PRONUNCIATION MODELING FOR LARGE VOCABULARY SPEECH RECOGNITION

BY

ARTHUR KANTOR

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Doctoral Committee:

Associate Professor Mark Hasegawa-Johnson, Chair
Professor Dan Roth
Associate Professor Margaret Fleck
Assistant Professor Karen Livescu, Toyota Technological Institute at Chicago

Abstract

The large pronunciation variability of words in conversational speech is one of the major causes of low accuracy for automatic speech recognition (ASR). Many pronunciation modeling approaches have been developed to address this problem. Some explicitly manipulate the pronunciation dictionary as well as the set of the units used to define the pronunciations of words. Others model the pronunciation implicitly by using the long duration acoustical context to more accurately classify the spoken pronunciation unit.

This thesis is a study of the relative ability of the acoustic and the pronunciation models to capture pronunciation variability in a nearly state of the art conversational telephone speech recognizer. Several methods are tested, each designed to improve modeling accuracy of the ASR. Some of the experiments result in a lower word error rate, but many do not, apparently because, in different ways, the accuracy gained by one part of the recognizer comes at the expense of accuracy lost or transferred from another part of the recognizer.

Pronunciation variability is modeled with two approaches: from above with explicit pronunciation modeling and from below with implicit pronunciation modeling within the acoustic model. Both approaches make use of long duration context, explicitly by considering long-duration pronunciation units and implicitly by having the acoustic model consider long-duration speech segments.

Some pronunciation models address the pronunciation variability problem by introducing multiple pronunciations per word to cover more variants observed in conversational speech. However, this can potentially increase the confusability between words. This thesis studies the relationship between pronunciation perplexity and the lexical ambiguity. The design of explicit pronunciation models presented here is informed by this study.

Acknowledgments

I would like to thank my advisor, Mark Hasegawa-Johnson for his generosity, encouragement, zen-like patience and invaluable advice on all aspects of my graduate school endeavor.

A lot of ideas (and some of the source code) in this thesis have their roots in the 2006 Workshop organized by John Hopkins Center for Language and Speech Processing. I am very grateful to Fred Jelinek and the workshop's organizers for the opportunity to participate. I am also grateful to the workshop's participants for the valuable discussions. In particular, thanks to Simon King, Özgür Çetin, Chris Bartels and our group's team leader, Karen Livescu.

I also want to thank Jeff Bilmes, for developing the Graphical Models Toolkit, the engine for my speech recognizer, for timely answers to my questions about it, for opening up its source code to me and for hosting me for a week at the University of Washington.

Thanks to the members of the statistical speech technology group and the prosody group at UIUC for stimulating discussions, especially Jennifer Cole, Margaret Fleck, Xiaodan Zhuang and Lae-Hoon Kim.

Some of the images in this thesis are derived from those in Wikipedia. The use of those images is gratefully acknowledged and all the images in this thesis are released under Creative Commons BY-SA license. I also gratefully acknowledge the support from the National Science Foundation grant 07-37624.

I want to thank Alexander Khibnik, for the opportunity to intern at the United Technologies Research Center and for his advice.

Thanks to my family for their encouragement. And, of course, thanks to Ida for giving me a *real* reason to finish.

Table of Contents

Chapter 1

INTRODUCTION

Speech recognition technology has advanced enough to be commercially usable for a variety of specialized tasks, such as small vocabulary and restricted grammar dialog systems, and even dictated speech [1]. However, accurate recognition of spontaneous conversational speech remains elusive. Much of the difficulty can be attributed to pronunciation variability of words. In conversational speech, the pronunciation of a given word depends on the words and phrases around it [2, 3], its syntactic, semantic and prosodic role [4], speaking rate [5, 6], the regional accent [7], the environment [8] and emotional state of the speaker [9, 10].

A typical large vocabulary speech recognizer roughly consists of three parts: the language model which models the relationship between words, the pronunciation model which models the relationship between words and sub-word units, such as phonemes and the acoustic model which models the relationship between sub-word units and some representation of the sound waveform.

In reality, the responsibilities of the language, pronunciation and acoustic models are not disjoint. For example, a pronunciation model commonly includes a dictionary that maps words to tri-phones. Ideally, the pronunciation model would model the pronunciation differences due to co-articulation, rate-of-speech and so on. But it is often convenient to model coarticulation due to neighboring words within the language model by introducing multi-words with specialized pronunciations (`KIND OF` $\Rightarrow$ `KINDA`).

Similarly, it is difficult to exclude the acoustic differences due to co-articulation from the acoustic models. So pronunciation variability is partly modeled by the acoustic models as well. It is difficult to get a word error rate improvement on conversational speech by manipulating only the pronunciation model without corresponding modifications in the acoustic model.

Experiments show that under some circumstances, pronunciation models

are responsible for a significant fraction of the errors in a speech recognizer [11, 12, 13]. Yet, straightforward extensions of the pronunciation models have been only moderately successful on conversational speech, not yielding the word error rate improvements for which pronunciation variability seems responsible.

On the other hand, substantial improvements in accuracy are achieved using acoustic models which consider long duration acoustic context. This long duration context allows the acoustic models to capture coarticulation effects, which is formally the domain of the pronunciation model.

This thesis is a study of the relative ability of the acoustic and the pronunciation models to capture pronunciation variability in a nearly state of the art conversational telephone speech recognizer. Several methods are tested, each designed to improve modeling accuracy of the automatic speech recognition (ASR). Some of the experiments result in lower word error rate, but many do not, apparently because, in different ways, the accuracy gained by one part of the recognizer comes at the expense of accuracy lost or transferred from another part of the recognizer.

Pronunciation variability is modeled with two approaches: from above with explicit pronunciation modeling and from below with implicit pronunciation modeling within the acoustic model[1]. Both approaches make use of long duration context, explicitly by considering long-duration pronunciation units and implicitly by having the acoustic model consider long-duration speech segments.

1.1 Thesis outline

In Chapter 2 we provide a (non-exhaustive) overview of the large amount of existing research on pronunciation modeling. The overview covers both explicit pronunciation modeling in terms of symbolic units, and implicit pronunciation modeling within the acoustic models. We also explore the relationship between the pronunciation perplexity and lexical confusability of the recognizer.

Finally, we discuss the circumstances under which allowing additional context independent pronunciation perplexity is expected to improve recogni-

[1]The names implicit and explicit pronunciation modeling come from [14].

tion, and present a motivation for our explicit pronunciation experiments. To illustrate the question, consider a pronunciation model that provides two alternate phonetic pronunciations for a given word:

$$\begin{aligned} \texttt{ACCOMPANIMENT} &\Rightarrow \texttt{AH K AH M P N AH M AH N T} \\ &\Rightarrow \texttt{AH K AH M P N IY M AH N T} \end{aligned}$$

On one hand, using two pronunciations rather than one covers more observed pronunciation variants of that particular word. But on the other hand, a sub-word acoustic model (IY) will appear in more words on average. So even if a sound is observed that fits a particular acoustic model well, there is more uncertainty about which word could have produced it. Introducing more pronunciations per word also introduces more confusability between different words and recognition quality may suffer. In Chapter 2, we explore the conditions under which such additional pronunciation perplexity raises or lowers the recognizer accuracy.

Because pronunciation variability of conversational speech touches all components of a speech recognizer, we need a realistic system in order to test our pronunciation modeling hypotheses. We build a near state-of-the-art baseline recognizer based on dynamic Bayesian networks (DBNs). Chapter 3 provides the background on DBNs, and describes how they can be used for speech recognition. It also describes the implementation details of the baseline recognizer, the datasets used, and the evaluation strategy.

Chapter 4 describes the selection of the language model (LM) for our recognizers. It provides the background for LM smoothing techniques and describes their interaction with model pruning, which is necessary for large vocabulary conversational speech systems.

In Chapter 5 we describe a general tool for generating phonetic pronunciations from word spellings. It is used to generate pronunciations for words and word fragments which are encountered in our training speech corpus and are missing from existing dictionaries.

Chapters 6 and 7 present the actual explicit and implicit pronunciation experiments. In Chapter 6, we design "corrective" pronunciation units which augment the triphone pronunciation model of the baseline to fix the common recognition mistakes it makes. Informed by the past pronunciation modeling experiments (Chapter 2), we design the new pronunciation units not to

increase pronunciation perplexity.

Due to the size of the training data and computational constraints, we could not compare the traditional and proposed approaches using the absolute word error rate, and instead we had to compare the performance between systems having a constrained model parameter budget. The upshot of these experiments is that while the word error rate did not improve, we were able to reach the same accuracy using a model with fewer parameters.

In Chapter 7 we describe two implict pronunciation modeling experiments, in both of which the acoustic models model longer-duration segments of speech, but in quite different ways. In the first experiment, we first look for a sequence of phonetically similar acoustic observations, and summarize them with a representative observation. The acoustic context of a given observation grows by the number of neighboring observations that were summarized away. This simple approach does improve the recognition accuracy, while reducing the computational cost of training.

The second implicit pronunciation modeling experiment describes low pass filtering the posterior distributions of phone given the observations. The filter window lengths are longer than the context length considered by typical recognizers. This approach is similar but much simpler than some previously published approaches. While, the WER improvement is not reported, the frame classification accuracy improves across almost all phonemes, and not just for the long duration ones.

Chapter 8 at the end of the thesis reviews the key findings, discussing experiments that worked, experiments that should have worked but didn't, possible explanations, and possible future work.

Chapter 2

BACKGROUND

A high-level overview of state of the art of ASR systems is presented in Fig. 2. This thesis focuses on recognition of large vocabulary conversational speech which remains difficult, with the best systems having accuracy of around 14% WER in 2009.

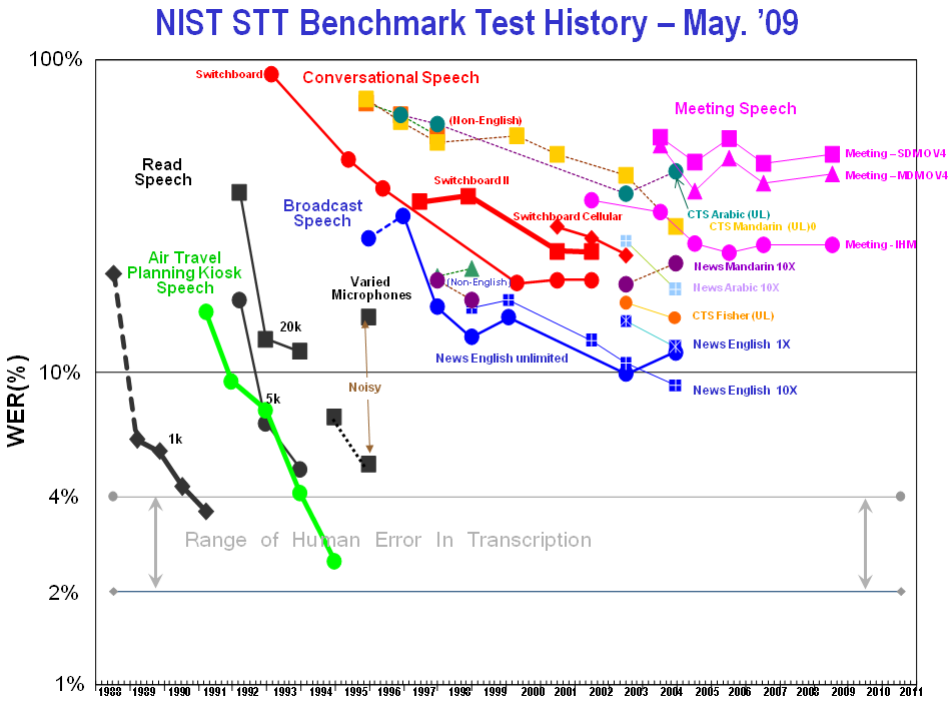

Figure 2.1: WER of top ASR systems for various types of speech participating in the NIST competitions, 1988-2009 [15, 16]. 1k, 5k and 20k refer to vocabulary size, and 1X and 10X refer to real-time and 10 times slower than real time recognition systems. This thesis focuses on conversational English telephone speech, colored red in this chart.

Experiments show that much of the recognition error can be blamed on pronunciation variability. Therefore handling the pronunciation variability well is critical to building an accurate speech recognizer [11]. The next sec-

tion describes the various ways to model pronunciation variability explicitly. Section 2.2 describes the situations where added pronunciation perplexity can be expected to improve the recognizer, and the last section talks about the pronunciation modeling taking place implicitly within the acoustic models.

2.1 Explicit pronunciation modeling

This section describes the most direct family of approaches which we call explicit pronunciation modeling. Explicit pronunciation modeling primarily deals with the selection of the set of sub-word units and the design of the pronunciation dictionary in terms of the sub-word units. This contrasts with the pronunciation variability that is implicitly captured within the acoustic models which is discussed in the next section.

A great deal of research has been done on explicit pronunciation models. In [17], Strik presents a summary of almost 100 articles on this topic. At its simplest, explicit pronunciation modeling consists of selecting a phone-set and constructing a pronunciation dictionary.

This manual pronunciation dictionary can be improved by learning rules mapping canonical base pronunciations to the surface pronunciations actually observed in the training data [18, 19, 20]. Words in frequent phrases sometimes have strong co-articulation effects on each others pronunciation. Explicitly adding such phrases to the pronunciation dictionary along with their idiosyncratic pronunciations often improves the WER [21].

Another kind of explicit pronunciation modeling uses syllable sub-word units. Syllable units are compatible with speech production models and are important in speech perception [22, 23, 24]. Syllables are stable units, having a deletion rate of 1% while phoneme deletion rate is 12% [25]. Syllable models allow lower entropy acoustic models when they are possible. In [25], Greenberg shows that the pronunciation variability of a given phone depends on the phone's position in the syllable. The syllable onsets are likely to be preserved, nuclei are more likely to be substituted but remain vocalic, and codas are more likely to be deleted. The syllable models allow low-entropy observation probability densities for states representing onset phones, while triphone models use the same high-entropy density for phones in onset, nucleus and coda.

Syllable models were proposed in the 70's [26] and a number of syllable-based ASR systems have been built for both small [27] and large vocabulary recognition [28, 29, 27, 30, 31, 32]. They generally perform on par with triphone models. However in all of the published experiments we've come across, the syllable models were context-independent — a major disadvantage against triphone models.

Triphone and syllable models can be further specialized and trained for different rates of speech. In [5], the authors train two sets of triphone models, one each on examples faster and slower than the median duration. The decoder hypothesizes two different pronunciations at each word transition, one using the fast models and one using the slow models. This approach lowers the WER on top of the gains from a multi-word dictionary, even if the total number of parameters in a model is kept constant. Indirect evidence for usefulness of this approach is also found in [33] where prosodic correlates of phone duration are used but the duration itself is not used.

During decoding, the recognizer hypothesizes not only the sequence of acoustic units, but also their duration. The WER improvement in [5] can perhaps be further improved if the fast or slow pronunciation model is determined by the word duration hypothesized by the decoder. For example, the fast pronunciation may be disallowed if the hypothesized duration is longer than the median duration for the word, and the short pronunciation is disallowed if the word ends before its duration reached the median pronunciation. We have not verified the effectiveness of this change.

It is sometimes difficult to tell what is actually modeled by explicit pronunciation models. For example, even though triphones (and quinphones) are explicit pronunciation models, it is uncertain what the triphone actually learns to model through data-driven training. The phoneme boundaries are not explicitly specified, and the triphone may simply represent some unknown sub-word unit that sounds like other sub-word units in other words.

2.2 Pronunciation variability's contribution to WER

The WER is a complicated function of the way language, pronunciation and acoustic models interact. Traditionally each of the models was designed independently of the others, but recently work has been done to design the

models so they work well together. A language model quality measure has been developed that takes into account a particular acoustic model used in a recognizer [34]. Approaches have also been developed to jointly optimize the language and acoustic models [35, 36] and to optimize the pronunciation model in context of a given acoustic model [37].

In this section we look at how the pronunciation model interacts with the acoustic model.

2.2.1 Recognition on synthesized data

In a set of illuminating experiments in [12], McAllaster et. al. explore the error contributions from the pronunciation and acoustic models. They present evidence that there is a significant mismatch between the acoustic and pronunciation models and data.

The authors train two acoustic triphone models: $p_g(o|q)$ and $p_r(o|q)$ on two disjoint sets of Switchboard conversational speech data. In these distributions, o is the acoustic feature observation and q is the triphone cluster that could have generated the observation. The authors use $p_g(o|q)$ to generate synthetic acoustic feature observations O from either word or phonetic transcriptions of a part of the Switchboard corpus. Then they perform recognition of the synthetic data with a recognizer which uses the $p_r(o|q)$ acoustic model. See Figure 2.2 for an overview of the process.

Synthesizing O by $p_g(o|q)$ guarantees that the true distribution of the data $p_g(o|q)$ is included in the family of distributions considered during the training of the recognizer. In this sense, there is no mismatch between the acoustic model and the data. The only difference between $p_g(o|q)$ and $p_r(o|q)$ used by the recognizer is due to the differences in the training speech data.

Performing ASR with the recognizer on real (non-synthesized) speech O_r is the baseline, with WER of 49%. Running the same recognizer on the synthetic data O_w synthesized from word transcriptions using the canonical dictionary pronunciations, the WER drops from 49% to 11%. This shows that by improving pronunciation and acoustic models, it is possible to reduce WER by 38% absolute, even if we keep the language model fixed.

In the second experiment, the authors generate synthetic O_p data from manual phonetic transcriptions instead of using the word transcriptions with

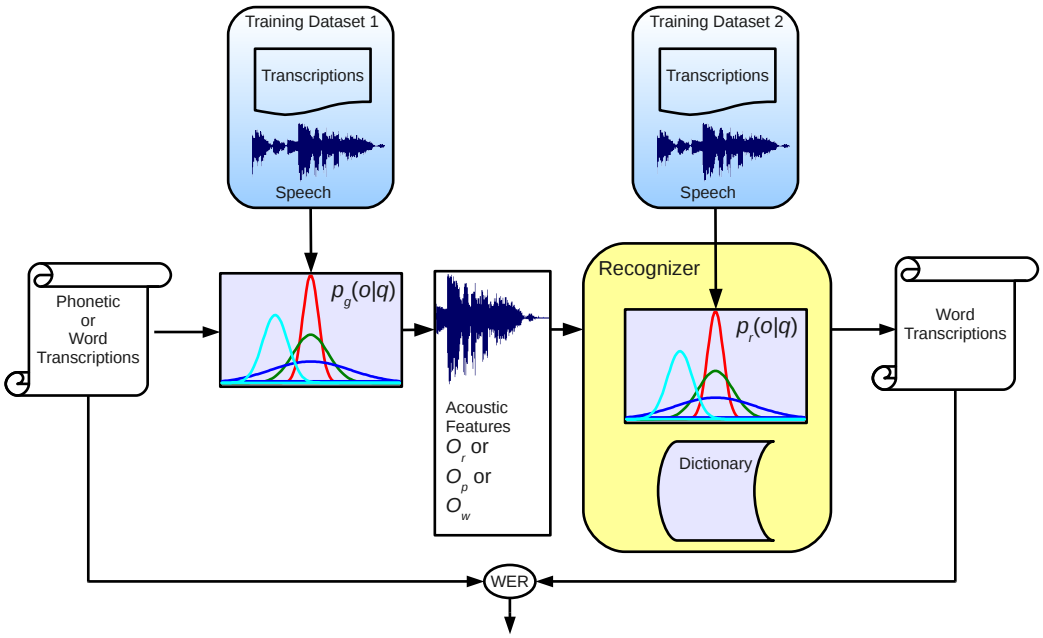

Figure 2.2: An overview of the Mcallaster's et. al. experiment of ASR on synthesized data [12]. The Acoustic features can be either real speech O_r or they can be synthesized by acoustic model $p_g(o|q)$ from manual phonetic transcriptions O_p, or from word-level transcriptions and a canonical pronunciation dictionary O_w.

canonical pronunciations. Running the same recognizer on O_p observations lowers the WER from 49% to 44%, just 5% better than the WER on real (not synthesized) data. This suggests that poor pronunciation modeling is responsible for most of the error *when the acoustic model* $p_r(o|q)$ *matches the observations well*. To verify this, the authors 'cheat' by adding all the non-canonical pronunciations of words that occur in the test set into the recognizer's pronunciation dictionary, and performing the recognition on O_p again. The WER further improves from 44% to 34%.

However, if the recognizer with the augmented dictionary is run on real data O_r, the WER increases from 49% to 61%.

To summarize, this experiment shows two things. First, the recognition of the conversational speech can be improved greatly by developing better pronunciation and acoustic models (49% to 11% WER). Second, this experiment shows that given a perfect acoustic model, growing the pronunciation model so it can choose among more pronunciations per word is helpful (44% to 34% WER), if the extra pronunciations are chosen well. However, if we

use an acoustic model with a high model/data mismatch, growing the pronunciation model is harmful (49% to 61% WER), even if it is grown in an optimal way, with only the pronunciations known to be in the test set.

