 12 to 7, respectively.

There is evidence, however, that the pronunciation model could be doing more. Running Kaldi's recipe for the original WSJ corpus on the WSJ corpus up to its final triphone alignment and generating pronunciation counts using the same method showed over 3500 probabilities being added to the lexicon, contrasted with only 300 in this experiment. The pronunciation modeling code is designed to estimate pronunciation probabilities only for alternatives already in the lexicon [12]. This makes the lexicon choice an important hyperparameter, as discussed in section 5.2.2.

Because PF-Star is small and has a great deal of overlap in transcripts, its functional vocabulary is very small, just over 1200 words. Generating new pronunciation probabilities for 1 in 4 of these words is actually relatively productive. The WSJ recipe's 3500 new probabilities were on a functional vocabulary of nearly 15,000 words, with over 10 times more training data to estimate pronunciations counts off of.

The obvious follow up experiment would be to use out-of-domain probability estimations. Unfortunately, this proved a dead-end. The only out-of-domain model I had trained was the original WSJ recipe, but after switching to the Librispeech language model and dictionary preparation scripts, the WSJ lexicon no longer matched the current one closely enough for the pronunciation probabilities to integrate correctly. I attempted to train the base Librispeech recipe to extract pronunciation counts there instead, but file system issues with a shared cluster made this impossible.

Figure 4.2: Per-word deletion counts



(a) Baseline model

(b) With pronunciation modeling

Chapter 5

Conclusions

5.1 Research Hypothesis

This research is essentially an extension of the experiments Povey et al. did on the Switchboard corpus in 2015[12] and 2017 [56]. Those researchers concluded that explicit pronunciation modeling created a small but statistically significant improvement in accuracy. I hypothesized that it would have an even more helpful effect on children's speech due to the unique qualities of children's articulation and the sparsity of the data available. Ultimately, the pronunciation modeling technique investigated in this research created the same effect as it did previously: a small but statistically significant improvement in accuracy. This represents a gentle rejection of my hypothesis that children's speech would be different.

5.2 Limitations of This Research

5.2.1 Data Normalization

The PF-Star dataset is challenging for a litany of reasons mentioned in Sec. 3.2. Unfortunately, many of these only became clear late in the experimentation. I noted the non-normalized spellings of certain words causing scoring errors (e.g. "tunafish") early on, but was hesitant to add code to modify these out of fear it would ruin the comparability of this work to earlier projects on PF-Star. I realized this was an incorrect conclusion and moved to rectify it, but attempts to run the experiment from the beginning again proved fruitless due to compute cluster file system issues. I also noticed the higher LM perplexity of the Librispeech LMs on British English words causing decoding

penalties (e.g. “favorite” instead of “favourite”, “color” instead of “colour”) but was hesitant to migrate to a different, less general, language model after criticizing previous researchers for fitting theirs to the corpus. I searched for tools that might allow normalizing the Librispeech transcripts to British English and retraining the LMs, but none of the tools I found were both freely accessible and well-reviewed by researchers that applied them to speech recognition. I attempted to train the RNNLM for the Librispeech set, believing it would be more effective at choosing British spellings of words given more context, but training it on a shared cluster proved to be intractable. I defend the fact that both of these problems didn’t get enough of my attention because I was too focused on engineering workarounds for computing issues, but ultimately, they were still my fault for being too hesitant until too late to effectively apply my knowledge of being neck-deep in this dataset in fear of doing something scientifically indefensible.

Another consequence of the large amount of time spent debugging shared hardware issues is that I could not afford to perform an extensive hyperparameter search on the baseline model. Each training run took over a day, so my tweaks to the parameters mitigating overfitting (see section 3.3.6) were largely guesses; I only checked two other sets of values before determining them to be worse and moving on.