In the extreme case, the recognition can be performed on a (presumably perfect) human-transcribed phonetic transcription. The acoustic model is bypassed entirely, and cannot contribute any errors to the recognition. One such experiment is described in [38], where a word in the lexicon is defined not as a string of phonemes, but as parallel strings of articulatory features. In this experiment, a word is defined via three strings of discretized positions of the lips, tongue and glottis. The decoder then predicts a sequence of words given the sequences of articulatory strings. Pronunciation perplexity is introduced by allowing asynchrony between articulatory features. For example, a word may be pronounced canonically, and it may also be pronounced with the lips at time position t taking on the value of lips at either time position $t-1$ or $t+1$, while the tongue and glottis take on only the canonical values. The WER is not reported, but the word classification error drops from 64.8% for the recognizer without asynchrony to 37.3% for the recognizer with asynchrony.

In another experiment [39], the authors create a 'surficial pronunciation model' in which they grow the lexicon with pronunciation variants seen in Switchboard training data, similar to the experiments by Mcallaster et. al.[12] described above. Using manual phonetic transcriptions as observations, they report a WER of 18.61% using the original lexicon, and 12.63% using lexicon enlarged with pronunciations from the training data.

In the cases where the acoustic model/data mismatch is non-existent, allowing more pronunciation perplexity improves the WER dramatically. In cases where the acoustic model/data mismatch is high, such as in conversational speech, the picture is less clear. In theses cases, we know of no experiments which lower the WER simply by replacing a pronunciation model with one that allows more pronunciation variants. It is sometimes possible to obtain WER improvements through introducing a small number of pronunciation variants combined with acoustic model retraining.

The next section discusses explicit pronunciation models in which pronunciation perplexity is allowed to grow and describes the conditions under which additional pronunciation perplexity is beneficial.

2.2.2 When is pronunciation perplexity helpful

The synthesized data recognition experiment from the previous section shows that permitting more pronunciation variability through explicit pronunciation is beneficial only if the acoustic model can accurately discriminate among the sub-word units. The meaning of "accurately" depends on the recognition task and is difficult to characterize. Nevertheless, we can still describe the conditions under which additional pronunciation perplexity can be expected to improve the WER.

A survey of the literature summarized in Figure 2.3 shows that increasing pronunciation perplexity is helpful in systems with high starting WER and harmful on the challenging tasks where the starting WER is low. The WER change caused by increasing the perplexity of the pronunciation model is a nearly linear function of the starting WER (correlation $R^2 = .80$). The trend appears to be true across different kinds of speech data and ASR systems. In particular, this trend is true for a variety of pronunciation models, some of which are described below.

In the JHU/CSLP 1996 workshop, the participants experimented with using decision tree (DT) pronunciation models for n-best rescoring on Switchboard conversational speech corpus [19, 6]. There was a slight improvement when the canonical pronunciations were replaced by pronunciations automatically learned from the training data (static dictionary), however allowing multiple pronunciations via the dynamic decision-tree-based pronunciation graphs performed worse than the static dictionary.

In the JHU/CSLP 2006 workshop, the recognizer was built using articulatory features (tongue, lips and glottis) instead of phonemes as pronunciation units, and pronunciation variability was modeled as asynchrony between the articulatory features [40, 43]. On Svitchboard[44] data, models with allowed asynchrony (and thus more pronunciations per word) underperformed the same models with no asynchrony, although a similar approach was beneficial when performing recognition on careful phonetic transcriptions [38].

Allowing sub-word state skips is another way to grow the pronunciation model in a context free way. Experiments show that this also hurts the WER on MALACH [29] and Switchboard [6] datasets.

It is also possible to reduce the pronunciation perplexity of an existing recognizer. For example, shrinking an existing human-made multi-pronunciation

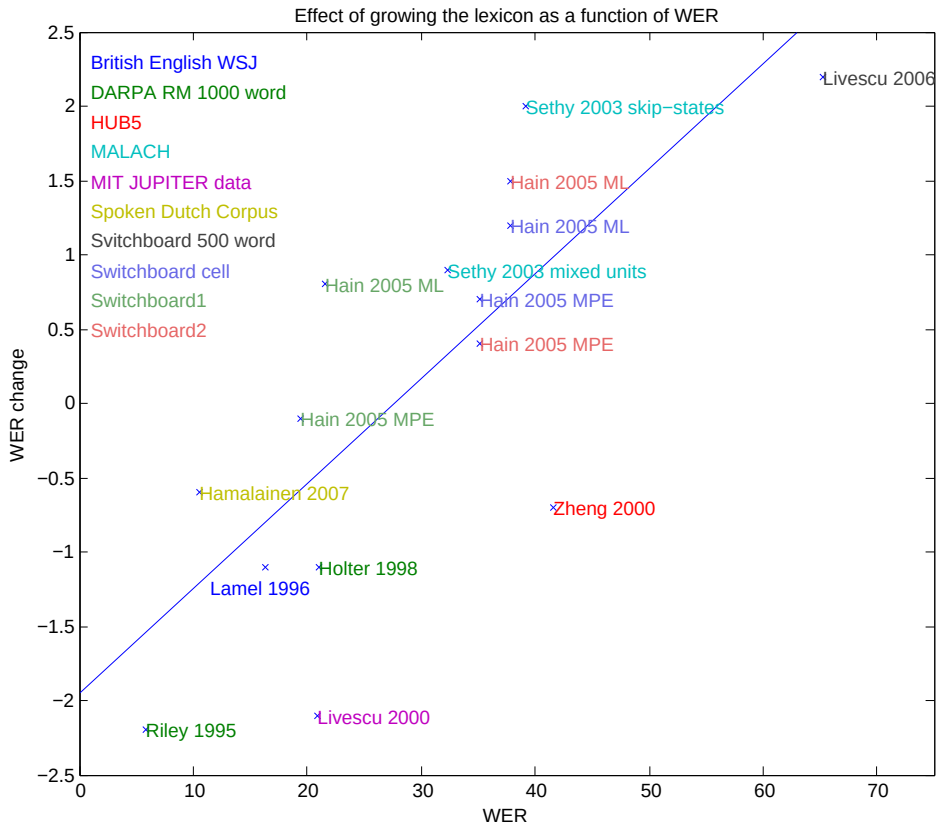

Figure 2.3: Results reported in literature showing changes in WER as the number of pronunciations per word is grown at various starting WERs. The data points are: Livescu 2006[40], Sethy 2003[29], Hain 2005[41], Zheng 2000[5], Fossler-Lussier 1999[6], Hamalainen 2007[31] Holter 1998[42] and Riley 1995[18]. Some reports describe multiple datasets and multiple ASR systems. The correlation coefficient is $R^2 = .80$, and without the Zheng 2000 data point it is $R^2 = .90$.

dictionary shows improvement on Switchboard data [6] and on other datasets [41].

In Figure 2.3, the experiments marked with 'Hain 2005 ML' all use the same recognizer trained with the maximum likelihood criterion. Similarly, 'Hain 2005 MPE' experiments all use the same recognizer trained with the minimum phone error criterion [41]. Presumably, the minimum phone error recognizer has a better acoustic model (lower model/data mismatch), and so growing the pronunciation perplexity is less harmful for the MPE than it is for ML, even though the WER difference between MPE and ML is not very large. The lowest starting WER point even benefits slightly from growing the pronunciation perplexity.

Systems with low WER benefit from more pronunciations per word. Allowing alternate multi-path syllable models [31] has yielded a slight improvement on the Spoken Dutch Corpus (described in [45]) and multiple phonetic pronunciations improved the already low WER even further on the DARPA RM and NAB tasks [18].

The trend in Figure 2.3 is not absolute. We found at least two systems with a low starting WER where increasing pronunciation perplexity helped when combined with acoustic retraining. In [5], two sets of triphone models are trained, one for fast speech and one for slow speech, and the decoder can choose at the word boundary to use one or the other set of triphones for the following word. The baseline has only a single set of triphones trained on both fast and slow speech. Pronunciation perplexity grows in the two phone-sets system, and yet this system outperforms the baseline, lowering the WER from 41.6% to 40.9%. It appears that a WER improvement is possible despite the growth in perplexity when the hypothesized strings of acoustic models are substantially different, even at high WERs. Even in this case, having more specialized acoustic models without adding pronunciation perplexity helps more. In a similar system, with 3 triphone sets, the third set being used exclusively for multi-words, the WER drops by 0.9%.

There are some practical implications from the above experiments. First, when comparing or designing new pronunciation models, the pronunciation perplexity should be controlled. Second, for high WER, the biggest improvements can be had from improving the acoustic model, either directly by improving the front-end, or by limiting the perplexity of the pronunciation model through the use of contextual constraints.

2.2.3 Context

The goal of low pronunciation perplexity should be reconciled with the huge pronunciation variability that occurs in conversational speech. See [25] for 80 different phonetic pronunciations of the word `AND` encountered in the Switchboard corpus. The distribution of pronunciations of `AND` is close to Zipf's law, and has an entropy of 4.6 bits. Other common words have similar distributions of pronunciations. Section 2.2.2 shows that for most conversational speech systems, increasing the pronunciation model entropy by 4.6

bits will degrade WER. In order to model pronunciation variability, almost all pronunciation perplexity must be disambiguated by context.

Short-duration context has been successfully incorporated into the acoustic models. For example, triphone models substantially improve the WER. Through context, they allow pronunciation variability while keeping the pronunciation perplexity at the same level as for monophone models.

Adaptation techniques which remove variability due to speaker, channel and noise differences can be considered as using global context to select (adapt) the acoustic model [46]. These adaptation techniques include vocal-tract length normalization, MLLR, spectral and cepstral mean subtraction and cepstral variance normalization. They have been critical for lowering the WER for conversational speech[41] and they also do not increase pronunciation perplexity.

The pronunciation and acoustic models sometimes address the same issues, and sometimes jointly ignore others. For example, the traditional triphones already do a good job modeling phone substitutions and deletions [47]. Tandem models show an additional improvement by representing short-duration context. And yet these same issues are addressed by skip-state pronunciation models [29] and some versions of articulatory-feature asynchrony models [43].

On the other hand, the pronunciation variability due to the word's long-term context or prosodic role is not captured by either the standard acoustic or pronunciation models. Using multi-words (e.g. `GOING TO` → `GONNA`) in the language model addresses the problem for the worst offenders by at least allowing the highly coarticulated pronunciation [48], but it also undesirably increases pronunciation model (or language model) perplexity. Conditioning acoustic models on rate of speech [5] also shows improvement but again increases pronunciation perplexity. Systems where the acoustic and language models are conditioned on prosodic factors show statistically significant improvements in careful experiments [49, 33].

In some circumstances the standard models are incapable of learning the difference between two different words because they have similar or identical phonetic transcriptions. For example word pairs like `KNOW` and `NO` have identical phonetic (and syllabic) transcriptions, and the recognizer must rely on the language model to choose correctly. Even worse, words like `RIGHT` and `I` have different phonetic transcriptions, but in conversational speech `RIGHT` tends to be pronounced as `I`, and so the pronunciation and acoustic models

may actually contribute to the confusion between the words. Both these word pairs were among the most confused in the JHU WS 06 experiments [40]. Other frequently confused word pairs were also generally acoustically similar monosyllabic words.

A possible solution is to model monosyllabic words with units distinct from general-purpose syllables or phones. For example, the expected duration of the word `KNOW` is 80ms shorter than of the word `NO` [27]. This distinction can be captured by a special whole-word model just for these words. In [27], a 2.4% absolute WER improvement is reported if mono-syllabic words and syllables are modeled separately.

While none of the approaches described in this section so far increased pronunciation perplexity, the majority of approaches are some combination of increased context and increased pronunciation perplexity. In such cases it is difficult to tell what affects the WER more and so such experiments do not belong in plot on the Figure 2.3.

To summarize the two sections above, in designing conversational speech recognizers, it is critical to handle a lot of pronunciation variability without introducing a lot of perplexity into the pronunciation model. This can be done by providing many pronunciations for the same thing, but conditioning them on wider context so that only one pronunciation is allowed to be active at the same time. The best results are achieved when the acoustic models are retrained for a particular pronunciation model, or better yet, the acoustic and the pronunciation models are jointly trained. The next section describes approaches where the pronunciation is modeled implicitly within the acoustic model.

2.3 Pronunciation modeling in the acoustic model

Hidden Markov Models (HMMs) commonly used for speech recognition assume that the observations are conditionally independent given the hidden state. However this assumption does not hold for speech signals. The current observation depends on the hidden state and also on the nearby observations. To put it another way, the identity of a phoneme (or even a context-dependent phoneme) depends not just on the instantaneous spectrum but also on long term evolution of the spectrum. For this reason, most LVCSR systems in-

clude long-term acoustic context in the acoustic model in some way. This section, describes some of the ways in which long-term acoustic context is incorporated into the acoustic model.

With the help of acoustic context, it is easier to disambiguate the pronunciation variability present in conversational speech. Providing additional acoustic context to the acoustic model does not increase the pronunciation perplexity as described in Section 2.2.2, so the approaches here should be useful even in recognizers with a high starting WER. Another advantage is that the acoustic models are optimized jointly with the part of the pronunciation model responsible for alternate pronunciations - they are the same thing. A disadvantage of this approach is that the pronunciation modeling in the acoustic models is treated as black box. It is difficult to analyze the acoustic model to see just what kind of pronunciation variability is captured. It is difficult to even separate pronunciation modeling from the acoustic modeling of individual speakers or background noise or channel effects. Nevertheless, ASR systems have been incorporating progressively more acoustic context in more complicated ways into the acoustic models. In this section we describe some of the approaches which make use of progressively longer context, some of which are also used in our experiments.

2.3.1 Using context in acoustic models

Many approaches can be described within the same framework. A string of observation frames is used to create an intermediate representation that better captures the dependencies between neighboring frames, and the intermediate representation is used in the recognizer both for decoding and to train the acoustic model. Some function

$$f : O^{2n+1} \Rightarrow Y$$

is applied to a vector of $2n+1$ frames $[o_{i-n}, ..., o_i, ...o_{i+n}]$ centered on frame i, with each frame $o \in O$, and creates the intermediate representation $y_j \in Y$. For many choices of f, $j = i$ so one feature vector y_i is generated for every input frame o_i. We now describe some of the implicit pronunciation modeling techniques in the order of increasing complexity.

Augmenting velocity and acceleration

A frequently used transformation f is

$$f_{\Delta}([o_{i-n}, ..., o_i, ...o_{i+n}]) = [o_i, \Delta_i, \Delta\Delta_i]$$

which simply concatenates o_i with the estimates of the slope or velocity (Δ_i's) and curvature or acceleration ($\Delta\Delta_i$'s) of o_i as a function of time. The velocity is typically computed with a regression function using two neighboring frames on each side ($n = 2$).

Hybrid models

Another approach is the hybrid approach [50]. It first trains a multi-layer perceptron (MLP) to perform phoneme classification at the frame level, and uses the normalized activations as the probability distribution of the phoneme given the frame and its neighbors. The MLP typically uses 9 input frames ($n = 4$). If we let

$$x = [o_{i-n}, ..., o_i, ...o_{i+n}]$$

the transformation is defined as

$$f_{hybrid}(x) = \frac{MLP(f_{\Delta}(x))}{\sum MLP(f_{\Delta}(x))}$$

In this case f_{hybrid} takes place of the mixture-of-Gaussians acoustic model and the phone posteriors are used directly as "virtual evidence".

Tandem models

Yet another approach is the 'tandem' model introduced by Sharma, Hermansky and Ellis [51], which combines f_{hybrid} and f_{Δ} transformations:

$$f_{tandem}(x) = [f_{\Delta}(x), PCA(\log(MLP(f_{\Delta}(x))))] \tag{2.1}$$

Figure 3.7 shows a block diagram of a particular tandem model. In this case, a mixture-of-Gaussians acoustic model is trained on $f_{tandem}(x)$ outputs. The log() 'undoes' the the non-linearity at the MLP's output layer. This makes the posterior phone distributions more Gaussian so that the mixture-

of-Gaussians fits them better. Principal Component Analysis is then applied to $\log(MLP(f_{\Delta}(x)))$ to decorrelate the resulting features. The resulting feature vectors are truncated (projected onto the largest principal components) to reduce the dimensionality of the vectors while retaining as much as of the total variance as possible. The PCA followed by truncation is denoted as $PCA()$ in Equation 2.1.

Recognizers using the tandem observation features have been used successfully to lower the WER in large vocabulary speech recognition [52, 53].

TRAPs

In [54, 55], TempoRAl Patterns (TRAPs) are introduced for speech recognition. A TRAP feature is constructed for every critical band in the speech frequency range by training a MLP classifier which uses as input the critical band log-energy trajectory over a duration of 101 frames (1 second of speech). The TRAP targets are phones. The outputs of the TRAP MLPs are fed into a combining MLP $MLP_{combine}()$, the outputs of which are concatenated with the 9 frame MLP as in the Tandem model:

$$f_{TRAP}(x_{50}) = [f_{tandem}(x_9), PCA(\log(MLP_{combine}(x_{50})))]$$

where $x_n = [o_{i-n}, ..., o_i, ..., o_{i+n}]$.

In [56], the authors show that combining TRAPs with a tandem system improve recognition on conversational speech against a high-quality discriminatively trained baseline (37.7% $\Rightarrow$ 34.4%). In [57], the authors show that a two-layer MLP using TRAP MLPs as input and first hidden layer, and $MLP_{combine}()$ as a second and output layers outperform a single hidden layer MLP which uses critical band log-energies as inputs and a comparable number of parameters.

fMPE

In the fMPE approach [58], a large number ($\sim$100k) of Gaussians are extracted from an existing mixture-of-Gaussians acoustic model, and the likelihood of each frame in x_n is calculated against each Gaussian. The authors use $x_n = x_9$. The 19 posterior vectors are partitioned into 7 groups, the vectors in each group are averaged, and the group average vectors are concatenated,

yielding a vector h with 700,000 elements. The feature transformation for fMPE is defined as

$$f_{fMPE}(x_9) = o_i + Mh$$

where M is a fat matrix trained with the on-line perceptron algorithm with the minimum phone error (MPE) objective. The transformed features are then used to do a ML update to the acoustic models, and the whole procedure can be iterated. This approach also gives significant WER improvements on conversational speech against a state-of-the-art baseline. On the NIST CTS task with a speaker independent system, the improvement was 27.9% $\Rightarrow$ 26.9% for a system using fMPE+MPE against a system using only MPE training.

We should also mention that the feature transformation function f need not create an output feature vector for every input frame, and the n in x_n can vary throughout the utterance. These kinds of transformation functions are closer to the segmental models described in [59].

Explicit and implicit pronunciation modeling can be (and typically is) used together in a recognizer. For example, a multi-word dictionary with a triphone acoustic model can use TRAP features. In the following chapters we present experiments with both kinds of pronunciation modeling.

Chapter 3

DYNAMIC BAYESIAN NETWORKS FOR A VERY LARGE SPEECH DATABASE

This chapter provides an overview of dynamic Bayesian networks in Section 3.1, and describes how they can be used for continuous speech recognition in Section 3.2. Section 3.3 describes the implementation of the baseline speech recognizer used in our main experiments and justifies some of its design choices.

Many aspects of the baseline system are tuned and analyzed, partly to make it of sufficient quality so that improvements against it would be relevant, and partly to compare the relative benefit of the techniques proposed in this thesis against the benefit of well known techniques.

3.1 Background

Many successful statistical ASR systems trace their beginnings to the work of Jelinek and others at IBM [60]. The goal of a statistical ASR system is to come up with a most likely sequence of K words $W_{1...K}$ that may have generated the observed sequence of T acoustic observation vectors $O_{1...T}$. So we are looking for

$$W^*_{1...K} = \arg\max_{K, W_{1...K}} p(W_{1...K}|O_{1...T})$$

or equivalently, because $O_{1...T}$ is fixed,

$$W^*_{1...K} = \arg\max_{K, W_{1...K}} p(O_{1...T}, W_{1...K})$$

Estimating or even representing the true distribution $p(O_{1...T}, W_{1...K})$ without any further assumptions is not practical because of the huge number of possible word and observation sequences. To solve this, the large state space of the two random variables (RVs) $O_{1...T}$ and $W_{1...K}$ is represented as an

equivalent marginal of a joint distribution of a set of n RVs $\mathcal{X} = X_1, ..., X_n$, and some of these RVs are made conditionally independent of each other.

This conditional independence allows us to represent the joint distribution $p(O_{1...T}, W_{1...K})$ compactly. For example, if $\mathcal{X}$ is a distribution that can be represented by 6 random variables $X_1, ..., X_6$ each with cardinality 2, the joint distribution is $p(X_1, ..., X_6)$ requires $2^6 - 1 = 63$ parameters. If some of the RVs are conditionally independent in such a way so we can write

$$p(X_1, ..., X_6) = p(X_1)p(X_2)p(X_3|X_1, X_2)p(X_4|X_3)p(X_5|X_3)p(X_6|X_4, X_5) \tag{3.1}$$

we need only $1+1+4+2+2+4 = 14$ parameters to describe the distribution.

Conditional independence among RVs also allows us to estimate $p(O_{1...T}, W_{1...K})$ from a limited amount of training data. Finally it allows us to do probabilistic inference over the RVs in $\mathcal{X}$: given some set of observed evidence RVs $\mathcal{O} \subseteq \mathcal{X}$, and some query RVs $\mathcal{Q} \subseteq \mathcal{X}$ in which we are interested, we can efficiently compute the distribution $p(\mathcal{Q}|\mathcal{O})$.

We can draw Eqn. 3.1 as a graph, where every RV is a node and the set of its inbound edges specifies a conditional probability of the node given its parents, as in Figure 3.1. Such a graph is called a *Bayesian network*.

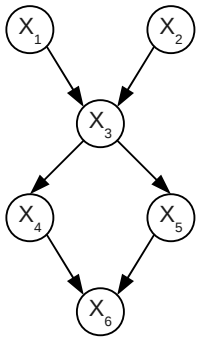

Figure 3.1: The Bayesian network for the example distribution in Eqn. 3.1

We can use *dynamic Bayesian networks* (DBN) to represent a set of variables as they evolve over time. A DBN is a sequence of Bayesian networks $\mathcal{X}_1, ...\mathcal{X}_T$, each with identical graph structure and identical conditional probabilities. Additionally, RVs in the ith DBN $\mathcal{X}_i$ can have parents in preceding DBNs $\mathcal{X}_j, j < i$. If the RVs in $\mathcal{X}_i$ only have parents in $\mathcal{X}_{i-k}, k = 0...n$, this is called a n-order DBN, analogous to n-order Markov chains. A DBN can

grow as necessary to cover observation sequences of different lengths.

In general, to solve the inference problem $p(\mathcal{Q}|\mathcal{O})$, we fix the $\mathcal{O}$ nodes to their observed values $\mathbf{o}$, and marginalize over the non-query RVs:

$$p(\mathcal{Q}|\mathcal{O}=\mathbf{o}) = \frac{\sum_{X\in\mathcal{X}\setminus\mathcal{Q}\cup\mathcal{O}} p(\mathcal{X})}{\sum_{X\in\mathcal{X}\setminus\mathcal{O}} p(\mathcal{X})} \tag{3.2}$$

The order in which we marginalize away the RVs $X \in \mathcal{X} \setminus \mathcal{Q} \cup \mathcal{O}$ in the numerator and $X \in \mathcal{X} \setminus \mathcal{Q}$ in the denominator is important. Through elimination, it is possible to create a partly marginalized distribution which is specified with many more parameters than the original non-marginalized distribution. The increase in the number of parameters of the partially marginalized distribution is exponential in the number of parents, children and parents of children of the RV being eliminated. Picking the variable elimination order that minimizes the sizes of the partly marginalized distributions on the way to $p(\mathcal{Q}|\mathcal{O})$ is an NP complete problem.

The numerator and denominator distributions of Eqn. 3.2 can be efficiently computed using the junction tree algorithm [61] if the joint distribution is represented as a DBN. The same algorithm can also calculate the single most likely assignment to $\mathcal{Q}$ instead of $\mathcal{Q}$'s marginal distribution. As a sub-problem, the junction tree algorithm also must select a good variable elimination order. On the DBNs used for speech recognition, this is an important problem, where a difference between a good and a bad elimination order can mean a 1000-fold difference in memory requirements and compute time.

The problem of learning the distribution $p(\mathcal{X})$ from training data can be solved with the expectation-maximization (EM) algorithm [62, 63]. The EM algorithm is an iterative meta-algorithm, designed to find the model parameters which maximize the likelihood (ML) of the observed RVs in the presence of hidden RVs.

The main idea is to compute the model parameters which maximize the log-likelihood of the data with the hidden RVs marginalized. The expectation of the log-likelihood is taken over the hidden RVs given the observed RVs. The expectation uses an existing estimate of model parameters. So if $\mathcal{Z} = \mathcal{X} \setminus \mathcal{O}$ is the set of the hidden RVs, and $\lambda_{(i)}$ is the estimate of the model parameters

at iteration i, the $\lambda_{(i+1)}$ parameters are estimated as

$$\lambda_{(i+1)} = \arg\max_{\lambda} E_{\mathcal{Z}|\mathcal{O};\lambda_{(i)}}[\log p(\mathcal{X};\lambda)] \tag{3.3}$$

Eqn. 3.3 is guaranteed not to decrease the log-likelihood of the observed data $\log p(\mathcal{O};\lambda_{(i)})$ at every iteration i but does not guarantee that the global maximum will be reached. The EM update equations have been derived for discrete distributions, for mixture of Gaussians models, and many other kinds of distributions.

A special case of the DBN is the hidden Markov model (Figure 3.2), for which the inference and training algorithms were developed before the algorithms on general DBNs, and which has traditionally been used for statistical speech recognition[64]. The junction tree algorithm reduces to the viterbi and the forward-backward algorithms, and the model is estimated with the Baum-Welch algorithm [65] which is an instance of the EM meta-algorithm.

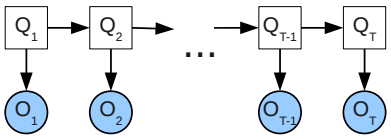

Figure 3.2: The hidden Markov model is a special case of the DBN. It consists only of the (shaded) observed and the query nodes. As any DBN, the template can be unrolled an arbitrary T times.

Probabilistic inference and EM parameter estimation on DBNs consisting of discrete random variables and continuous random variables with certain conditional distributions is implemented in the Graphical Models Toolkit (GMTK) [66]. GMTK is a general DBN toolkit, but it is designed to work efficiently with the large DBNs used in speech recognition.

3.2 Continuous speech recognition with DBNs

DBNs can handle sequences of unknown and observed random variables and they can construct the joint distribution by incorporating information from a variety of external sources. This makes them well suited for speech recognition, and they have been used for this purpose in previous work.

The different sources of information going into the recognizer typically include at least three parts: The first part is the language model, which concerns itself with the distribution over possible sequences of words $p(W_{1...K}, K)$. The second part is the pronunciation model, which maps a word sequence into a sequence of pronunciation units, such as context dependent phonemes and the third part is an acoustic model, which is a continuous distribution over the acoustics given a sub-word unit. Each of the three models (or distributions) have very different characteristics and are often trained with different techniques from different sources information.

In [46] Jeff Bilmes shows how all three speech recognizer components can be concisely represented within the DBN framework: mixtures of Gaussians for the acoustic models, decision trees for context dependent pronunciation models, as well as interpolated and backed-off n-gram language models (see Chapter 4). Besides the three main speech recognizer components, DBNs can represent other techniques commonly used in speech recognition, such as principle component analysis, acoustic classifier combination and some acoustic adaptation techniques.

In [67], Geoff Zweig gives a thorough overview of inference and learning with DBNs, and also uses DBNs for large vocabulary isolated word classification. He uses the additional flexibility of DBNs to reach a higher classification accuracy over HMMs.

In this section we describe in detail the DBN for our baseline continuous speech recognizer. It is derived from the recognizer used in JHU 2006 speech and language processing workshop [40].

3.2.1 Refining the state space

During training and recognition, RVs represent the complete state of the language, pronunciation and acoustic models for every observation frame. The state for each frame contains the word, sub-word unit, and the sub-word unit substate. Because we are using only word-internal context, the addition of shorter-duration units to the state is a refinement of the state space described by the larger units. The RVs representing this pattern are shown in Figure 3.3. Variants of this pattern were previously described in Chapter 6 of [67] and also in [40].

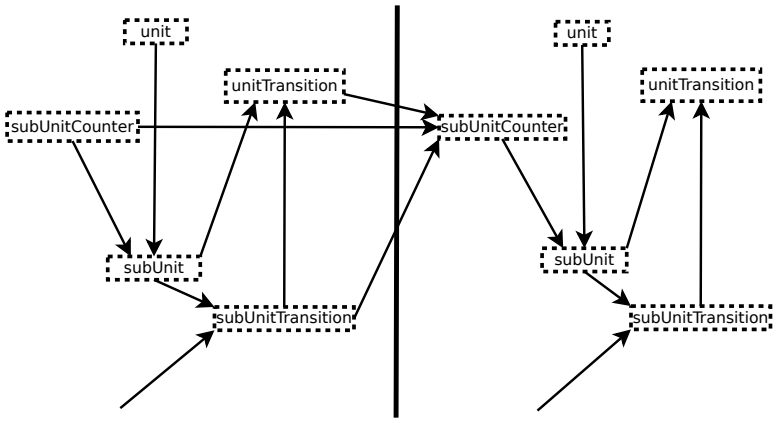

Figure 3.3: The RVs in two neighboring frames $t-1$ and t which participate in the state refinement pattern. Dashed node boundaries mean that the RVs are deterministic given their parents, and square node boundaries mean that the RVs are discrete.

All the RVs in Figure 3.3 are deterministic given its parents. The $subUnitCounter_t$ keeps track of the position within the unit in terms of sub-units. It is a counter running from 1 to the number of sub-units in in the unit. Its cardinality is the longest sub-unit string for any unit. The degenerate probability distribution

$$p(subUnitCounter_t|subUnitCounter_{t-1}, subUnitTransition_{t-1}, unitTransition_{t-1})$$

is defined by applying one of the following rules to $subUnitCounter_t$, in order of preference.

1. Reset to 0 whenever there was a unit transition in the previous frame ($unitTransition_{t-1}$ is true).
2. Increment if $subUnitTransition_{t-1}$ is true.
3. Set equal to $subUnitCounter_{t-1}$ otherwise.

The distribution $p(subUnit_t|subUnitCounter_t, unit_t)$ is the definition of the unit in terms of sub-units, which comes from some external information source.

The RV $unitTransition_t$ is a binary variable which becomes true when triggered by some condition of sub-units. For example when each unit ends in a special `end-of-unit` sub-unit, we can make $unitTransition_t$ true when both the $subUnitTransition_t$ is true and the $subUnit_t$ is set to `end-of-unit`.

This simpler approach works when it is acceptable to have a special `end-of-unit` sub-unit take up an extra observation frame. This is the case for relatively long units, such as utterances or words. When we cannot have an observation-emitting `end-of-unit` sub-unit , we can have a "last state" table for every unit, so that a $unitTransition_t$ requires a $subUnitTransition_t$ and also for $subUnit_t$ to be in its last state. This structure is necessary for short units, such as phones in terms of sub-phone states. $unitTransition_t$'s relationship to $unit_t$ is analogous to $subUnitTransition_t$'s relationship to $subUnit_t$. The former is an example of the structure with `end-of-unit` sub-unit, while the later is an example of using a "last state" table.

3.2.2 The decoding DBN

The decoding DBN of our triphone baseline ASR system is shown in Figure 3.4.

The center portion of the graph represents a typical non-edge frame t. For each utterance consisting of T frames, the center portion of the graph is replicated (unrolled) $T - 2$ times so the DBN spans the duration of the utterance.

Consistent with the previous figures, shaded nodes are observed, and unshaded nodes are hidden. Circles denote continuous RVs and squares denote discrete RVs. A node with a dashed frame is deterministic, in the sense that its conditional entropy given its parents is 0. A dashed incoming edge into a node is used to switch the node's distribution conditional on the node's remaining parents.

All our utterance transcriptions begin and end with the special `start-of-utterance` and `end-of-utterance` words. The phonetic word pronunciations also end in the special end of word `EOW` marker. Therefore, the first and last frames of the DBN have more observed states to define the boundary conditions.

$word_t$, $prevWord_t$, and $wordTransition_t$ encode the trigram language model described in Section 3.3.4. If $wordTransition_{t-1} = 1$, the smoothed trigram language model

$$p(word_t|word_{t-1}, prevWord_{t-1})$$

is used. Otherwise, $word_t$ is deterministically set to the value of $word_{t-1}$.

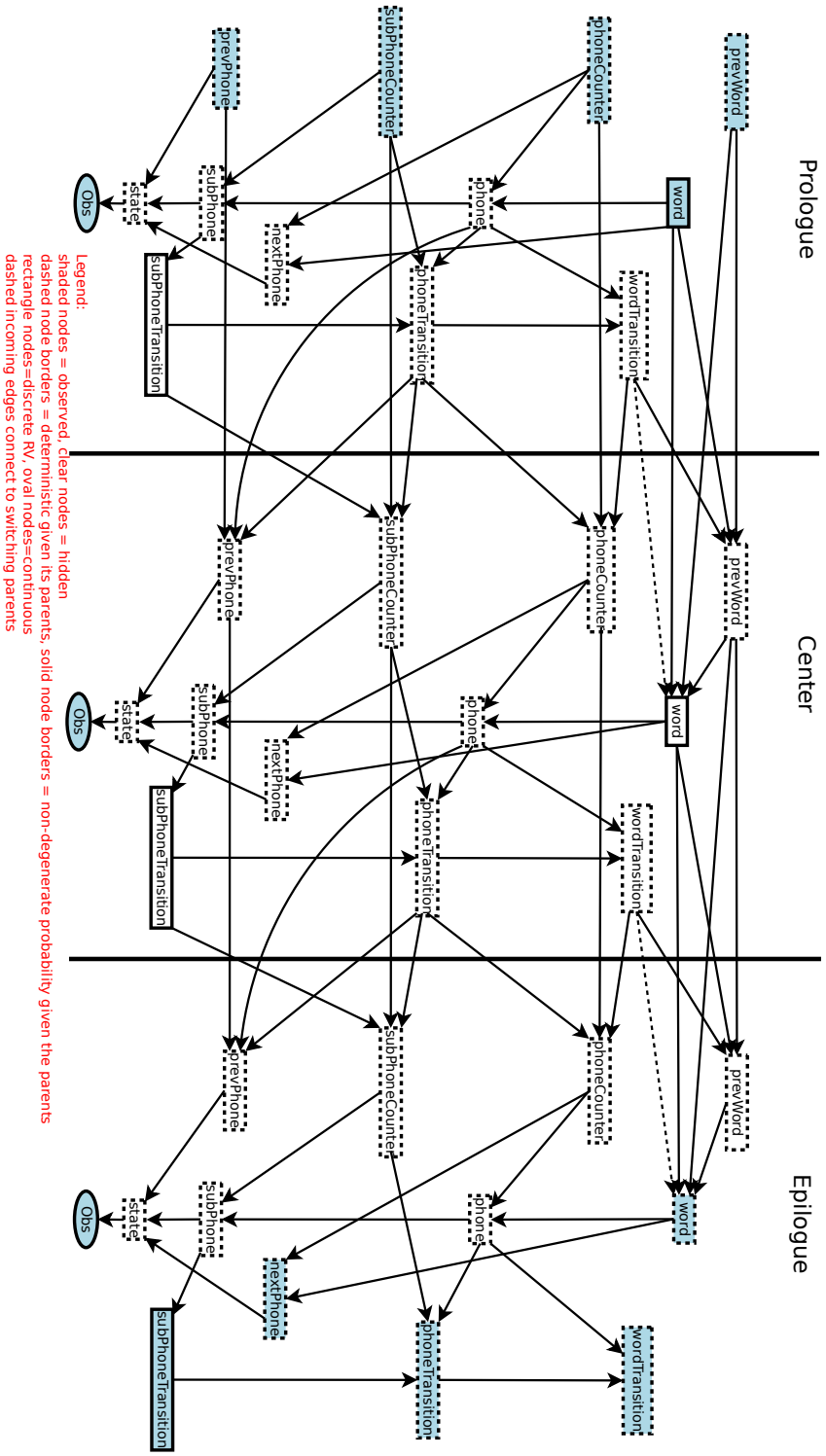

Figure 3.4: The DBN for the baseline word-internal triphone decoder using a trigram language model.

Minor variations of the refinement pattern in Figure 3.3 repeat two times during decoding, refining word → sub-word unit → substate.

The RVs $phoneCounter_t$, $phone_t$, $subPhoneCounter_t$ and $subPhone_t$ describe the pronunciation model. Their conditional distributions follow the refinement template. $subPhoneCounter_t$ ranges over two or three states, depending on the phone. The number of substates for each phone is given in Appendix A.1.

The $prevPhone_t$ and $nextPhone_t$ RVs are used to implement the triphones.

$$p(prevPhone_t|prevPhone_{t-1}, phone_{t-1}, phoneTransition_{t-1})$$

simply copies $phone_{t-1}$ or $prevPhone_{t-1}$ into $prevPhone_t$ based on whether $phoneTransition_{t-1}$ was true or not. The distribution for $nextPhone_t$ is defined so

$$p(nextPhone_t|word_t, phoneCounter_t) = p(phone_t|word_t, phoneCounter_t+1)$$

In case $phoneCounter_t$ points past the end of the word, $nextPhone_t$ is set to `EOW`.

$prevPhone_t$, $nextPhone_t$ and $subPhone_t$ are combined to select the Gaussian mixture RV $state_t$. The distribution

$$p(state_t|prevPhone_t, nextPhone_t, subPhone_t)$$

encodes the rules learned through decision tree clustering of triphones. The observation model $p(Obs_t|state_t)$ is a Gaussian mixture. The triphone clustering and the acoustic model are described in Section 3.3.6.

$subPhoneTransition_t$ is a binary RV that depends non-deterministically on the current frame's subphone, and is true only in the last frame of the current subphone. During decoding, $word_t$, $subPhoneTransition_t$ and Obs_t are the only RVs that are non-deterministic given their parents. The other RVs and their distributions are there just for maintaining the hidden state in each frame.

Applying the junction tree algorithm to this DBN merges all the hidden RVs from within the same frame in one large clique (some of the RVs may come from the next or previous frames, depending on the chosen elimination order). In this case, the junction tree algorithm treats the DBN as an HMM,

where all the hidden RVs form the HMM's hidden state, and the observed Obs_t variable depends only on the hidden state in frame t.

As is typical in continuous speech recognition, the state space in each frame is too large to explore completely. Therefore, all of the decoding is done with fairly aggressive beam pruning: at each frame t only the 10000 highest likelihood hypotheses are retained out of all the legal value assignments to the hidden variables.

3.2.3 The training DBN

The DBN for word-aligned training (Figure 3.5) is similar to the decoding DBN, except in the way the language model RVs are treated. During training the refinement pattern from Figure 3.3 is used once more, refining utterance $\rightarrow$ word.

At the utterance $\rightarrow$ word level, the transcription is specified by the $word_t$ and $wordCounter_t$ RVs. $wordCounter_t$ specifies the position of the word in the utterance, and the actual $word_t$ RV depends on the position. The utterance RV is not specified, but a different distribution $p(word_t|wordCounter_t)$ is given for each utterance. The $wordTransition_t$ RV is a binary variable which is true only if a word is ending in frame t.

Since Figure 3.5 depicts word-aligned training, both $wordTransition_t$ and $word_t$ are observed. For embedded training where the word boundaries are not specified, the only change to the DBN would be making $wordTransition_t$ and $word_t$ hidden.

Multiple pronunciations in training and decoding can also be implemented by introducing a new variable $pronunciation_t$ with distribution

$$p(pronunciation_t|word_t, pronunciation_{t-1}, wordTransition_{t-1})$$

which depends non-deterministically on $word_t$ if $wordTransition_{t-1}$ was true, and otherwise would deterministically copy the value of $pronunciation_{t-1}$.

3.2.4 Cross-word triphone DBN

We've also attempted to construct a cross-word triphone recognizer in which the triphone context can span word boundaries, rather than being word-

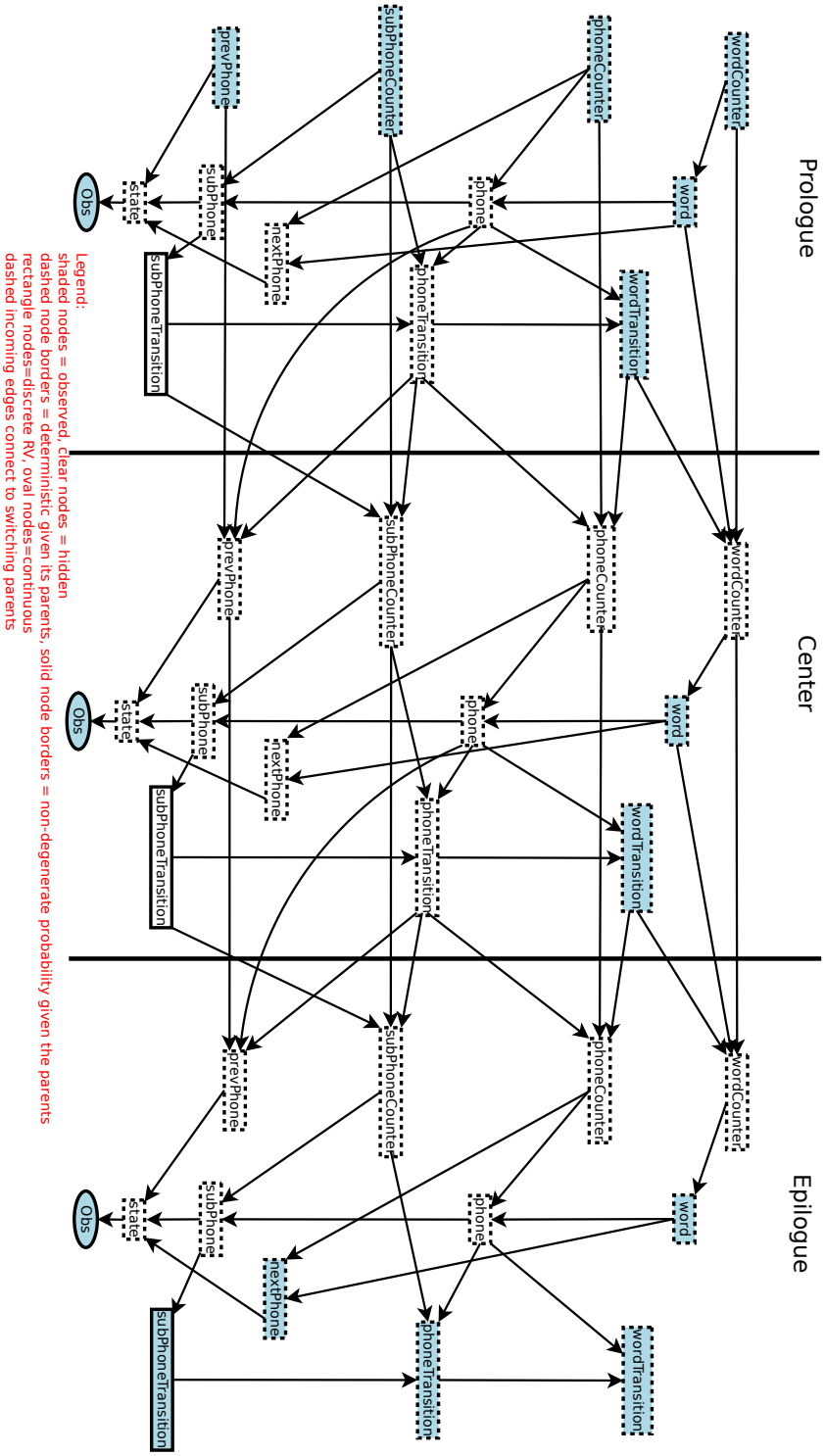

Figure 3.5: The DBN for the word-aligned training of a baseline word-internal triphone speech recognizer.

internal. The cross-word training DBN that was acceptable to GMTK is given in figure 3.6.