5.2.2 Pronunciation Modeling Implementation

Looking hard at the results of the pronunciation modeling experiment, it seems that it suffered from the same problem that everything does on children’s speech, not enough data. Using a data-driven pronunciation modeling scheme will suffer from data sparsity just like anything else. The number of pronunciation probabilities it estimated was good considering how small the dataset was, but I had hoped for better, and specific words I had in mind for pronunciation modeling (e.g. “draw” and “foot”) had their pronunciation alternatives scored extremely low, despite being represented in the training data to my ears. However, previous research has suggested that many of the relationships data-driven pronunciation modeling is capable of estimating are very similar to those that child speech development experts would identify [16], so my anecdotal assessment is probably just that, anecdotal.

One possible underlying cause is the lexicon choice. The Librispeech lexicon recipe uses CMUDict at its core, which is a pronunciation dictionary originally designed for American English, not the British English heard in PF-Star. The pronunciation modeling code only estimates probabilities for existing pronunciation alternatives

in the lexicon, making this choice more impactful than I intended when I first made it. Experimenting with a different lexicon would have been an excellent further data point to examine the performance of pronunciation modeling. It was one of several experiments I was disappointed to not have the time to perform.

Ultimately, however, my largest disappointment with this result is the inescapable fact that the context-aware "phones" that are the output vocabulary of the acoustic model do not directly correspond to specific speech sounds. For any given word, the set of allowable phonetic realizations of its phonemic definition is large and ill-defined. If the phones of a neural acoustic model truly represented individual phonetic units, these different pronunciations would be enumerated and the models conclusions could be precisely defined. This ends up not being the case because expanding a recognizer's dictionary by a large factor creates far more confusion than it resolves. Therefore, some level of implicit pronunciation modeling remains necessary, wherein the acoustic model itself allows for sufficient variation in phonetic realizations of sounds. This has conceptual problems, though. Spoken phonemes are defined contrastively, not mathematically, and the model is aware of only acoustic context, not some hypothetical graph of all minimal pairs for every phoneme and word in its vocabulary, so it cannot have sufficient information to deduce the acceptable realizations of a given phoneme in context.

Collectively, this means that the "phones" in an acoustic model represent neither objectively-defined acoustic phones, nor contrastively-defined linguistic phonemes. They exist in a muddy in-between space, allowing a specific amount of variation that minimizes an error function and hoping that the final probability distribution over phones combined with the language model is enough to sort out the rest. Of course, this is by design. Strong model generalization and regularization are valued traits in machine learning. They just don't integrate cleanly with efforts to add explicit knowledge to a model.

5.3 Future Directions

Given a larger amount of time and computing power, there are certain follow-up experiments that I believe would be particularly helpful in addition to fixing some of the issues identified in section 5.2. The very first would be to estimate pronunciation counts from a different corpus. My attempts on WSJ were thwarted, but with more time others would be practical. The Accents of the British Isles (ABI) corpus comes to

mind, perhaps with a specifically selected subset of the accents used to train the model. This could be compared with a similar experiment on other children's speech datasets like Ultrasuite or CU Kids.

It's also clear that a great deal of progress could be made on the PF-Star dataset by implementing some normalization practices. To begin with, the division of training and test set chosen by the corpus' curators seems almost arbitrary. Repartitioning it randomly instead of by speaker or merging it into one set and encouraging users to apply k-fold cross validation would both be valid alternatives. Also, resolving the confusion of which audio channel corresponds to which microphone would be a boon to future research on the dataset. In the documentation, it is strongly implied that in the stereo audio files, the left channel is from a head-mounted microphone, and the right is from a table-mounted microphone, but this is not stated unambiguously [7] and to my ears, it sounded like they switched in different recording. Comparing experiments run separately on these channels would help inform future projects using the dataset.

5.4 Final Words

This project set out to demonstrate that an often-overlooked technique in the ASR toolkit could have an outsized effect when applied to an often-overlooked language group. In the end, that hypothesis was not sustained. However, many ancillary discoveries were made along the way, particularly about the idiosyncrasies of the PF-Star corpus, the great care that must be taken when choosing a language model, and the dirty, hands-on work that implementing ASR systems still involves, regardless of how advanced the tools have become. In particular, I hope my identification of irregularities in PF-Star may prove helpful to future researchers. Having these issues identified ahead of time can make novel research much more productive and comparable to previous work.