In the cross-word triphone DBN, the conditional probabilities for $prevPhone$ and $nextPhone$ become more complicated. Instead of copying the value of $phone$ into $prevPhone$ on each word transition, $phone$ is copied into $prevPhone$ only if $phone$ is not `EOW`. Otherwise $prevPhone$ its value from the previous frame.

The right context $nextPhone$ is set in the same way as the word-internal version, unless it's about to be set to `EOW`. In that case the $nextPhone_t$ is assigned $nextPhone_{t+1}$ which propagates forward all the way to a frame $t+k$ in which $wordTranstion_{t+k}$ is true, at which point $nextPhone_{t+k}$ is assigned the first phone of the next word $nextPhone_{t+k} = phone_{t+k+1}$.

Because a frame depends on the future frames, the cross-word triphone DBN is not a first order DBN, but the training and inference algorithms can theoretically handle it. However, training with this cross-word DBN was slower by about factor of 30. The cause for this was $nextPhone_t$'s dependence on the RVs from the future: $nextPhone_{t+1}$ and $phone_{t+1}$. This dependence comes into play only when $wordTranstion_t$ was true, but the GMTK inference engine was considering all possible values for $nextPhone_t$ regardless of whether $wordTranstion_t = 1$ was hypothesized, because it has no way of knowing this 'conditional dependence' without analyzing the actual conditional probability distribution table. It is likely that future versions of GMTK will be able to handle this case efficiently.

Making only the left context cross-word and the right context word internal made training and testing practical. However the WER slightly increased in the left context cross-word: from 53.0% to 53.6%. This happens possibly because word-boundary left context was not distinguished from word-internal left context, and so the triphone clustering had no way to learn separate models for word-internal and word-boundary contexts, even though the acoustics were different. This was not explored further, since our main experiments did not require a cross-word recognizer.

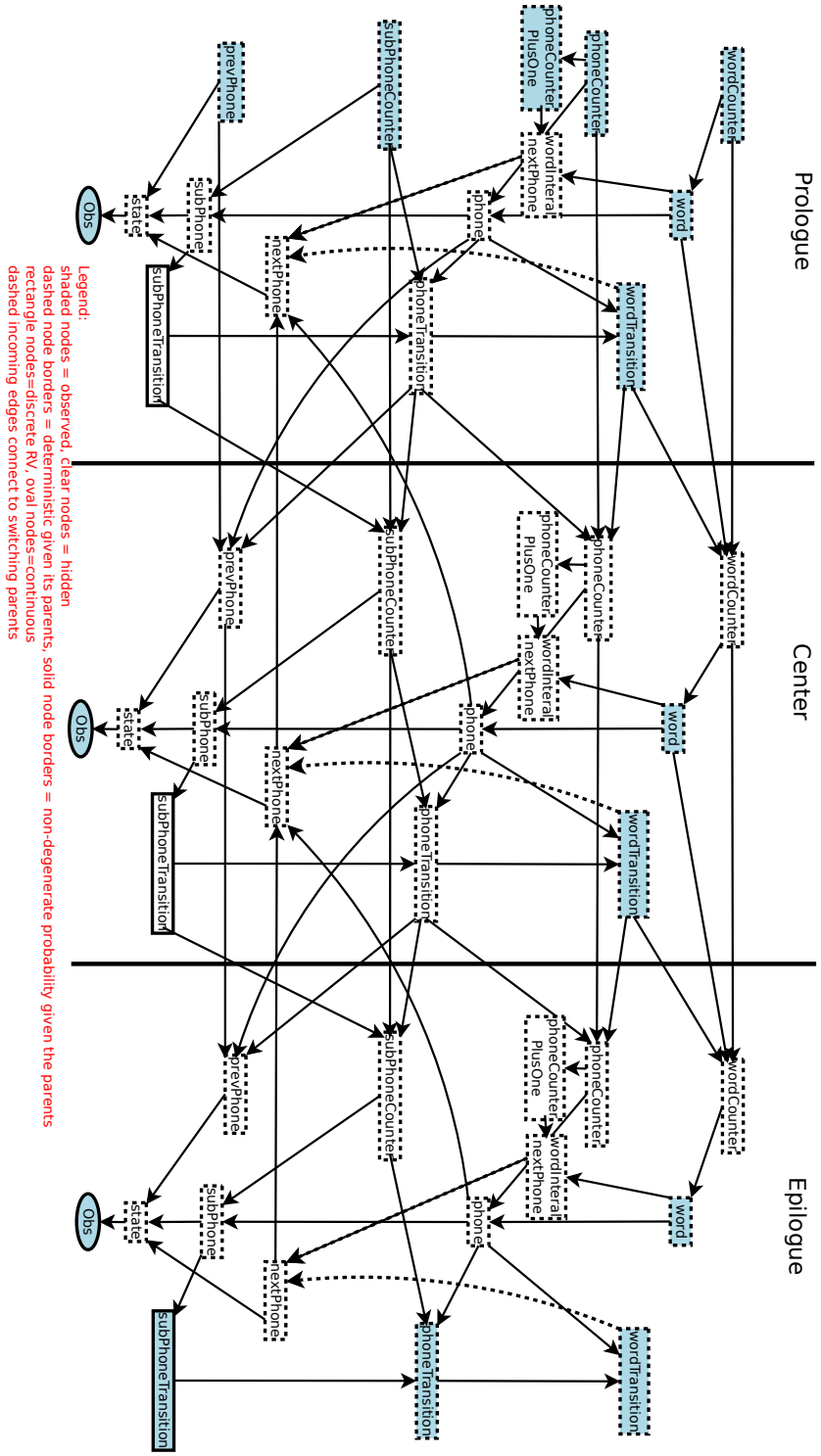

Figure 3.6: The DBN for the word-aligned training of a baseline cross-word triphone speech recognizer.

3.3 The baseline recognizer

This section describes the details of the baseline recognizer used in our main experiments, and also describes experiments used to justify some of the choices made during its design.

The baseline system includes some of the techniques that have traditionally yielded WER improvements, but some important improvements (e.g. vocal-tract normalization, MLLR, gender- and speaker-dependent models) were not included due to time constraints. Many aspects of the system were tuned on the Fisher corpus to get the best WER possible given the computational constraints.

Even though the outcomes of the experiments below could be anticipated, the experiments were performed to have a baseline against which one could quickly eyeball the WER contributions of an experiment's changes along multiple factors. For example, adding new pronunciation units might double the number of model parameters, which could be compared against doubling the components per mixture. Additionally, with these experiments one can see when the benefits of extra training are no longer worth the extra compute time, which is significant on the Fisher Corpus.

The experiments all differ slightly across the sections below, and the WERs are thus not comparable between sections.

3.3.1 The datasets

The Fisher conversational telephone speech corpus [68] was used in our speech recognition experiments. The 2000 hours of speech was divided into training, development and test sets of roughly 80%, 10% and 10% respectively. The train, development and test set boundaries fall on conversation boundaries, so no conversation can span two sets. All training (including the language model) used the training set exclusively, tuning was done on the Dev set, and the test set was only used for reporting results.

3.3.2 Front-end processing

The Fisher corpus segments the raw conversation recordings into utterances, and transcription is performed only on segments where speech is present

(which is about 1/3 of the time of the actual recording). The observations are extracted only from the speech waveform for which transcriptions exist.

The telephone signal is transmitted in the 0-4 kHz frequency band and the original Fisher recordings are sampled at 8 kHz. The recordings are further band-pass filtered using 125 Hz-3800 Hz band. This filtering was found in [69] to slightly improve the WER when the features are extracted as described here.

The acoustic observation features o (described in the next section) are built upon 12 perceptual linear predictive coefficients (PLPs) [70], plus the 0th PLP coefficient, taken every 10ms on a 25ms hamming-filtered window. The coefficients are augmented with the coefficient velocity and acceleration for a total of 39 features per observation vector. The observations are then mean and variance normalized per conversation side, as recommended in [69]. Then the observations are filtered with an auto-regressive moving average (ARMA) with an order-2 filter as described in [71]. These 39-element vectors are called PLP+Δ+$\Delta\Delta$.

The ARMA filtering was found to be helpful on the Aurora data set (noisy telephone speech for digit and digit-string recognition). ARMA filtering does not reduce the WER substantially on clean speech, while greatly improving the WER on noisy speech. The improvements are present regardless of whether mean-variance normalization/ARMA filtering took place before or after the velocity and acceleration computation step.

No attempt was made to check whether the ARMA filtering was as beneficial on conversational speech as on continuous digit recognition.

These observation features without ARMA filtering were created as inputs for the training of multi-layer perceptrons (MLPs) which are used in the tandem models (See Section 3.3.6). The ARMA filtered features could successfully be used as TANDEM model inputs as well [72].

3.3.3 Pronunciation model

Even though a multi-pronunciation dictionary for the Fisher corpus was available (as described in Section 5.5), we chose to derive from it a single pronunciation (spron) dictionary for the baseline recognizer.

The spron dictionary is derived by running the EM algorithm on the

training DBN augmented with the $pronunciation_t$ RV, as described in Section 3.2.3. In this way the distribution $p(pronunciation_t|word_t)$ over pronunciation variants is estimated, and the most frequent variants are retained for the single pronunciation dictionary.

There is evidence that a carefully designed spron dictionary outperforms mpron dictionaries on conversational telephone speech [14]. This is still true, but to a lesser extent even when discriminative training of the acoustic model is used.

Interestingly, the advantage of the spron over the mpron dictionary increases as the recognition task becomes more difficult. Low WER speakers are harmed by switching to a spron dictionary while high WER are helped by the switch (see Fig. 4 in [14]). The same paper also shows that going from mpron to spron raises the WER on well recognized read speech and lower the WER on the more difficult conversational speech. See Section 2.2 for a more detailed exploration of this trend.

In our own experiments with a monophone recognizer on conversational speech, the single pronunciation dictionary outperformed the multi-pronunciation dictionary: 63.9% to 64.7% WER. The decoding was also faster by a third. Therefore, the spron dictionary was used in the baseline experiments.

3.3.4 Language model

The language model (LM) used in the baseline is selected based on experiments described in Chapter 4. The LM is a backed-off, Good-Turning smoothed, entropy-pruned 3-gram model with a 10,000 word vocabulary.

The LM distribution is modified with insertion penalty a and scale b of the form

$$a + b \log(p(word_t|word_{t-1}, prevWord_{t-1}))$$

The penalty a controls the rate of word insertions: if a is smaller, longer words will be preferred and there will be fewer of them. The LM scale b controls the relative confidence of the LM against the acoustic model. Both a and b are tuned to minimize the WER on the development set. It is not readjusted for experiments other than the baseline, which may slightly disadvantage the experiment scores against the baseline.

3.3.5 Word-aligned training

To speed up training, one can specify hard word boundaries during training, instead of allowing the word boundaries to range anywhere within an utterance.

In our experiments, the word boundaries are determined by force-aligning training data to word transcriptions using a monophone model. This bootstrap model itself is trained without specifying hard word boundaries.

Without specifying word boundaries, it took around 18 days to train the bootstrap model on a 32-CPU cluster using 80% of the Fisher corpus. With word boundaries specified, the training took only 3.5 days, almost 6 times faster. As expected, the accuracy dropped somewhat: from 64.7% to 65.6% WER on a monophone model. Since most of our experiments required the model to be re-trained from scratch, the lower training time required is well worth the slightly lower WER for these experiments, so word-aligned training is done whenever possible.

To speed up training further, we reduced the amount of training data used to train the Gaussian models from 80% to 21%. As a result the WER of a triphone recognizer increased from 50.6% to 51.6%, but it is reasonable to expect that the conclusions of the experiments will still hold when models are trained on more data.

3.3.6 Acoustic models

In this section we describe the acoustic models (AMs) of the form

$$p(o|q) \tag{3.4}$$

which describes the kind acoustic observations o that can be expected given the hidden state q. The hidden state q is typically a phone, a triphone or some other sub-word unit, and o is a vector of features extracted from a short segment of speech.

Observation features

We use two kinds of observation features o in the experiments presented in this thesis. The simplest is to directly use the PLP+Δ+ΔΔ vectors described in Section 3.3.2 as observation features.

We also use tandem PLP+MLP observation features which are generally described in Section 2.3.1. The computation of the PLP+MLP observations is diagrammed in Figure 3.7.

The PCA dimensionality reduction in Equation 2.1 is designed to retain 95% of the total variance. In our case, this reduces the number of features from 48 to 23. The resulting 23-feature vector is mean-variance normalized and concatenated with the original PLP+Δ+ΔΔ observations. These are known as the 'PLP+MLP' tandem observation features are used for training and recognition.

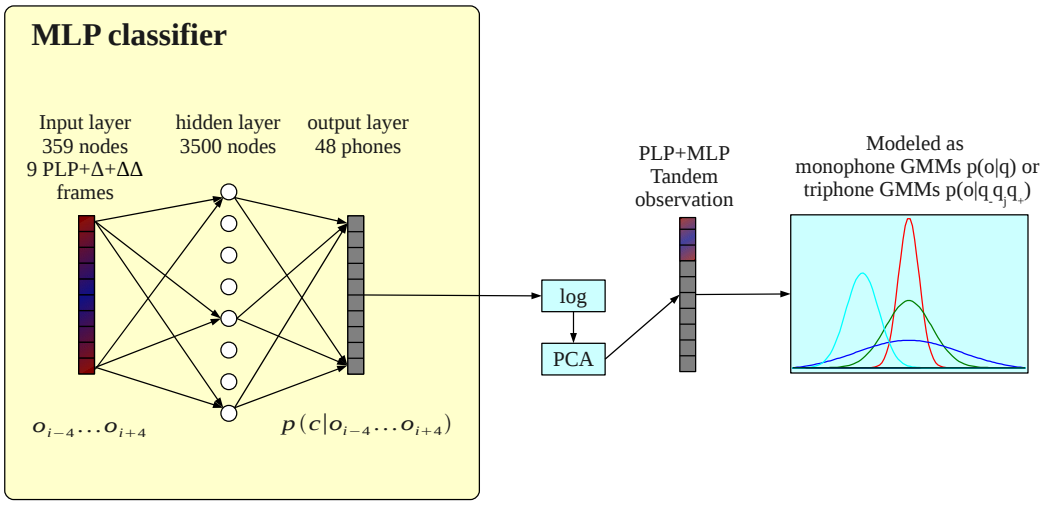

Figure 3.7: Overview of the acoustic model and computation of the PLP+MLP observation features.

MLP training

The MLP is trained on the entire training set of Fisher (see Section 3.3.1). The MLP training parameters and convergence schedule are analogous to the one described in [73]. The MLP considers the 4 frames on each side of the target frame (a total of 9 frames), and learns the function $MLP(i) = p(c|o_{i-4}, ..., o_{i+4})$. Each frame is PLP+Δ+ΔΔ vector. The c takes a value of

one of 48 phonemes, described in Section 5.4.2.

The target activations used to train the MLP are obtained by forced alignment of the Fisher transcriptions to the observations, using the single pronunciation dictionary. The MLP topology has 351x3500x48 nodes in the input x hidden x output layers, with neighboring layers fully connected.

The frame classification accuracy into phonemes is 59.39% on the training set and 57.07% on the cross-validation set.

The hidden state and triphone clustering

We used a clustered triphone substate as the hidden state q in Eq. 3.4. We use the triphone notation q_-qq_+, where q_-, q_+ and q are the left context, right context and center phone respectively. Let J_q be the number of sub-triphone states for triphones of form $*_-q*_+$. This means that the number of sub-states for each phone is independent of the context. J_q is allowed to be $J_q \in \{2, 3\}$. The phone `/EOW/` is special in the sense that $J_{\texttt{EOW}} = 1$. The number of sub-states for each phone is given in Appendix A.1.

A separate GMM model is trained for every hidden state $q_-q_iq_+$, however some of the hidden states share the means and the variances. Each non-speech phone (`SIL`, `NOISE`, `LAUGH`, etc.) is made context independent by tying across all contexts for each sub-phone state, so for example all $*_-NOISE_1*_+$ hidden states share the same mean and variances and are trained from the same data.

The remaining hidden states are partitioned into sets of form $*_-q_i*_+$. The hidden states are clustered using decision trees (DTs) as described in Chapter 3 of [74]. For each (unclustered) hidden state, a Gaussian model is trained over observations with the Baum-Welch algorithm. For each set of form $*_-q_i*_+$, a decision tree is constructed by asking binary questions about the phonetic features of left and right contexts. For example, a common question is "Is the right context q_+ a vowel?".

The decision tree is built top-down recursively from the root node. A node D in the decision tree represents a subset of $*_-q_i*_+$ hidden states selected by appropriately answering the questions of D's ancestor nodes. Let O_D be the acoustic training data generated by the states in node D. For each node D, a Gaussian model $\mathcal{N}(\mu_D, \sigma_D)$ is estimated from the node's data O_D. The total log-likelihood

$$\mathcal{L}_D = \ln(p(O_D; \mu_D, \sigma_D))$$

of the node's data can then be computed.

Define q_D to be the question asked at node D. q_D is chosen to maximize the difference between the sum of log-likelihoods of descendants of D and the log-likelihood at D:

$$q_D = \arg\max_q \left(\sum_{c \in C_q(D)} \mathcal{L}_c \right) - \mathcal{L}_D$$

where $C_q(D)$ is the set of children of D if question q is asked at node D. The recursive splitting of nodes stops when occupancy counts of the leaf nodes drop below some threshold, or the likelihood increase is below some threshold, or there are no more questions to ask.

This procedure works even if we don't know that observation $o \in O$ was generated from some single hidden state $j \in *_- q_i *_+$. All that is needed is an occupancy distribution γ_j specifying the probability that o was generated by state j. This occupancy distribution can be computed by the forward-backward algorithm. See [74] for more details.

With some reasonable assumptions, the clustering specified by the set of DT leaves locally maximizes the total likelihood of the training data over all possible clusterings. Once the decision tree is trained it can always assign an observation to a cluster, even if the observation was generated from a hidden state not seen in the training data.

While training on the Fisher corpus, the number of parameters (the size) of the model was not limited by the amount of the training data or the worry of over-fitting the model to the training data. Instead, the model size was limited simply by the amount of RAM available, and the desire to train the model efficiently. If our goal is to keep the size of a model roughly constant, we can consider the effects of the model parameter allocation on the WER. For example, we can keep the size of the model constant by doubling the number of triphone clusters (GMMs) and halving the number of Gaussians per mixture in each GMM.

Table 3.1 presents some coarse tuning results exploring this trade off. The only change was in the number of GMMs and the number of Gaussians per mixtures. In these experiments the minimum leaf occupancy was set to 100 and the minimum relative log-likelihood improvement was varied to achieve the desired number of clusters. After the clusters were determined,

the system was converged as described in the next section.

Rows 1-3 of the Table 3.1 shows that once the number of triphone clusters is large enough, the marginal WER benefit of more components per mixture outweighs the marginal WER benefit of more clusters. The best WER to model size ratio is reached by growing the number of clusters and Gaussians per mixture in a balanced way. For the desired model size of 64k components, our coarse tuning experiments, show that 1000 clusters with 64 Gaussians give a WER close to a local minimum. Row 5 shows that increasing the model size continues to significantly improve the WER.

Row	Total Components	Min improvement	Triphone clusters	Comp Per Mixture	WER
1	63728	0.3%	503	128	54.3%
2	65900	0.1%	1033	64	52.3%
3	61475	0.01%	3845	16	53.7%
4	30742	0.01%	3845	8	56.5%
5	245479	0.01%	3845	64	49.3%

Table 3.1: The WER trade-off of the number of triphone clusters vs. number of components per mixture trade off. "Min improvement" is the minimum relative log-likelihood improvement threshold for splitting a DT node.

The above experiments were performed for tuning purposes only. In the final baseline models, MinImprovement=.1 generated 1037 clusters for the PLP observation features, while for the PLP+MLP features 1193 clusters were generated.

Gaussian mixture models

The probability of observation given the (clustered) hidden state $p(o|q)$ is modeled as a Gaussian Mixture Model (GMM), with diagonal covariances. Typically, the GMMs are obtained by starting with 1 Gaussian per mixture (GPM) and iterating the EM algorithm to convergence using that model. Then the GMM is 'split' by adding a perturbed copy of some of the GMM's components back into the GMM, and again converging with the EM algorithm. The splitting and converging can be iterated until the model is contains the desired number of GPM. The number of EM iterations to con-

vergence and the strategy for splitting the GMMs is called a convergence schedule. For the baseline, we have experimented with different numbers of GPM, as well as different convergence schedules.

A common approach is to increase the number of GPM until the WER no longer improves. We were not able to reach this point with our baseline system, even by splitting up to 256 GPM (see Table 3.2). Larger mixtures were not possible due to RAM limitations. There is an improvement of 1.5% WER by going from 128 to 256 GPM, and it appears that further improvement is possible with even more GPM.

GPM	WER	WER Δ
1	75.9%	
2	69.6%	6.3%
4	65.9%	3.7%
8	63.5%	2.4%
16	60.5%	3.0%
32	57.3%	3.2%
64	55.1%	2.2%
128	53.4%	1.7%
256	51.9%	1.5%

Table 3.2: WER on 500 utterances of the development set as a function of number of GPM. The system used PLP observation features and 503 triphone clusters. Gaussians with a low mixture weight were removed from the mixture. So for example the model in the last row contained 127971 components, which is less than the expected $256 * 503 = 128768$ components.

We also explored the WER as a function of the number of GPM and the number of iterations to convergence, with the results presented in Figure 3.8. The figure suggests that models with more GPM benefit from more iterations. For example models with 4 GPM initially improve rapidly, but after 5 iterations, the improvement is slight, while models with 64 GPM improve more gradually, but appear to continue to improve even after 5 iterations.

The convergence schedule we finally chose always uses 8 EM iterations to converge, regardless of the log-likelihood improvement at each iteration. Once a model is converged, it is split by doubling of the number of GPM for each mixture. If any mixture component weights fall below 1/256 after a split and a single EM iteration, it is removed from the mixture. Only a small percentage of components were removed that way. We split a total of

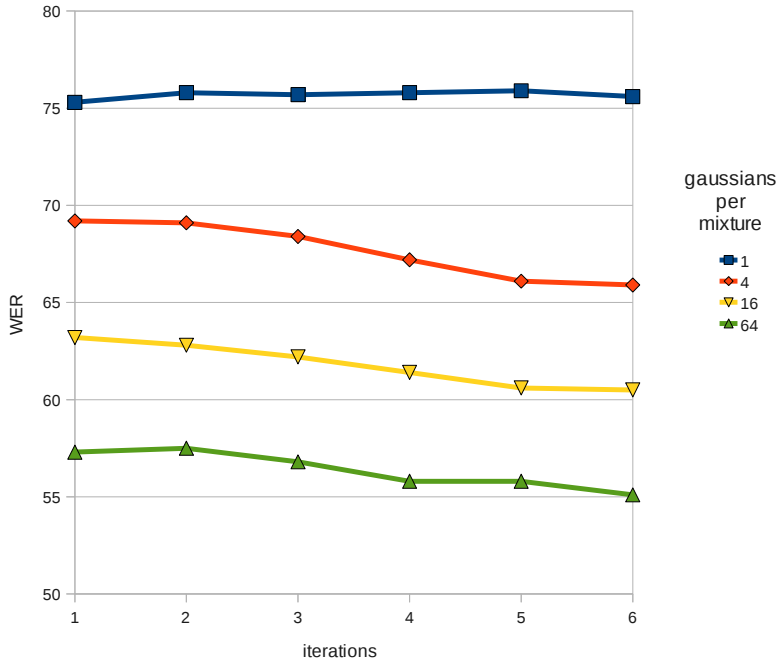

Figure 3.8: WER on 500 utterances of the development set as a function of GPM and iterations to convergence.

6 times so that the final models have on average slightly less than 64 GPM.

In a seperate test, moving from monophone to triphone model improved WER from 63.9% to 55.1%. In this experiment, the same convergence schedule was used for both triphones and monophones, with 6 EM iterations per split, and splitting up to 64 GPM. The number of states grew from 137 sub-monophones to 503 sub-triphone clusters.

3.4 Evaluation

The transcription normalization and scoring rules are described in detail, so that the results presented here can be compared to those of other approaches.

3.4.1 Transcription normalization

In our experiments, transcription normalization and scoring follows the NIST competition rules [75] as closely as possible. Both the hypothesis and reference transcriptions are normalized before scoring. The normalization is done with the NIST's `tranfilt` tool and the rule file is a slightly modified

version of the one used in the 2003 Spring competition [76]. The rewrite rules standardize spellings in the transcription (`KINDA` ⇒ `KIND OF`), correct misspellings, and apply other rules to support scoring as described below.

The Fisher reference transcription shows more variability and than the one used in NIST's competition, and so the NIST's normalization rules do not completely normalize the Fisher transcription. However, no additional rules are added, and therefore the WERs reported here probably overestimate a reasonably-scored WER.

The normalized hypothesis transcription is compared against the reference transcription with the NIST's `sclite` scoring tool. See section 8.1 of [75] for the way the special orthographic conditions are handled in the competition. Our scoring rules differ slightly from the competition rules and here we describe the actual scoring rules used in our experiments. The 'optionally deletable' words are marked in the reference transcription, so that if during scoring a word is deleted in the hypothesis transcription, it is not counted as deleted in the calculation of the WER. Optionally deletable words always contribute to the denominator in the WER calculation of Eqn. 3.5. The scoring rules are as follows:

Word Fragments: Word fragments (those with a '-' at the beginning or end of a word) are optionally deletable. Additionally if they correctly match the prefix or the suffix of a word, they will be marked correct. For example if the reference word is `TH-` and it is aligned with aligned with `THEY`, `THEY` will be marked as correct. This is the same as in NIST competition scoring. In our experiments we achieved the highest WER when the language model did not allow word fragments to be generated during the decoding.

Unintelligible or Semi-Intelligible Words: Each word inside indistinct phrases surrounded with `(( ))` are marked as optionally deletable, as in the NIST competition scoring.

Foreign Words: Foreign words are not treated differently from normal words, because they are not specially marked in the Fisher transcriptions. This differs from NIST competition scoring, where foreign words are optionally deletable.

Pause Fillers: Only the paused fillers recognized in NIST competitions are mapped to a single form *%HESITATION* and it is marked as optionally deletable. The NIST pause-fillers are: `UH`, `UM`, `EH`, `MM`, `HM`, `AH`, `HUH`, `HAH`, `HA`, `ER`, `OOF`, `HEE`, `ACH`, `EEE`, `EW` and `MHM`. There are some other similar pause fillers that occur in Fisher transcriptions (e.g. `ERM`) which are nevertheless NOT marked optionally deletable.

Non-lexical events: Fisher marks 11 non-speech sounds and events: `[NOISE]`, `[LAUGH]`, `[BREATH]`, `[COUGH]`, `[LIPSMACK]`, `[LAUGHTER]`, `[MN]`, `[SIGH]`, `[SNEEZE]`, `[PAUSE]` and `[[SKIP]]`. These words are deleted from both the reference and hypothesis transcriptions.

Multiple Spellings: The alternate spellings are normalized only if they are found in the NIST's normalization rules file, otherwise they are left alone.

Homophones: If the correct word is replaced by its homophone, it will be marked wrong, the same as in NIST competitions.

Compound Words: Compound words are expanded into component words if they are in NIST's normalization rules: (`BALLGAME` ⇒ `BALL GAME`). Compound word containing hyphens (`BIO-TERRORISM`), are split into component words as well (`BIO TERRORISM`).

Contractions: Contractions can optionally be pronounced as their expanded form: `CAN'T` will match `CAN'T` and `CAN NOT`. This differs from NIST where all contractions are expanded in the reference transcription. This expansion is ambiguous (`HOW'S` ⇒ `HOW IS` or `HOW HAS`) and cannot be done automatically in Fisher. Scoring contractions with our rules will will report a higher WER than scoring with NIST's rules.

Back-channel acknowledgments: Words like `UM-HUM`, `UH-HUH` (and other similar ones recognized in NIST's normalization rules) are mapped to a special positive acknowledgment word before scoring. Similarly, `UH-UH` and `HUH-UH` are mapped to negative acknowledgment. They are not made optionally deletable.

Using this kind of transcription normalization lowers the WER by about 1.8% absolute on the baseline system compared to computing the WER on the decoded transcription directly.

3.4.2 Recognizer evaluation

The Word Error Rate (WER) is a common metric for evaluating recognizer quality, and it is the one we used in most of our experiments. It is calculated by aligning the reference transcription with the hypothesis transcription by dynamic programming, and the WER% for the test set is defined as

$$WER\% = \frac{100 * (S + I + D)}{N} \tag{3.5}$$

where S, I and D are the total number of substitutions, insertions and deletions in the hypothesis transcription against the reference transcription, and N is the total number of words in the reference transcription. The WER can be greater than 100% if, for example, $I > N$.

It is useful to check whether one recognizer is statistically significantly better than another recognizer using the WER metric. The matched pairs test [77] has been developed to compare the recognizers on continuous speech recognition. In this test, the errors are assumed to be independent if they are separated by a sufficient number of correct words so that an error cannot be affected by the previous error through the language model. So for example, errors separated by two correct words are independent if the recognizer is using a tri-gram model. Whether the WER difference between two recognizers is statistically significant depends only on the errors where only one of the recognizers made an error. This means that the same WER difference on the same test set between two recognizers may or may not be statistically significant.

All our results reported on the test set are statistically significant beyond the $\alpha = .001$ level (and most of the tests reach a much higher level of statistical significance).

3.5 Discussion

This chapter presented the basic framework within which our pronunciation modeling experiments are performed. We described the DBN structures for training and recognition, the details of the recognizer and the evaluation framework. The DBN models were used because they offer more flexibility for pronunciation modeling over the traditionally used HMMs.

We use a clustered word-internal triphone recognizer which uses PLP+MLP tandem observation features. A number of parameters were tuned to make this baseline system perform as well as possible. The pronunciation model used here (phonetic single pronunciation dictionary) is simple, but it is similar to what is commonly used in state-of-the-art recognizers. The experiments against this baseline are presented in Chapters 6 and 7. The baseline recognizer and the recognition task are designed to be realistic enough so that the WER of our experiments relative to the baseline will be relevant in real world applications.

Chapter 4

LANGUAGE MODELING

The goal of the experiments described in this chapter was to produce a language model (LM) that brings the word error rate of our conversational telephone speech baseline system as close as possible to the state of the art. The best LM for this task has been demonstrated to be a function of the style of speech and the amount of RAM installed on the recognizer system. The most important parameters that affect system performance include vocabulary size, memory of the language model (the n in n-gram), the method for reducing the size of the language model by pruning away "unimportant" n-grams, and the method for smoothing or estimating the probabailities of n-grams never or rarely observed in the training data. These parameters interact to affect the LM quality in consistent but perhaps non-obvious ways.

The pruning and smoothing techniques, along with the LM quality metrics, are described in section 4.1. The way to use the LMs described here in a GMTK recognizer is briefly described in 4.2. Experimental tests varying these parameters on the Fisher corpus and results are described in 4.3. Section 4.3.1 also discusses the interaction of pruning and smoothing and Section 4.4 describes the parameter choice for the LM actually used in the baseline recognizer and all our experiments.

4.1 Background

A language model is a probability distribution over strings of words. If we have a sequence $w_1, ..., w_l$ of l words, the language model is the distribution

$$\begin{aligned} p(w_1, ..., w_l) &= \prod_{i=1}^{l} p(w_i | w_1, ..., w_{i-1}) \\ &\approx \prod_{i=1}^{l} p(w_i | w_{i-n+1}, ..., w_{i-1}) \\ &= \prod_{i=1}^{l} p(w_i | h) \end{aligned} \tag{4.1}$$

where Equation 4.1 assumes that words are conditionally independent, given they are separated by a long enough history h of $n-1$ words. n is the order of the language model: so if $n = 3$, our LM is a trigram LM. If $i - n + 1 < 1$, we can simply pad the beginning of the text with a special <BEGINNING> token.

Typically the LM is optimized independently of the acoustic model. Instead of optimizing the LM for WER directly, the LM is optimized to minimize cross-entropy $H_{pq}(w|h)$ on the development data:

$$H_{pq}(w|h) = -\sum_{w,h} q(w,h) \log p(w|h)$$

$p(w,h)$ and $q(w,h)$ are the distributions over word sequences estimated from the training and development data, respectively.

The LM perplexity (PP) is sometimes used as a measure of the LM quality instead of cross-entropy and it is defined as

$$PP = 2^{H_{pq}(w|h)}$$

.

4.1.1 Relationship between the WER and LM perplexity

The relationship between WER and PP is difficult to describe analytically. Recently, some authors have bypassed this question by jointly optimizing the

acoustic and language models to minimize the WER [35, 36]. For the systems where LM and the acoustic model are designed independently, a relationship between the WER and LM perplexity is observed. In [78], the authors give convincing empirical evidence that the WER and LM perplexity are related by the power law:

$$\log WER = a + b \log PP$$

where a and b are constants that depend on the data and the quality of the acoustic model. In other words, relative WER improvement is proportional to decrease of cross entropy of the LM. This assumes no out-of-vocabulary words in the test data.

On planned speech (Broadcast News corpus, DARPA 1996 and 1997 competitions), the relative WER improvement is 12%-20% for each bit decrease of cross-entropy of the LM [78]. For conversational speech, we can expect a slightly lower relative WER improvement per bit because the acoustic models will be less accurate.

4.1.2 Language model smoothing

Given some training text, the maximum likelihood LM estimate is simply

$$p_{ML}(w|h) = \frac{C(h,w)}{C(h)} \tag{4.2}$$

where h is a sequence of words, and $C(x)$ is the count of occurrences of the word string x in the training text. An important property of human languages is that a large probability mass of all word strings is contributed by a very large number of low probability strings. A reasonable-sized LM will always need to assign probabilities to strings which have never been seen in training data. If all of the possible strings are seen in the training data, performance can usually be improved by building an LM over longer strings, where training data sparsity again becomes a problem.

Now it apparent why minimizing the cross-entropy $H_{pq}(w|h)$ is reasonable. It can be written as

$$H_{pq}(w|h) = H_p(w|h) + D_{KL}(p(w|h)||q(w|h))$$

so we are minimizing the sum of conditional entropy of training distribution and the conditional KL-divergence between the training and development distributions. If the test distribution was identical to the training distribution $p = q$, then $D_{KL}(p(w|h)||q(w|h)) = 0$ and the cross-entropy $H_{pq}(w|h) = H_p(w|h)$ is minimized by the $p_{ML}(w|h)$ estimate. $D_{KL}(p(w|h)||q(w|h))$ penalizes the mismatch between p and q, which can be expected to be high on word string distributions due to the properties of human language.

If a word string h, w is never seen in the training data, $p_{ML}(w|h) = 0$ is an underestimate of the true probability of the string h, w and this will preclude the recognizer from considering this word string, no matter how well the acoustics may match. To address this, an LM is *smoothed*, by increasing the probability of low-probability strings, and decreasing the probability of high-probability strings.

A very thorough overview smoothing techniques is given in [79, 80]. The two smoothing strategies that generally give the lowest cross-entropies are the Good-Turing smoothing with Katz back-off and Kneser-Ney smoothing, and we repeat the definitions here to make a point later about the way smoothing interacts with pruning.

Good-Turing smoothing

Good-Turing smoothing groups n-grams by the number of times an n-gram was seen in the training data. The main idea is that probability mass assigned to all n-grams seen r times is spread equally among the n-grams seen $r - 1$ times. Let us define the event of encountering *any* n-gram that has been seen r times in the training data as M_r. If there are a total of n_r n-grams each of which has been r times, then according to the ML distribution, the probability of seeing event M_r is

$$p_{ML}(M_r) = \frac{n_r r}{N}$$

where N is the total number of n-grams. Good-Turing distribution p_{GT} is defined to satisfy

$$p_{GT}(M_r) = p_{ML}(M_{r+1}) \tag{4.3}$$

In particular, the probability mass assigned to all unseen n-grams is $p_{GT}(M_0) = p_{ML}(M_1)$.

In [79], the authors give a theoretical justification why satisfying Equation 4.3 is a good criterion. It can be satisfied by pretending that any n-gram seen r times is instead seen r^* times. Expanding Equation 4.3, we get

$$\begin{aligned} p_{GT}(M_r) &= p_{ML}(M_{r+1}) \\ \frac{n_r r^*}{N} &= \frac{n_{r+1}(r+1)}{N} \\ r^* &= \frac{n_{r+1}}{n_r}(r+1) \end{aligned}$$

This GT smoothing only works if $n_r > 0$, which is typically true for low values of r (say $r < 5$). For the high values $r > 5$ where GT is impossible, the counts are assumed to be reliable, smoothing can be omitted and the distribution is renormalized to sum up to 1.

Katz smoothing combines the n-gram LM with (n-1)-gram LM for n-grams unseen in the training data. While Good-Turing smoothing spreads the $p_{GT}(M_0)$ probability mass equally among unseen n-grams, Katz smoothing spreads the $p_{GT}(M_0)$ mass according to distribution based on (n-1)-grams. Let $h = w_{i-n+1}, ..., w_{i-1}$ be the history of an n-gram $(h, w_i) = (w_{i-n+1}, ..., w_i)$. If an n-gram (h, w_i) occurs r times in the training data, the Katz-smoothed distribution is defined as

$$p_{katz}(w_i|h) = \begin{cases} d_r \frac{C(h,w_i)}{C(h)} & \text{if } r > 0 \\ \alpha_h p_{katz}(w_i|w_{i-n+2}, ..., w_{i-1}) & \text{if } r = 0 \end{cases}$$

d_r is the discount taken off the observed n-grams. For Good-Turing discounting,

$$d_r \approx \frac{r^*}{r}$$

The equality is approximate and not exact because only low values of r are discounted, while for high values of r, $d_r = 1$ (see [79] for details).

α_h is chosen so that the probability mass to be allocated by the $(n-1)$-gram model is equal to the probability mass discounted from the $r > 0$ n-grams. Note that this α is computed separately for each history h using only the n-grams with that history. Keeping with our notation, the probability mass allocated to the event of encountering any n-gram unseen in the training data

given a history h is

$$p_{katz}(M_0|h) = 1 - \sum_{\{w_i:C(h,w_i)>0\}} d_{r(h,w_i)} \frac{C(h,w_i)}{C(h)}$$

where $r(h, w_i)$ is the number of times the n-gram h, w_i was seen. Then α_h must satisfy

$$\alpha_h \sum_{\{w_i:C(h,w_i)=0\}} p_{katz}(w_i|w_{i-n+2}, ..., w_{i-1}) = p_{katz}(M_0|h)$$

and solving for α completes the definition of the Katz/Good-Turing smoothed language model $p_{katz}(w_i|h)$.

Kneser-Ney smoothing

The Kneser-Ney smoothing approach is based on the observation that $(n-1)$-order LMs are influential only when the n-order LM does not contain the n-gram being modeled. Therefore, the $(n-1)$-order models should be made accurate on $(n-1)$-grams which do not occur only as a part of n-gram modeled by the n-order model. The example used in [79] is a bigram language model where the phrase "SAN FRANCISCO" is frequent, and "FRANCISCO" is almost always preceded by the word "SAN". The unigram probability of "FRANCISCO" will be high, and with $p_{katz}(w_i|h)$ it will have a high probability following some unseen history. But this is probably wrong, because "FRANCISCO" should only follow the one history "SAN". The Kneser-Ney smoothing addresses this situation.

Kneser-Ney smoothing uses absolute discounting where some fixed discount $D \leq 1$ is subtracted from the count of every n-gram seen in the training data. As in [79], we define

$$N_{1+}(h, \bullet)$$

to be the number of unique n-grams seen in the training one or more times with history h. Smoothing with absolute discounting has the form

$$\begin{aligned} p_{KN}(w_i|h) = & \frac{\max\{C(h,w_i) - D, 0\}}{\sum_{w_i} C(h,w_i)} \\ & + \frac{D}{\sum_{w_i} C(h,w_i)} N_{1+}(h,\bullet) p_{KN}(w_i|w_{i-n+2}, ..., w_{i-1}) \end{aligned} \quad (4.4)$$

and this form is still a valid distribution.

The original motivation of Kneser-Ney smoothing was for the smoothed distribution marginalized over the oldest word in the history to equal the marginalized ML distribution:

$$\sum_{w_{i-n+1}} p_{KN}(w_{i-n+1}, ..., w_i) = p_{ML}(w_{i-n+2}, ..., w_i) \tag{4.5}$$

In [79], the authors show that combining Equations 4.4 and 4.5 yields

$$p_{KN}(w_i|w_{i-n+2}, ..., w_{i-1}) = \frac{N_{1+}(\bullet, w_{i-n+2}, ..., w_i)}{\sum_{w_i} N_{1+}(\bullet, w_{i-n+2}, ..., w_i)} \tag{4.6}$$

The estimate $p_{KN}(w_i|w_{i-n+2}, ..., w_{i-1})$ itself could be improved through smoothing, and here again Kneser-Ney smoothing can be used. The details for this recursion are clearly explained are in the SRILM manual [81]. The distribution $p_{KN}(w_i|w_{i-n+2}, ..., w_{i-1})$ is completely different from the original $p_{KN}(w_i|w_{i-n+1}, ..., w_{i-1})$. It is not an n-gram estimate with a shorter history.