Bibliography

[1] Ibm 100: Pioneering speech recognition. https://www.ibm.com/ibm/history/ibm100/us/en/icons/speechreco/. Accessed: 2019-08-05.

[2] Kaldi. http://kaldi-asr.org/doc/index.html. Accessed: 2019-08-07.

[3] S. Alharbi, A.J.H. Simons, S. Brumfitt, and P.D. Green. Automatic recognition of children?s read speech for stuttering application, November 2017. @ 2017 International Speech Communication Association. Reproduced in accordance with the publisher's self-archiving policy.

[4] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. Openfst: A general and efficient weighted finite-state transducer library. In Jan Holub and Jan Žďárek, editors, *Implementation and Application of Automata*, pages 11–23, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[5] L. Bahl, P. Brown, P. de Souza, and R. Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 49–52, April 1986.

[6] J Baker. The dragon system–an overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29, 1975.

[7] Anton Batliner, Mats Blomberg, Shona D'Arcy, Daniel Elenius, Diego Giuliani, Matteo Gerosa, Christian Hacker, Martin Russell, Stefan Steidl, and Michael Wong. The pf-star children's speech corpus. In *Ninth European Conference on Speech Communication and Technology*, 2005.

[8] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Statist.*, 41(1):164–171, 02 1970.

[9] Maximilian Bisani and Hermann Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434 – 451, 2008.

[10] Aldo Bizzocchi. How many phonemes does the english language have? *International Journal on Studies in English Language and Literature (IJSELL)*, 5:36–46, 10 2017.

[11] H. Bourlard and N. Morgan. Continuous speech recognition by connectionist statistical methods. *IEEE Transactions on Neural Networks*, 4(6):893–909, Nov 1993.

[12] Guoguo Chen, Hainan Xu, Minhua Wu, Daniel Povey, and Sanjeev Khudanpur. Pronunciation and silence probability modeling for asr. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[13] Shona D'Arcy and Martin Russell. A comparison of human and computer recognition accuracy for children's speech. In *Ninth European Conference on Speech Communication and Technology*, 2005.

[14] Joachim Fainberg, Peter Bell, Mike Lincoln, and Steve Renals. Improving children's speech recognition through out-of-domain data augmentation. 2016.

[15] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. In *Proceedings of the 17th International Conference on Artificial Neural Networks*, ICANN'07, pages 220–229, Berlin, Heidelberg, 2007. Springer-Verlag.

[16] Eva Fringi and Martin J Russell. Analysis of phone errors attributable to phonological effects associated with language acquisition through bottleneck feature visualisations.

[17] Mark Gales, Steve Young, et al. The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, 2008.

[18] S. Ganji and R. Sinha. Exploring recurrent neural network based acoustic and linguistic modeling for children's speech recognition. In *TENCON 2017 - 2017 IEEE Region 10 Conference*, pages 2880–2884, Nov 2017.

[19] John S. Garofolo, William M. Fisher, Jonathan G. Fiscus, David S. Pallett, and Nancy L. Dahlgren. Darpa timit: : acoustic-phonetic continuous speech corpus cd-rom, nist speech disc 1-1.1. 1990.

[20] John J Godfrey, Edward C Holliman, and Jane McDaniel. Switchboard: Telephone speech corpus for research and development. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520. IEEE, 1992.

[21] Alex Graves, Santiago Fernndez, and Faustino Gomez. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *In Proceedings of the International Conference on Machine Learning, ICML 2006*, pages 369–376, 2006.

[22] Rainer E Gruhn. *Statistical Pronunciation Modeling for Non-Native Speech Processing*. Signals and communication technology. Springer, Dordrecht, 2011.

[23] Thomas Hain. Implicit modelling of pronunciation variation in automatic speech recognition. *Speech Communication*, 46(2):171–188, 2005.