Note that the weight of the lower-order model is proportional to the number of new words following the history h, while the fraction of the weight going to a particular $n-1$-gram, is proportional to the number of (leftmost words of) histories. So if the only history for "FRANCISCO" is "SAN", $p_{KN}(FRANCISCO|APPLE)$ will be low even if FRANCISCO is common.

Generally Kneser-Ney smoothing yields lower cross-entropies than Katz/Good-Turing smoothing when model pruning is not used.

Kneser-Ney smoothing constructs the $(n-1)$-order model to complement the n-grams modeled by the n-order model. This contrasts with Katz/Good-Turing smoothing, which builds the $(n-1)$-order model independently of the n-order model to be the best estimate of the true distribution $p(w|w_{i-n+2}, ..., w_i)$, while accounting for training data sparsity. The difference is important when considering the interaction of smoothing techniques with entropy-based model pruning, which is described in the next section.

4.1.3 Entropy-based language model pruning

The 3- and 4-gram LMs are usually too large to be used directly in an LVCSR system. The goal then is to prune n-grams from the LM without raising the

cross-entropy of the LM. One simple strategy is to exclude n-grams with counts less than k. But this approach gives only coarse control of the model size, and for a given model size, lower cross-entropies can be achieved with other pruning methods.

Instead, entropy-based pruning described in [82] is often used. This approach can prune an arbitrary number of n-grams, raises the entropy less than removing low-count n-grams, can efficiently update the n-gram probabilities and back-offs and only needs the information in the LM being pruned, so there is no need to keep around the original n-gram counts.

Entropy-based pruning seeks to minimize the KL-divergence between the original LM $p(w_i|h)$ and the pruned model $p'(w_i|h)$.

$$D_{KL}(p(w_i|h)||p'(w_i|h)) = \sum_{w_i,h} p(w_i,h)\left(\log p(w_i|h) - \log p'(w_i|h)\right)$$

In [82], the authors specify exactly how $p(w_i|h)$ is updated into $p'(w_i|h)$ by removing a single n-gram. $D_{KL}(p(w_i|h)||p'(w_i|h))$ increase is computed due to removing every n-gram individually. Only those n-grams are removed that do not increase $D_{KL}(p(w_i|h)||p'(w_i|h))$ by more than some threshold.

In [82] the authors show that entropy pruning can reduce the size of the LM by a factor of four without increasing the WER of their recognizer, and raising the LM cross-entropy only slightly. Entropy-pruning an n-gram model down to the size of an $(n-1)$-gram model yields a lower cross-entropy model than just using an unpruned $(n-1)$-gram model.

4.2 Experimental methods

The experiments are performed using the SRILM language modeling toolkit [83]. The model is saved in the standard ARPA language model format, which contains the conditional $p(w|h)$ probabilities for n-grams, and all lower-order grams as well as the back-off weights for (n-1) and lower order n-grams.

This ARPA file is used by a constellation of RVs in the decoding DBN described in Section 3.2.2. In the case of the trigram and using the names of RVs in Figure 3.4, the distribution $p(word_i|prevWord_i, prevWord_{i-1}) = p(w|h)$ is used whenever $wordTransition_{i-1} = 1$.

4.3 Experimental results

We have experimented with various parameters of the LM, searching for a compromise model that is both low-cross entropy and computationally tractable within our speech recognizer. The language model is built using the training set transcriptions with partial words repaired as described in Section 5.4.1. Each unique partial repaired word is treated as a separate word.

The model quality is evaluated as the cross-entropy per word and out-of-vocabulary (OOV) rate. All combinations of the design choices in Table 4.1 are tried on both the dev and test sets, with the results presented in 4.2. A sortable table can found in [84]. The cross-entropy values are computed in the same way as in [79] and can be compared to the results reported there.

Parameter	**tried values**
order	2,3,4-gram
vocabulary size	500, 1k, 5k, 10k, 20k, 70957 (all words and fragments)
smoothing	Kneser-Ney, Good-Turing
pruning	none, entropy-based

Table 4.1: All combinations of the above language model parameters were tried, with model perplexities and OOV rates reported in Table 4.2.

In these tests, the $\langle UNK \rangle$ token replaces all words not in the vocabulary, and the LM can have the $\langle UNK \rangle$ token in it's N-grams. Word fragments are allowed.

n-gram order	**vocab**	**total n-grams**	**smooth**	**prune**	**dev**		**test**	
					H	**OOV%**	**H**	**OOV%**
4-gram	500	7576736	KN	no-pr	5.4	15.46	5.45	15.09
4-gram	500	2456011	GT	no-pr	5.4	15.46	5.46	15.09
4-gram	500	264641	GT	pr	5.43	15.46	5.49	15.09
4-gram	500	569016	KN	pr	5.7	15.46	5.75	15.09
4-gram	20k	16181859	KN	no-pr	6.76	0.81	6.77	0.75
4-gram	20k	3856702	GT	no-pr	6.79	0.81	6.8	0.75
4-gram	20k	464271	GT	pr	6.89	0.81	6.89	0.75
4-gram	20k	762636	KN	pr	6.99	0.81	7.01	0.75
4-gram	10k	15603196	KN	no-pr	6.64	1.66	6.66	1.53

Continued on next page

Table 4.2 – continued from previous page

n-gram order	vocab	total n-grams	smooth	prune	dev		test	
					H	OOV%	H	OOV%
4-gram	10k	3717947	GT	no-pr	6.67	1.66	6.69	1.53
4-gram	10k	448768	GT	pr	6.76	1.66	6.78	1.53
4-gram	10k	719912	KN	pr	6.86	1.66	6.9	1.53
4-gram	5k	14591103	KN	no-pr	6.47	3.12	6.5	2.92
4-gram	5k	3534140	GT	no-pr	6.49	3.12	6.52	2.92
4-gram	5k	425285	GT	pr	6.57	3.12	6.6	2.92
4-gram	5k	681505	KN	pr	6.68	3.12	6.72	2.92
4-gram	1k	10204831	KN	no-pr	5.81	10.01	5.85	9.81
4-gram	1k	2899330	GT	no-pr	5.82	10.01	5.86	9.81
4-gram	1k	326234	GT	pr	5.87	10.01	5.91	9.81
4-gram	1k	602069	KN	pr	6.05	10.01	6.1	9.81
4-gram	all	16621480	KN	no-pr	6.88	0.27	6.89	0.25
4-gram	all	4031432	GT	no-pr	6.91	0.27	6.91	0.25
4-gram	all	512518	GT	pr	7.01	0.27	7.01	0.25
4-gram	all	862439	KN	pr	7.11	0.27	7.12	0.25
3-gram	500	1844779	KN	no-pr	5.45	15.46	5.51	15.09
3-gram	500	919391	GT	no-pr	5.46	15.46	5.52	15.09
3-gram	500	201453	GT	pr	5.5	15.46	5.55	15.09
3-gram	500	456598	KN	pr	5.57	15.46	5.62	15.09
3-gram	20k	6366532	KN	no-pr	6.8	0.81	6.82	0.75
3-gram	20k	2516904	GT	no-pr	6.84	0.81	6.85	0.75
3-gram	20k	417045	GT	pr	6.94	0.81	6.94	0.75
3-gram	20k	760104	KN	pr	6.94	0.81	6.96	0.75
3-gram	10k	5934715	KN	no-pr	6.69	1.66	6.71	1.53
3-gram	10k	2342585	GT	no-pr	6.72	1.66	6.74	1.53
3-gram	10k	400424	GT	pr	6.81	1.66	6.82	1.53
3-gram	10k	716844	KN	pr	6.82	1.66	6.85	1.53
3-gram	5k	5251482	KN	no-pr	6.51	3.12	6.54	2.92
3-gram	5k	2098252	GT	no-pr	6.54	3.12	6.57	2.92
3-gram	5k	375374	GT	pr	6.62	3.12	6.65	2.92
3-gram	5k	673910	KN	pr	6.63	3.12	6.67	2.92
3-gram	1k	2902901	KN	no-pr	5.86	10.01	5.9	9.81
3-gram	1k	1309423	GT	no-pr	5.88	10.01	5.92	9.81
3-gram	1k	267219	GT	pr	5.93	10.01	5.96	9.81
3-gram	1k	544497	KN	pr	5.95	10.01	5.99	9.81
3-gram	all	6737974	KN	no-pr	6.93	0.27	6.93	0.25
3-gram	all	2713750	GT	no-pr	6.96	0.27	6.96	0.25
3-gram	all	465498	GT	pr	7.06	0.27	7.05	0.25

Continued on next page

Table 4.2 – continued from previous page

n-gram order	vocab	total n-grams	smooth	prune	dev		test	
					H	OOV%	H	OOV%
3-gram	all	858818	KN	pr	7.07	0.27	7.08	0.25
2-gram	500	127889	KN	no-pr	5.8	15.46	5.86	15.09
2-gram	500	127889	GT	no-pr	5.8	15.46	5.86	15.09
2-gram	500	60296	GT	pr	5.85	15.46	5.9	15.09
2-gram	500	70752	KN	pr	6.14	15.46	6.2	15.09
2-gram	20k	1196182	KN	no-pr	7.19	0.81	7.21	0.75
2-gram	20k	1196182	GT	no-pr	7.21	0.81	7.23	0.75
2-gram	20k	387048	KN	pr	7.25	0.81	7.26	0.75
2-gram	20k	243371	GT	pr	7.27	0.81	7.28	0.75
2-gram	10k	1013070	KN	no-pr	7.08	1.66	7.11	1.53
2-gram	10k	1013070	GT	no-pr	7.1	1.66	7.12	1.53
2-gram	10k	327131	KN	pr	7.13	1.66	7.15	1.53
2-gram	10k	224037	GT	pr	7.15	1.66	7.17	1.53
2-gram	5k	781554	KN	no-pr	6.91	3.12	6.94	2.92
2-gram	5k	781554	GT	no-pr	6.92	3.12	6.96	2.92
2-gram	5k	265487	KN	pr	6.94	3.12	6.98	2.92
2-gram	5k	196611	GT	pr	6.96	3.12	6.99	2.92
2-gram	1k	266359	KN	no-pr	6.24	10.01	6.28	9.81
2-gram	1k	266359	GT	no-pr	6.25	10.01	6.29	9.81
2-gram	1k	100363	GT	pr	6.29	10.01	6.33	9.81
2-gram	1k	121297	KN	pr	6.45	10.01	6.5	9.81
2-gram	all	1413153	KN	no-pr	7.32	0.27	7.32	0.25
2-gram	all	1413153	GT	no-pr	7.33	0.27	7.34	0.25
2-gram	all	503370	KN	pr	7.37	0.27	7.38	0.25
2-gram	all	292708	GT	pr	7.39	0.27	7.39	0.25

Table 4.2: The cross-entropy (H) and OOV rate on the development and test sets of the LMs trained on the training set. Various combinations of model order, vocabulary size, smoothing and pruning are tried. Smoothing is either Kneser-Ney or Good-Turning and either no pruning is done or entropy-based pruning [82] is performed. The experiments are sorted by n-gram order, vocabulary, and then the number of n-grams in the model.

2-gram, 3-gram and 4-gram models were built. n-grams for $n > 4$ do not yield significant improvements. Going from 4-grams to 5-grams reduces the cross-entropy by .06 bits in experiments described in [80]. Such a small improvement is not worth the extra computational resources required by 5-gram model. Essentially nothing is gained by LMs larger than 5-grams.

The models tested use only $N \in \{500, 1000, 5000, 10000, 20000, 70957\}$ most frequent words, mapping the rest to the $\langle UNK \rangle$ word. 70957 is the total number of words and repaired words in the training data set.

Both Kneser-Ney (KN) and Good-Turning (GT) smoothing is tried, in combination with and without entropy-based pruning [82]. Without entropy pruning, KN smoothing outperforms GM smoothing which is consistent with the results in [80]. The difference between the smoothing methods is typically no more than .04 bits, or a relative WER difference of about $12\% * 0.04 = 0.5\%$ when using the estimate from the previous section. As Goodman succinctly states "...in practice, most language models are built with high count cutoffs, to conserve space, and speed the search; with high count cutoffs, smoothing doesn't matter.[80] "

However when entropy pruning is used, things look different. For almost all experiments (all experiments for 3- and 4-grams, and 2-gram experiments with vocab size $\leq$ 1k), GT-smoothed models have lower cross-entropy than KN-smoothed models. This is true despite the GT models having fewer total n-grams than the KN models. This is consistent with results presented in [85, 86]. The reason for this is explored in [86] and summarized in Section 4.3.1. In our LM experiments the smoothing and pruning choices affect the LM cross-entropy by at most .3 bits (at most $12\% * .3 = 3.6\%$ relative WER) which can be a meaningful difference in the overall WER of a recognizer.

We also note that while KN-smoothed models outperform GT-smoothed models on large vocabulary unpruned LMs, they do worse on unpruned LM with reduced vocabularies (5000 words or less) across all n-gram orders. The reason for this is is also discussed in the next section.

4.3.1 Interaction of pruning and smoothing

In [86], an explanation is given for why Kneser-Ney smoothing followed by aggressive entropy based pruning yields poor models.

Removing an n-gram h, w_i from $p(w_i|h)$ changes it only through estimates

involving history h, and no other histories. Therefore we can write

$$D_{KL}(p(w_i|h)||p'(w_i|h)) = \sum_{w_i} p(w_i, h) \left(\log p(w_i|h) - \log p'(w_i|h)\right)$$
$$= p(h) \sum_{w_i} p(w_i|h) \left(\log p(w_i|h) - \log p'(w_i|h)\right) \quad (4.7)$$

The entropy-based pruning algorithm takes advantage of the fact that every quantity in Equation 4.7 can be computed from the language model itself. In particular, it calculates $p(h)$ as

$$p(h) = p(w_{i-n+1}, ..., w_{i-1}) = p_{model}(w_{i-n+1}) \prod_{j=1}^{n-2} p_{model}(w_{i-j}|w_{i-n+1}, ..., w_{i-j-1})$$

where $p_{model}(w_{i-n+1})$ and $p_{model}(w_{i-j}|w_{i-n+1}, ..., w_{i-j-1})$ are calculated from the smoothed language model itself.

This makes sense for Katz/Good-Turing smoothing. For Kneser-Ney smoothing, the lower order models are not an estimate for the true n-gram distribution. Kneser-Ney lower-order models are trained by excluding tokens which occur in higher-order n-grams. Therefore $p(h)$ calculated from a Kneser-Ney smoothed LM will be a poor estimate of the true distribution, and the $D_{KL}(p(w_i|h)||p'(w_i|h))$ will be inaccurate.

Even if $p(h)$ is correctly estimated (say from maximum likelihood or Katz/Good-Turing smoothed models) and the correct n-grams are targeted for removal, simply removing n-grams from higher-order Kneser-Ney smoothed models introduces problems. This is because the (n-1)-order models are not designed to model n-grams which occur in the upper-level models. In [86], the authors perform a similar experiment and confirm that correcting the $p(h)$ greatly improves the pruned Kneser-Ney performance, but not enough to outperform pruned Katz/Good-Turing.

For similar reasons, Kneser-Ney performs poorly on LMs with reduced vocabularies. Since the words with low token counts are removed, the n-grams containing them in their histories are also pruned from the n-order model, and (n-1)-models are forced to model (n-1)-grams that were excluded from their training.

4.4 Discussion

As a sanity check, we can compare our LMs to those built on other corpora of English conversational speech. For example, Figure 5 in [79] shows a trigram model built on the Switchboard corpus with cross-entropy of 6.5 bits while our trigram model built on Fisher is 6.8 bits. Considering the shorter sentences in Fisher (8.8 words per sentence on average vs 16 words per sentence in Switchboard), and the larger vocabulary due to repaired word fragments, perplexities of the two models seem to be on par.

Which parameter choices of those listed in Table 4.1 are best for ASR? In our experiments, entropy pruning reduces the number of n-grams in a model by 2 to 9 times while raising the LM cross-entropy by at most .09 bits. This seems like a clearly worthwhile trade-off and so we should use Good-Turing smoothing and entropy-based pruning.

In our Viterbi decoding implementation, all of the $vocab^n$ potential n-grams are enumerated for each frame, so this makes LM order $n > 3$ impractical, and a 10k vocab seemed like a reasonable compromise between the decoding speed and the OOV rate. 4-gram models would have been beneficial with a more efficient decoder implementation.

In summary, the baseline recognizer language model consists of 10,000 word vocabulary, and uses a backed-off, Good-Turning smoothed, entropy-pruned 3-gram model. The vocabulary does does not contain any partial words (words starting or ending with a '-'). The language model also does not contain the special $\langle UNK \rangle$ token denoting all words not in the vocabulary. Removing word fragments from the LM as well as removing all n-grams with the $\langle UNK \rangle$ token each improved the WER rate. The language model has an OOV rate of 2.22% and cross-entropy of 6.773 bits on the dev set. The OOV rate and entropy reported here differ from those in Table 4.2 due to the way unknown words and word fragments are treated.

Chapter 5

HMM-BASED LEXICON GENERATION

Chapter 2 described methods of adapting an existing (often human-generated) pronunciation lexicon to improve the WER. In this chapter, we discuss generating a phonetic pronunciation from an orthographic spelling of words. This is useful in a variety of contexts, including text-to-speech applications, automatic spelling correction, and generating a pronunciation lexicon for a new training dataset which contains out-of-vocabulary (OOV) words. The next section describes the previous work, Section 5.2 describes the letter-to-phone function and 5.3 evaluates the accuracy of the system.

We've used the mechanism described here to generate the pronunciations for OOV words and word fragments in the Fisher corpus. Section 5.4 describes some statistics of the Fisher transcriptions and the lexicon, and 5.5 describes how the Fisher lexicon was generated. Section 5.6 discusses the lexical ambiguity which remains even after the phonetic transcription is known.

5.1 Background

A variety of methods have been used to design a mapping from orthographic strings to their phonetic pronunciations. A system which learns a set of context-dependent letter-to-phone rules achieves a ∼6% phone error rate (PER) using a context of up to eight letters in [87]. The system is used to generate OOV words for a new lexicon. The same paper gives a more extensive bibliography of related work.

A similar but improved pronunciation model is used in [88] to correct spelling mistakes in the cases where the typist makes no typos but does not know the proper spelling of a word. In this case, the mistyped word will be pronounced similarly to the desired word, and looking for correctly-spelled alternatives in the pronunciation space is beneficial.

All of the pronunciation models (including the one presented in the next section) only require a dictionary - they do not depend on language, alphabet or phone sets. All of them can be used for completely new word generation, and for determining a pronunciation for fragments of existing words, which are common in conversational speech transcriptions.

5.2 HMM-based letter-to-phone mapping

The system described here generates pronunciations for an arbitrary string of letters with HMMs, representing each phoneme as an HMM and treating the letters as observations. A person tries to pronounce a word with phonemes, but the phonemes get corrupted due to the noise in the channel, the word comes out written with letters instead.

The HMM structure representing each phoneme is shown in Figure 5.1. The HMM is greatly constrained: there are no self loops, and it can emit exactly one, two or three letters. There are no shared distributions among any of the HMM states. The HMMs are flat-start initialized. Any phoneme is permitted to follow any other phoneme, and a bigram model gives the distribution of phoneme sequences.

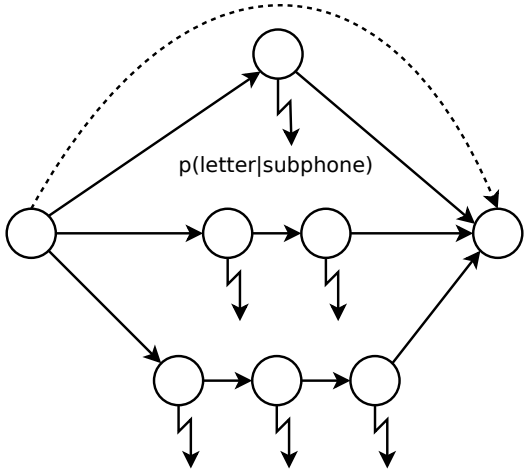

Figure 5.1: The finite state automaton view of an HMM representing a single phoneme. Only the subphone states with lightning arrows emit letters. They emit exactly one letter since there are no self loops. The dashed transition is only possible in a tee-model, and if that transition is taken, no letters will be emitted by the phone.

The monophone HMMs are then cloned into triphone HMMs and re-estimated after the 'teeing' step described next.

In English using the standard letter set and the CMU phone set, occasionally two phones are required to produce a single letter, which is disallowed by HMM structure. To solve this, an HMM can be modified into a *tee-model*, where the entire HMM can be optionally traversed without emitting even a single letter. We have implemented the system in the HTK toolkit [89], which allows tee-models, but does not allow two consecutive tee-models either during training or decoding.

We can consider the triphones as weighted nodes in a graph, where the nodes are adjacent if the triphones are neighbors, and node weights are the number of times the node co-generates a single letter with other nodes. Then, selecting the set of non-adjacent triphones with maximum total weight is exactly the *maximum weighted independent set* problem, which is NP-complete.

Instead of finding the optimal solution, we select a set of tee-models greedily. We consider all the words that have more phones than letters, which must contain letters mapping to more than one phone. We make a list of all the triphone models occurring in these words, and sort the list by number of occurrences. Then we simply go down the list and tee any model which does not conflict with any of the models tee'd so far.