[24] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[25] Kai-Fu Lee. *Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system*. PhD thesis, 1988. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2016-03-17.

[26] Qun Li and Martin J. Russell. An analysis of the causes of increased error rates in childrens speech recognition. In *INTERSPEECH*, 2002.

[27] Hank Liao, Golan Pundak, Olivier Siohan, Melissa Carroll, Noah Coccaro, Qi-Ming Jiang, Tara N. Sainath, Andrew Senior, Franoise Beaufays, and Michiel Bacchiani. Large vocabulary automatic speech recognition for children. In *Interspeech*, 2015.

[28] Richard P. Lippmann. Review of neural networks for speech recognition. *Neural Computation*, 1(1):1–38, 1989.

[29] Bruce P. Lowerre and B. Raj Reddy. Harpy, a connected speech recognition system. *The Journal of the Acoustical Society of America*, 59(S1):S97–S97, 1976.

[30] M.L. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. Mit Press, 1972.

[31] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. Deep belief networks for phone recognition. In *Nips workshop on deep learning for speech recognition and related applications*, 2009.

[32] Mehryar Mohri. Finite-state transducers in language and speech processing. *Comput. Linguist.*, 23(2):269–311, June 1997.

[33] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted automata in text and speech processing. *CoRR*, abs/cs/0503077, 1996.

[34] Mehryar Mohri, Fernando Pereira, and Michael Riley. *Speech Recognition with Weighted Finite-State Transducers*, pages 559–584. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[35] N. Morgan and H. Bourlard. Continuous speech recognition using multilayer perceptrons with hidden markov models. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 413–416 vol.1, April 1990.

[36] Katia Moskvitch. The machines that learned to listen. *BBC*, Feb 2017.

[37] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[38] Douglas B. Paul and Janet M. Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 357–362, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[39] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[40] Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, and Sanjeev Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks.

[41] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[42] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. 2016.

[43] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880. ACM, 2009.

[44] Martin Russell and Shona DArcy. Challenges for computer recognition of childrens speech. In *Workshop on Speech and Language Technology in Education*, 2007.

[45] Richard Schwartz, Chris Barry, Yen-Lu Chow, Alan Derr, Ming-Whei Feng, Owen Kimball, Francis Kubala, John Makhoul, and Jeffrey Vandegrift. The bbn byblos continuous speech recognition system. In *Proceedings of the workshop on Speech and Natural Language*, pages 94–99. Association for Computational Linguistics, 1989.

[46] Prashanth Gurunath Shivakumar, Alexandros Potamianos, Sungbok Lee, and Shrikanth Narayanan. Improving Speech Recognition for Children using Acoustic Adaptation and Pronunciation Modeling. *Proceedings of Workshop on Child Computer Interaction*, September 2014.

[47] G. W. Stewart. On the early history of the singular value decomposition, 1992.

[48] Mahendra Verma, Dimitrios Prezas, Thomas Russell, Mostafa Sherif, and Reed Thorkildsen. Novel applications of speech processing in at&t network systems products. 69(5):77–86, 1990.

[49] T. K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57, Jan 1968.

[50] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[51] A Waibel, T Hanazawa, G Hinton, K Shikano, and K.J Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

[52] P.D. Wasserman and T. Schwartz. Neural networks, part 1: What are they and why is everybody so interested in them now? *IEEE Expert-Intelligent Systems and their Applications*, 2(4):10–14, 1987.

[53] J. G. Wilpon and C. N. Jacobsen. A study of speech recognition for children and the elderly. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 349–352 vol. 1, May 1996.

[54] Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia, Changsheng Xu, and Qi Tian. Hmm-based audio keyword generation. In *Pacific-Rim Conference on Multimedia*, pages 566–574. Springer, 2004.

[55] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The htk book.

[56] Xiaohui Zhang, Vimal Manohar, Daniel Povey, and Sanjeev Khudanpur. Acoustic data-driven lexicon learning based on a greedy pronunciation selection framework. *arXiv preprint arXiv:1706.03747*, 2017.