Once all the possible HMMs have been tee'd, all the HMMs are re-estimated with the EM algorithm. Decoding is done with the viterbi algorithm, which uses a bi-gram language model over the phones. Viterbi search takes place over a small state space, and so search beam pruning is not necessary.

The letter-to-phone function trained on the english CMU dictionary is available on the web [90].

5.3 Evaluation of letter-to-phone mapping

Our dictionary generation system is trained on a random 90% subset of the CMU dictionary, and tested on the remaining 10%. The monophone and triphone systems have 28.5% and 14.2% PER respectively, while the best systems reported by [87] and [88] have 6% and 8.5% PER respectively. These PERs are not completely comparable because different dictionaries were used. For example, only the subset of words that was identically pronounced in 4

out of 11 dictionaries was used for training and evaluation in [87]. Our HMM letter-to-phone system underperforms previously published ones. Some easy improvements to the system are described in Section 8.1.1.

We have not spent much time perfecting our system, since we used it only to generate pronunciations for OOV words covering about 1.21% of the transcriptions (see Table 5.2). Out of those, we were able to force align words covering 0.46% of the transcription – a task which can be performed accurately by our model (although we could not evaluate the force-alignment accuracy for lack of reference pronunciations). Even poor pronunciation definitions for the remaining 0.75% of the training tokens is not likely to greatly effect the final recognizer models.

5.4 Lexicon generation for the Fisher corpus

We have used the HMM-based pronunciation model described above to generate phonetic pronunciations for OOV words encountered in the Fisher transcriptions. In this section we give some statistics of the Fisher transcriptions and the lexicon.

5.4.1 The Fisher transcriptions

The Fisher corpus is automatically segmented into utterances, and each utterance is word-transcribed by a human. If some part of speech was not clear, the transcriber would put her best guess for what was said in (( )) parentheses. If speech was truly unintelligible, the empty (( )) would be used. A closed set of non-speech sounds (e.g. laughter, coughing, etc...) is also transcribed and is enclosed in [ ] brackets. Partial words are transcribed, with a - marking the missing part either in the beginning or the end of the word (e.g. IN- might be the first part of the word INDIVIDUAL). The Fisher transcription statistics are summarized in Table 5.1.

It is not obvious how a partial word should be pronounced: BI- in BIT is different from BI- in BITE. However, partial words are commonly involved in repetition disfluencies, and the full word is often repeated completely within a few words of the partial word. In that case it is usually reasonable to repair the word fragment as part of the nearby full word. Letting *nearby* mean to

be within 6 words seems to give reasonable results. The partial words in the transcription of the entire corpus were repaired this way. Coverage statistics are given in Table 5.2.

	count	**percent**
total non-empty utterances	2223159	
total uncertain words or phrases enclosed in (( ))	283935	
total unique words or partial words	64924	100%
unique words occurring once in the corpus	23192	35.72%

Table 5.1: Some statistics on the transcriptions of the Fisher corpus.

	words	**tokens**	**coverage**
all words in corpus (including uncertain words)	79742	21905137	100%
singleton words	32703	32703	0.1492%
all words not in the CMU dictionary	39311	822146	3.76%
total non-speech markers (enclosed in [ ]) e.g. [LAUGH]	11	559629	2.555%
partial words (starting or ending in -)	21552	153098	0.6990%
repaired partial words		101550	0.4636%
whole words not in CMU dictionary	16654	103244	0.47%
multi-words not in CMU dictionary e.g. ANTI-FRENCH	1094	6354	0.0290%

Table 5.2: Coverage for transcriptions of the Fisher corpus with partial words repaired.

5.4.2 The phone set

The phone set is the 39-phone set used in CMU dictionary with stress information stripped and augmented with non-speech sounds: `SIL NOISE LAUGH BREATH COUGH LIPSMACK SIGH SNEEZE`. Additionally there is an end-of-word marker `EOW` to make word transition bookkeeping easier, for a total of 48 phones. Each phone consists of a sequence of 2 or 3 sub-phone states, with the exception of EOW, which has only 1 state. The complete phoneset is given in Appendix A.1.

5.5 Generating the Fisher pronunciation dictionary

The multi-pronunciation CMU dictionary [91] with all stress information stripped from it was used as the foundation for the Fisher pronunciation dictionary. Table 5.2 shows that in addition to the word fragments, a number of whole words used in Fisher transcriptions are missing from the CMU dictionary. These rare words and fragments cover only 1.18% of the corpus, and are not likely to be included in a recognition vocabulary. However, they can be used in training if phonetic transcriptions for these words were generated. The missing transcriptions were automatically generated in one of the following ways, in the order of preference:

1. If the word is in the CMU dictionary, use that pronunciation.
2. Otherwise, if the word is an acronym (matches `([A-Z]_)*[A-Z]`) or contains numbers, do direct letter-to-phones replacement, so `401K` ⇒ `F AO R OW W AH N K EY` and `I_B_M` ⇒ `AY B IY EH M`.
3. Otherwise, if the repaired partial word in the CMU dictionary, determine the fragment pronunciation by using the HMM model to force-align the letters of the repaired word against it's phonetic pronunciation.
4. Otherwise, attempt to use HMM model to decode the phonetic pronunciation.
5. If all of the above fail, manually phonetically transcribe the word.

Only 134 words reached step 5 above, and a manual transcription was required, either because some illegal alphabet was used (e.g. `M&M`) or the HMM model was unable to find a suitable sequence of hidden phonemes, because there were more letters than phones and skips were impossible (e.g. `C_D-ROM`).

From a cursory check, forced alignment is almost always correct, while decoding works mostly, but fails in some cases where a single letter is observed from multiple phones (e.g. `X` in `ASPHYXIATES` ⇒ `AH S F AY K AY AH T S`).

5.6 Analysis

The dictionary contains different words which are pronounced as an identical sequence of phonemes. These words are homophones in the specific pronunciation representation used by the recognizer - they need not be homophones in a more nuanced pronunciation representation. The speech recognizer will have problems with homophones, relying only on the language model to distinguish between them. In this section, we measure perplexity seen by the recognizer on the Fisher corpus due to homophones.

For the Fisher corpus, the perplexity of word W given phone sequence S is calculated from the transcriptions as follows. Let $p(W|S = s)$ be the probability of a token being the word W, given that the token is pronounced as s. This can be estimated from the data as

$$p(W|S = s) \simeq \frac{C(W)}{C(s)}$$

where $C(W)$ is the count of tokens of word W, and $C(s)$ is the count of all words pronounced as s. This works if each word has a single pronunciation.

If a word W has M pronunciations, and occurs in the corpus N times, we assume a uniform distribution over pronunciations, so W is pronounced with any given pronunciation $\lfloor \frac{N}{M} \rfloor$ times in the corpus. Using a uniform distribution here is sub-optimal, as the pronunciations per word tend to have something like a Zipf's law distribution [25], but certainly not a uniform distribution. However, we have no basis to make a better guess without phonetically transcribing the tokens for the multi-pronunciation words. The word perplexity given a pronunciation calculated with this assumption will over-estimate the true word perplexity.

Let $H(W|S = s)$ be the entropy of the distribution $p(W|S = s)$. Then

$$H(W|S) = E_{p(S)}[H(W|S = s)]$$

where

$$p(S) = \frac{C(S)}{\sum_S C(S)}$$

$H(W|S)$ was estimated using the multi-pronunciation dictionary and the Fisher corpus with partial words repaired as described in Sec. 5.4.1 and the key measurements are given in Table 5.3.

	token count	**token %**
Total Tokens	26351455	100%
Tokens affected by the multi-pronunciation redistribution	7866152	29.85%
$H(W\|S)$ ignoring the <S> and </S> markers	0.2275 bits	
$2^{H(W\|S)}$ perplexity (words per pronunciation)	1.1708	

Table 5.3: The perplexity of the word given a pronunciation using the distribution estimated from the Fisher corpus, and using a multi-pronunciation dictionary. Tokens affected by the multi-pronunciation redistribution are the tokens with pronunciations that could have been generated by a multi-pronunciation word. These are the tokens affected by the 'uniform distribution over multiple pronunciations' assumption.

A perplexity of 1.17 is equivalent to 17% of the tokens from typical conversational speech having two pronunciations. Those 17% of the tokens pronounced in isolation cannot possibly be identified uniquely by an ASR system using a typical multi-pronunciation dictionary because the ASR system cannot learn consistent acoustic distinctions even if they exist.

The semantics of the conversation are important for identifying those ambiguous 17% of tokens, but there are also some indications that uncaptured acoustic distinctions do exist. As mentioned in Section 2.2.3, words having the same phonetic transcriptions differ in average duration depending on the identity of the word. The above observation combined with the high value of $2^{H(W|S)}$ is the motivation to explore explicit pronunciation models with more specialized sub-word units as well as implicit pronunciation models (Chapters 6 and 7).

There are also some questions about the pronunciation consistency and the maximum achievable accuracy of the explicit pronunciation model. Should the word EXCITE be pronounced as /IH K S AY T/ which is suggested by the CMU dictionary or /EH K S AY T/ as suggested by our HMM pronunciation model? It is likely that for any reasonable choice of features and context, the Bayes error for a symbolic pronunciation model will be greater than 0. The pronunciation ambiguity is perhaps better handled by the implicit pronunciation models.

Chapter 6

EXPLICIT PRONUNCIATION MODELING EXPERIMENTS

In this chapter, we report on explicit pronunciation modeling experiments in which the pronunciation dictionary is manipulated to improve the recognition of error-prone words. These are not the first experiments of this kind: Section 2.1 describes some of the previous experiments, only scratching the surface of the body of work on explicit pronunciation modeling. The same section presents some warnings for explicit pronunciation modeling in difficult recognition tasks such as conversational speech, where the starting WER is relatively high. In particular, for high WER recognizers, any kind of explicit pronunciation modeling should keep pronunciation perplexity close to 1. The pronunciation experiments described here heed this warning.

Unfortunately, keeping pronunciation perplexity low does not guarantee that the new pronunciation models will improve the WER over the standard pronunciation lexicon. Even though the experiments described here were designed explicitly to minimize WER, they yield only small improvements over the baseline recognizer.

The main idea of these experiments is to allow pronunciation variability without permitting more pronunciation perplexity. Context-dependent phoneme sequences which are frequently misrecognized are identified, and the worst offenders are modeled by their own acoustic models. This kind of gradual addition of corrective units would allow error-prone homonyms to be distinguished, and it could automatically introduce multi-words, if the mistake context is allowed to span word boundaries.

Section 6.1 describes how the new units are selected in general. Section 6.2 describes the characteristics of the mistakes typically made by our baseline recognizer. Section 6.3 describes the evaluation strategy. Sections 6.4 and 6.5 describe the new units introduced to correct the baseline recognizer and compares the corrected system to the baseline in terms of WER and number of model parameters.

6.1 Mistake-based unit selection

The new units are constructed by first finding the phone sequences which are often misrecognized during decoding by the baseline recognizer. The baseline triphone recognizer is run twice on the training data, once in the forced alignment mode to obtain a phonetic reference labeling of frames, and once in the decoding mode using the language model. The two generated transcriptions are compared frame by frame to find the mistake tokens made by the baseline recognizer.

A *mistake token* is defined to be the longest possible contiguous sequence of incorrect reference phone tokens. A reference phone token is correct if and only if at least one frame of those spanned by the reference phone token has the hypothesized label equal to the reference label. A single matching frame is sufficient, because it means that the phone will show up in the hypothesized transcription. In this discussion, a *mistake* means a mistake type. A mistake is only a sequence of phones - some mistakes occurring in the corpus are mistake tokens, but others mistakes may actually be recognized correctly. It is still possible for a correct hypothesized phone sequence to generate a mistake if none of decoded frames line up with the force-aligned reference transcription, but these situations should be rare.

An example of a mistake token is shown in Figure 6.1. An /EOW/ phoneme at the edge of a mistake is excluded from the mistake. In general, a mistake can span word boundaries and correcting such a mistake would involve introducing a multi-word into the language model.

The *longest* contiguous sequence of incorrect phones is defined as a mistake because a mistake is often caused by co-articulation between a sequence of phonemes. With this definition of a mistake, the short mistakes which are also substrings of longer mistakes will not be double-counted and will not be preferred over longer mistakes.

A possible problem is that the language model may spread the mistake range beyond the phones distorted through co-articulation. In the experiments we present here this is not an issue because we do not consider mistakes spanning word boundaries.

A mistake context is whatever information we suspect makes the mistake acoustically consistent. This could be global information, such as rate of speech, or local information such as whether the word or syllable is stressed.

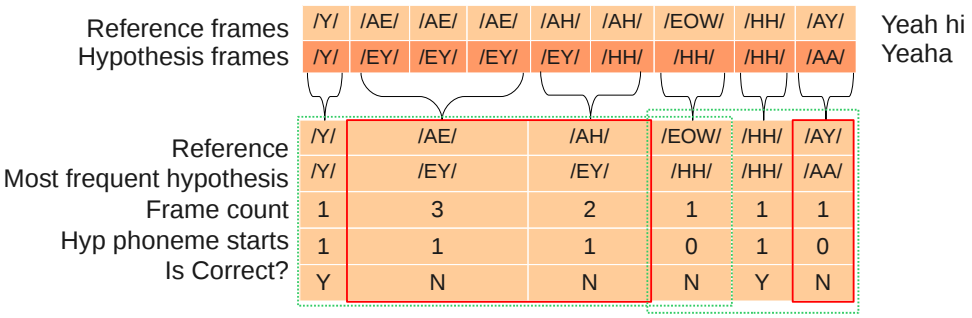

Figure 6.1: Example of extraction of mistake tokens and their contexts from the reference and baseline transcriptions. The two mistake tokens are highlighted in red, and the dashed green boxes denote the contexts for the mistake tokens. Each reference phone in a mistake is also annotated with the frame counts, the most frequent hypothesized phone, and hypothesized phoneme counts, which are used for designing the new units.

In our simple experiments, the *mistake-in-context token* is defined as the mistake token plus the phonemes surrounding it out to the enclosing word boundary. The *mistake-in-context* denotes the type, analogous to the definition of mistake. The context also includes the identity of the spanned words, so that two homonym mistakes-in-context are distinct.

To select new units for inclusion in the model, we need to count the mistakes-in-contexts. Figure 6.2 shows a single mistake, its mistakes-in-context and the correct and incorrect counts. Sufficiently frequent mistakes-in-contexts are candidates for new context dependent units.

	Word	left	mistake	right	Correct		Incorrect	
					count	hyp phone starts mean(var)	count	hyp phone starts mean(var)
unit			*AH N*		*2083*	*1.97(0.65)*	*3675*	*1.49(1.09)*
contexts	AN	EOW	AH N	EOW	87	2.43(0.87)	486	1.56(1.30)
	THAN	EOW DH	AH N	EOW	135	2.13(0.76)	268	1.74(1.23)
	ISNT	EOW IH Z	AH N	EOW	26	2.38(0.70)	103	1.37(1.09)
	COULDNT	EOW K UH D	AH N	EOW	21	1.90(0.56)	79	1.15(0.76)
	FUN	EOW F	AH N	EOW	58	1.98(0.95)	112	1.96(1.79)
	DEFINITELY	EOW D EH F	AH N	AH T L IY EOW	85	1.93(0.11)	136	1.43(0.97)
	...	...	AH N	...	...	...	...	...

Figure 6.2: The mistake `AH N`, and its contexts, sorted by the difference (incorrect counts)-(correct counts). The hypothesized phoneme starts are counted during the period of time spanned by the reference labels `AH N`. The count of hypothesized phone starts during correctly and incorrectly recognized tokens of `AH N` can be used to determine the number of states in the new model.

Figure 6.2 also shows hypothesized phoneme starts occurring within time spanned by the reference phoneme string. The incorrect tokens of this mis-

take are phoneme strings that are shorter on average than the phonemes strings of correct tokens, suggesting that reduction is partly a cause of the mistake.

A separate unit model can be trained for each mistake-in-context, or models with different contexts can be tied based on their acoustic similarity, as with triphones, to improve training coverage.

6.2 Kinds of mistakes

We first analyze the kinds of mistakes made by our baseline recognizer. We perform recognition of the first 100,000 utterances of our training data with the baseline recognizer, and find mistakes-in-context (as defined above) which occur 5 or more times. For this purpose, the mistakes are allowed to span word boundaries, and the context extends from the mistake boundaries to the outmost word boundaries, as in Figure 6.2. There are 6606 such mistakes-in-context.

Some of the mistakes-in-context overlap, in the sense that if one mistake-in-context is replaced by a new unit, it interferes with a similar replacement of another mistake-in-context. For example, some tokens of the word "THAN" have mistakes in context `EOW DH [AH N] EOW` and others have `EOW [DH AH] N EOW`, and replacing the first precludes replacing the second. To resolve this, we sort all mistakes-in-context by mistake length, then context length, then by occurrence count, and greedily pick only the mistakes-in-context that do not conflict with any that were picked already. The longer mistakes and more specific contexts are preferred to minimize the number of conflicts. This prunes the mistakes-in-context down to 46% of the original 6606. Of the remaining mistakes-in-context, 55% are subwords, 17% are full words and 28% are multi-words. These remaining mistakes-in-context are candidate models considered for inclusion in the speech recognizer. If all of them were added as models, 23,000 new states would be added (compared to roughly 1,100 states in the baseline system).

6.3 Evaluation and parameter budget

Often an improvement can be tested by including it in an optimized baseline and testing it against the baseline without the improvement. In the experiments presented here this was not practical, because the existing methods of improving the baseline were not exhausted due to hardware limitations. For example, Tables 3.1 and 3.2 show that improvement is likely if we grow the number of Gaussians per mixture (GPMs) beyond the 64 used in baseline. More leaf nodes in the triphone DTs should also lower the WER. However, these larger models do not fit in the RAM of the machines in our compute cluster.

Since it is not possible to bottom out the WER with established techniques, an alternative evaluation strategy is to make the number of parameters equal in both the baseline and the experiment. Thus, in the augmented unit recognizers, the number of GPMs is chosen so that the number of trainable parameters is roughly equal (within a factor of 2) to number of trainable parameters in the baseline.

The model parameters budget is defined as the number of parameters in a model modified by the EM algorithm. This is dominated by the (total number of DT clusters)x(GPMs)x(1+2(observation vector dimension)), where the last term is the weight of the Gaussian plus the mean and diagonal variance vectors of the Gaussian.

6.4 Experimental methods

No matter how a mistake-in-context m is selected, it is difficult to predict whether modeling m with its own unit u will improve the WER over the baseline. If we initialize u from its component phonetic models and train the system the same as the baseline, the EM algorithm guarantees that the likelihood of training data will not decrease, but it says nothing about whether the model including u will reach a higher log-likelihood than the baseline.

We use a very simple (and simplistic) model to estimate the improvement due to including a new unit to the baseline model. The mistake-in-context m is selected to be modeled by an independent unit in a greedy way, to minimize the classification error on the training data. Let $\mathcal{M}$ be a set of compatible

mistake-in-context candidates, selected as described in Section 6.2. We can partition the reference transcriptions, into a set of tokens $\mathcal{X}$, which consists of all correct and incorrect tokens of all types in $\mathcal{M}$, along with tokens of types not in $\mathcal{M}$ which are not being considered for replacement. Also, let $ref(x)$ be the reference phone string for the string of observations x, let $\lambda(x)$ be the baseline model's maximum likelihood phoneme string for x and define $\lambda_{(m)}(x)$ similarly for the baseline augmented with a separate unit modeling m. Then we should chose m^* to be modeled by a separate unit to maximize the expected improvement in the classification error:

$$m^* = \arg\max_{m \in \mathcal{M}} E_{x \in \mathcal{X}}[\lambda_{(m)}(x) = ref(x)]$$

where the predicate in [ ] evaluates to 1 if if true, and 0 otherwise.

To compute the expectation, we make some very simplistic assumptions. First we assume that including m in the model, affects only the tokens of m and of no other mistakes. Second, we simply assume that the replacement is going to correct a mistake token of m with some probability p and introduce a mistake into a correct token of m with probability q. q can be greater than 0 for two reasons: 1) if u models a very reduced phone sequence, it can become more confusable with other phone strings than the original phoneme string and 2) u can over-fit the smaller amount of training data. Let $\mathcal{X}_m \subset \mathcal{X}$ be the set of all correct and incorrect reference transcription tokens of type m. Then we can write

$$m^* = \arg\max_{m \in \mathcal{M}} \sum_{x \in \mathcal{X}_m} p[m \neq \lambda(x)] - q[m = \lambda(x)] \tag{6.1}$$

We would expect better performance if p and q were measured by actually performing the substitution and counting resulting errors in the identified segments. A suboptimal greedy search using this approach would require $O(MN)$ full recognizer tests, where M is the number of mistake candidates, and N is the number of new subword units we intend to create. But if we could efficiently train and test that many recognizers, we could directly greedily minimize the error without resorting to p and q probabilities at all.

Since p and q are unknown, we guess and set each to .5. With these values, Eqn. 6.1 reduces simply to picking the unit with the biggest difference

between observed mistake tokens and correct tokens.

The greediness of the algorithm comes from the way $\mathcal{M}$ is selected. Once $\mathcal{M}$ is known and given the assumptions, the selection of m^* is optimal.

Let $m_1, m_2, ...$ be the order in which m^* would be chosen. A batch of the top N^* units $m_1, ..., m_{N^*}$ are added to the baseline at the same time, where N^* is determined by the allotted parameter budget of S states:

$$N^* = \arg\max_N N \qquad s.t. \sum_{i=1}^{N} s_{m_i} \leq S$$

where s_{m_i} is the number of states used in model m_i.

One option is to set s_{m_i} based on the number of hypothesized phones during the mistake units. In almost all the cases, the incorrect tokens have fewer phoneme starts and greater variance than the correctly recognized tokens, similar to what's shown in Figure 6.2.

Alternatively, s_{m_i} could be set to the sum of the number of states in the phonemes that are replaced by m_i. The new states could be initialized from those component phonemes. This is what we do in our experiments.

At this point we can tie the parameters of model candidates that share the same mistake but have different contexts, in the same way that triphone models are tied. In our experiments we try two things: leaving all models untied, and tying all contexts with a shared mistake. Another (not yet tried) option would be to use a data-driven tree-structured tying algorithm, similar to the usual method used to tie triphones. With tied parameters, we are able to model more mistakes with separate units for a given budget, but these units will need to model more acoustic variability.

Even though 28% of mistakes-in-context span word boundaries, in our experiments we restrict the units to be word-internal. If we allowed multi-word mistakes-in-context, we would also have to compare against a baseline with multi-word pronunciations manually added to the dictionary.

Appendix A.2 lists all sub-word-units introduced into the model as described above. There are 69 units total, and 66 of them are whole word units. In this mode, the recognizer is simply using a combination of whole word models and phonetic pronunciation models. In all units, the number of hypothesized phoneme starts is on average lower for incorrectly recognized units than for correctly recognized ones.

The phone sequences are replaced by the $m_1, ..., m_{N^*}$ units in the pronunciation dictionary, and the recognizer is retrained from scratch.

6.5 Experimental results

We compare the WER improvement on models with roughly the same number of parameters, where the two competing approaches are: 1) growing the number of Gaussians per mixture (GPMs) and 2) adding new sub-word units. A third obvious way to grow the baseline model is by allowing more clusters in the triphone decision tree clustering. We tune the DT clustering parameters on the baseline (with the tuning experiments presented in Table 3.1) and fix the DT clustering parameters for the baseline and and the experiment recognizers.

Because our triphones are word internal, the 66 whole-word models are context independent. Even though there is nothing preventing the three newly introduced sub-word units from being context dependent in the same way that triphones are, all the contexts are tied during DT clustering, and the sub-word units are also context independent.

In each experiment in Figures 6.3 and 6.4, the only forced differences are the number of pronunciation units and the number GPMs. All other parameters (such as the thresholds for DT-based triphone-clustering) are the same in the experiment and the baseline. The tests are performed on 2000 utterances.

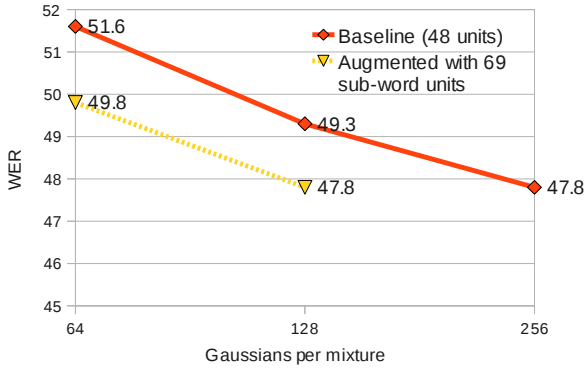

Figure 6.3: The WER as a function of Gaussians per mixture (GPM) for the baseline recognizer (48 phonemes) and the tested system (48 phonemes + 69 additional pronunciation units derived from baseline recognizer's mistakes).

Figure 6.3 shows that the effect on WER of increasing the number of pronunciation units is largely complimentary to increasing the GPM. When the number of GPMs is low to start with (64), doubling the number of GPMs is more fruitful than introducing new units. When the starting number of GPMs is higher (128), doubling the number of GPMs is just as effective as introducing new units. We know that growing the GPMs yields progressively smaller marginal improvements to the WER (See Table 3.2), and with sufficiently large models, including context dependent units should be another source of WER improvements.

Figure 6.4 shows that at a parameter budget between 130k and 250k parameters, the parameter budget is better spent on increasing the number of units, than on growing the GPMs. The same WER of 47.8% is reached by both recognizers, while the model with extra units is smaller by 20%.

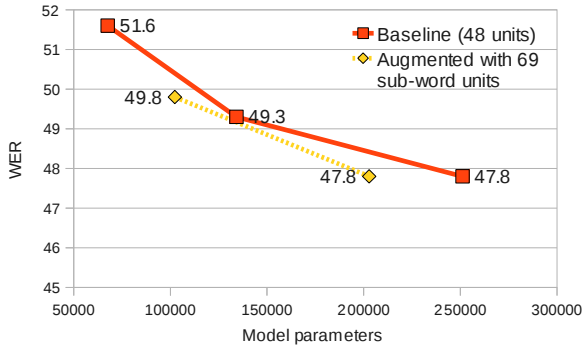

Figure 6.4: The WER as a function of number of parameters in the model, for the baseline recognizer (48 phonemes) and the tested system (48 phonemes + 69 additional pronunciation units).

We ran another experiment where all the contexts for each mistake were tied, and the same model was used within all mistakes-in-context that had the same mistake. Since most of the context-dependent models have the form `EOW [MISTAKE PHONES] EOW`, the context dependent models replace the phoneme strings only in whole words, while context independent models replace the phoneme strings in some word-internal contexts. This approach worked poorly, giving a 50.0% WER at 124k model parameters, underperforming the context-dependent units (49.8 WER at 102k model parameters). We suspect that the reason for this is the new units have very different acoustics which depend on the context, but we are modeling them with the same model.

Chapter 7

IMPLICIT PRONUNCIATION MODELS

In this chapter, we describe experiments with what we call *implicit pronunciation modeling*: the pronunciation modeling that takes place within the acoustic model when long duration acoustic context is made available to it. With this approach, the acoustic model is treated as a black box in the sense that it is unknown which aspects of pronunciation are captured by the acoustic model. This contrasts with explicit pronunciation modeling approaches described in Chapter 6, where pronunciation is manipulated at the symbolic level, by changing the pronunciation dictionary, or introducing new acoustic model designed specifically for some specific part of word or multi-word.

Some of the previous approaches that fall under the implicit pronunciation modeling category are described in Section 2.3. In this chapter, we present two new related approaches. In both approaches, we rely on the distribution $p(q|o)$ of the phone q given the observation o.

In the first approach (Section 7.1), we use $p(q|o)$ to measure the recognizer's frame prediction confidence and describe how high-confidence frames can replace neighboring lower-confidence frames, thereby lowering the WER. Under some circumstances, replacing a string of similar high-confidence frames with just a single representative frame also lowers the WER.

The second approach (Section 7.2) deals with observation frame classification, rather than with full recognition. In it, we show that filtering the phone posteriors $p(q|o)$ with a very long smoothing window improves the frame classification accuracy across almost all phones, not just the long duration vowels and silences.

7.1 Simple segmental tandem model

In this section we describe how Multi-Layer Perceptrons (MLPs) can be used to find continuous high-confidence segments of observations. The frames in these segments can be replaced by copies of some 'representative' frame, resulting in a lower word error rate (WER) on conversational telephone speech when compared to a triphone baseline using PLP+MLP tandem features. A frame error analysis shows that this kind of segment replacement benefits almost all phonemes, while being particularly beneficial to silence detection, nasals and vowels misrecognized as plosives and consonants misrecognized as schwa-like vowels.

Hidden Markov models (HMMs) commonly used for speech recognition assume that the observations are conditionally independent given the hidden state. The hidden state is typically a phone, a triphone or some other sub-word unit. The conditional independence assumption does not hold for speech signals, and multiple approaches have been introduced to address this problem.

One such approach is the 'tandem' model described in Sections 2.3.1 and 3.3.6. In short, a multi-layer perceptron (MLP) gives the distribution over possible phonemes q given the observations neighboring the frame i:

$$MLP(i) = p(q|o_{i-n}, ..., o_{i+n})$$

A suitably transformed $p(q|o_{i-n}, ..., o_{i+n})$ can be concatenated with the traditional PLP to give PLP+MLP tandem observation for frame i.

The success of the tandem models is partly attributed to their relaxing of the assumption of conditional independence between neighboring observation frames. Another way to address this problem is with segmental models [92, 59] in which a hidden state emits an observation of varying duration. In this experiment we combine tandem observations with a segmental model. The segmental model we use is a very simple 'sample and hold' model used in a 1-pass recognizer, while the models described in [59] model intra-segment acoustic dynamics and the 2-pass recognizer in [92] considers multiple hypothetical segments simultaneously.

It has been observed that the frames where MLP predictions are confident are classified more accurately and tend to be phone-central [93]. Let S be

a contiguous segment of frames with high-confidence MLP predictions, all predicting the same phone. We might replace all frames in S with the frame $i^* \in S$, where i^* has the highest MLP confidence, or lowest entropy: $i^* = \arg\min_{i \in S} H(MLP(i))$. This *segment replacement* of all $i \in S$ with i^* makes the decoder more likely to label the segment the same as the high-confidence MLP prediction. An added benefit is that, if a high confidence segment really does belong to the same phoneme, the segment can be compactly represented as a single frame, and so reduce computation during training and testing.

The approach is similar to the one presented in [94], where recognition is performed on variable frame rate observations. In this work, a frame is sampled from speech only if it sufficiently different from preceding frames according to the SNR-weighted energy distance. This method selects more frames where the speech is changing rapidly. Large WER reductions are reported on noisy speech, when the average variable frame rate is set to equal the baseline fixed frame rate. The authors also report good voice activity detection results with this method.

In the next section we describe the segmentation procedure in more detail. In Section 7.1.2, we test this idea on the Fisher Corpus and compare the results to a tandem triphone baseline system and in 7.1.3, we analyze the source of the improvement.

7.1.1 The segmental model

We segment the original sequence of observations using the same MLP classifier used to generate the PLP+MLP observation features (see Sections 2.3.1 and 3.3.6). The MLP is trained on the first 80% of the conversations of the Fisher Corpus.

Segmentation

We partition the sequence of observations O into two sets: The high confidence set O_h and the low confidence set O_l. O_h consists of those frames for which the MLP predicts a single class with probability higher than some threshold τ, so $O_h = \{o_i; \max(MLP(i)) > \tau\}$. To keep the segmentation unambiguous we only test values of τ in the range $0.5 < \tau < 1$. The low confidence set is the remaining frames: $O_l = O \setminus O_h$. We define a *segment* as

a contiguous sequence of high-confidence frames belonging to the same class. Each observation in O_l is a segment by itself.

Let the segment S consist of T_S consecutive frames $(m_1, \ldots, m_i, \ldots, m_{1+T_S-1})$. A representative frame m_{i^*} for the segment S is selected to be

$$m_{i^*} = \arg\min_{m_i \in S} H(MLP(m_i)) \tag{7.1}$$

where $H(X)$ is the entropy of the distribution X.

If we assume that observations in S were generated from some constant hidden state q, the probability contributed by the observations of the segment S is

$$p(S|q) = \prod_{i=1}^{T_S} p(o_{m_i}|q)$$

If the summary frame m_{i^*} is similar to all other frames in the mth segment, we can make the following approximation

$$\prod_{i=1}^{T_S} p(o_{m_i}|q) \approx p(o_{m_{i^*}}|q)^{T_S} \tag{7.2}$$

During training we can make sure that no segment spans a phoneme boundary by breaking up the segment, thus ensuring that the hidden state q is indeed constant. Then we can safely use this approximation, and replace all the frames of segment m with the single exponentiated representative frame, thus reducing the computation needed during training time.

Choosing the confidence threshold

While the main experiments presented in this section are on the Fisher corpus, we have also tried this segment replacement technique on the SVitchboard 500-word recognition task [44] using a context-independent articulatory-feature based recognizer [43]. We were able to experiment more quickly on this smaller corpus, and we used it to tune the confidence threshold τ to minimize the WER. The WER as a function of τ is presented in Figure 7.1.

Using Eq. 7.2 to summarize a segment as a single frame can yield significant computational savings, where the exact savings depend on the audio data. The SVitchboard utterances are bounded by relatively long silences, which

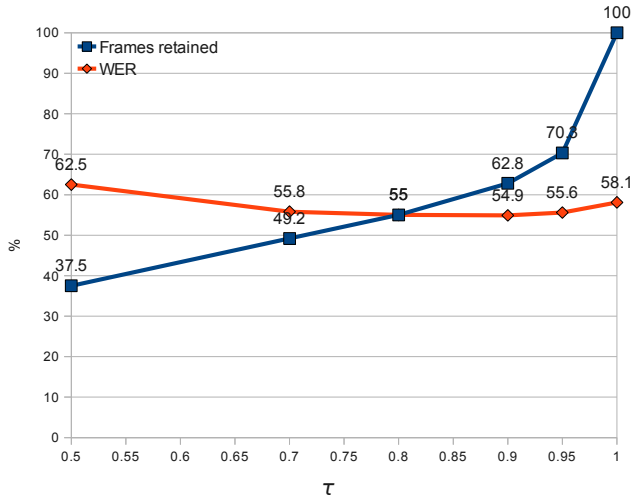

Figure 7.1: Tuning the confidence threshold. At each level of τ, segments with classification probability greater than τ are replaced by a single representative frame. The "frames retained" plot specifies the percentage of frames remaining after replacement. The WER is reported on the 500-word SVitchboard development set (D-set). The baseline (no frame summarization) is the $\tau = 1$ point.

on average make up 51% of the total utterance. At the optimal $\tau = .9$, 37% of the frames were summarized away.

The Fisher corpus already has the speech extracted from the original recordings. The silences at utterance boundaries make up only 13% of all the frames and only 6.7% of the frames were dropped at $\tau = .9$.

The τ chosen on the SVitchboard corpus is likely sub-optimal on the Fisher corpus, because the speech in SVitchboard and Fisher corpora is different, the recognizers are significantly different, and the phoneset used for segmentation was different from the one used for recognition. However, even with these caveats, these simple segmental models with $\tau = .9$ outperform the baseline system, as described in the next section.

7.1.2 Experiments

Methods

The baseline systems for these experiments are described in detail in Chapters 3 and 4. In short, the two baselines use either the PLP features or the PLP+MLP tandem features. A word internal triphone recognizer is used,

with its acoustic model trained on 20% of the Fisher data. The language model is fixed in the baseline and in experimental systems.

There are three different ways in which we can use the MLP segmentation of the observation frames. We can leave the frames unchanged (the *no* choice in results Table 7.1). The baseline systems do not change the frames in either training nor testing.

We can also perform *segment summarization* (SS), where each segment is replaced with a single frame using the approximation in eq. 7.2. The confidence threshold used is $\tau = .9$ as determined in Section 7.1.1. Finally, we can do *segment replacement* (SR in Table 7.1) where each frame in the segment is explicitly replaced with a copy of the representative frame. The main difference between SS and SR is that the HMM observing an SR stream can choose to place a phoneme boundary in the middle of the segment (i.e., between two identical frames), whereas an HMM observing an SS system can not. The HMM observing the SR also accumulates the state transition penalty for transitioning between intra-segment frames, while there is no such intra-segment penalty in the SS stream.

When doing SS on the training data, we have the phone boundaries available from forced alignment, and we ensure that a segment does not span multiple phones by breaking the segment up at the boundary. The phone boundaries are not available for test frames, so in this case we simply assume that boundary-spanning high-confidence segments are rare, which is a reasonable assumption if we believe we have a very good MLP classifier.

Results

The WERs of the baseline recognizers using only the PLP observations and the PLP+MLP tandem observations are reported in rows 1 and 4 of Table 7.1. Note that the baselines are identical to the segmental model with $\tau = 1$. The more interesting combinations of segment summarization and segment replacement applied to training and test data are also reported in Table 7.1.

Table 7.1 shows SR and SS harming the WER when using only PLP observations, while SR and SS using PLP+MLP observations improves the WER. This suggests that using MLP outputs as observations in more complex segmental models can be fruitful.

row	seg. in train	seg. in test	test WER	dev WER
PLP features				
1	no	no	58.4%	51.8%
2	SS	SS	59.7%	53.4%
3	SS	SR	59.2%	52.6%
PLP+MLP features				
4	no	no	54.8%	47.7%
5	SS	SS	53.9%	47.5%
6	SS	SR	53.7%	47.1%
7	SS	no	53.3%	46.3%

Table 7.1: Recognition WERs on the test set (20,000 utterances) and the development set (2000 utterances). On the test set, all WER differences are statistically significant well beyond the $p < .001$ level using the matched-pairs test [77]. Segment Summarization (SS), Segment Replacement (SR) and no segmentation (no) are tried on combinations of training and decoding observations. Rows 1 and 4 are considered baselines.

When we perform SR on the test data, we need not make the assumption that high-confidence segments rarely span boundaries. The difference in WER between rows 2 and 3 and rows 5 and 6 shows that SR is better than SS on test data.

Error analysis suggests two explanations for this result. First, it turns out that our MLP classifier occasionally violates the assumptions of SS by creating boundary-spanning high-confidence segments. Pairwise analysis of mistakes shows that violating the assumption introduces 3 times as many mistakes as it fixes, so doing segment summarization on the test data is a bad idea for the Fisher data, although it was helpful on the SVitchboard task, where there were long periods of noisy silence at utterance boundaries.

It is also possible to perform training and segment summarization iteratively, if we assume that the high confidence frames also have the same or higher likelihood than the frames they are replacing:

$$p(o_{m_{i^*}}|q) \geq p(o_{m_i}|q) \tag{7.3}$$

where m_i and m_{i^*} are defined as in Eqn. 7.1. The inequality 7.3 is an assumption, because the representative frame m_{i^*} need not be the frame with

the highest conditional likelihood in the segment. So

$$\arg\min_{m_i \in S} H(MLP(m_i)) = \arg\max_{m_i \in S} \left(\max_q p(o_{m_i}|q) \right)$$

is not necessarily true because o_{m_i} is a PLP+MLP feature vector, but it often should be.

Then, at each iteration j, we have

$$\log p(O_{j-1}, W; \lambda_{j-1}) \leq \log p(O_{j-1}, W; \lambda_j) \tag{7.4}$$

$$\leq \log p(O_j, W; \lambda_j) \tag{7.5}$$

where λ_j and O_j are the model parameters and training observations at iteration j, and W is the word sequence. The inequality 7.4 comes from the EM algorithm, and 7.5 comes from the assumption.

On the other hand, by changing O_j at each iteration, the training data is no longer coming from the same distribution as the test data, and the accuracy may go down for this reason.

We test this idea by doing baseline training up to 32 GPMs and splitting the Gaussians to get 64 GPMs. At this point, we EM-converge the baseline system on the original training data, and EM-converge the test system on the Segment Summarized (SS) training data. The training data log-likelihood is indeed higher in the test system than in the baseline (the assumption holds true in our experiment), but the WER is slightly worse in the test system than in the baseline.

7.1.3 Error analysis

It is illuminating to see which pronunciation aspects are better modeled by the high-confidence segment replacement. In this section, we analyze phoneme frame error rates to determine where the improvement in the WER is actually coming from.

The first question is which phonemes actually contain high-confidence regions. Figure 7.2 shows that roughly 24% of high-confidence frames occur in non-speech 'phonemes' such as /SIL/ (silence), /EOW/ (end of word) and /LAUGH/ (laugh). In this case, replacing a high-confidence non-speech segment by a single representative frame is the same as preprocessing the observations through a voice activity detector (VAD). This is consistent with

the large number of frames removed on the SVitchboard utterances which contained long silences at the utterance boundaries.

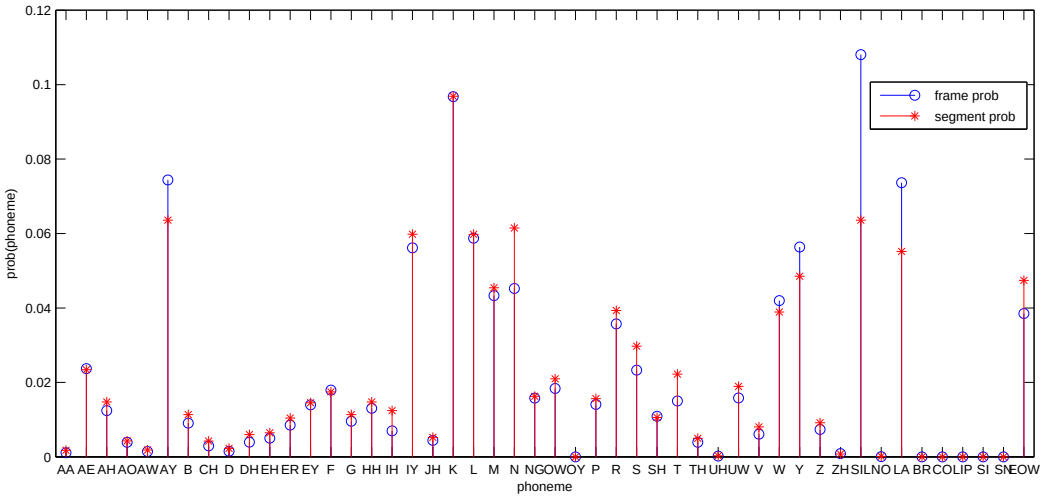

Figure 7.2: Summarized segments, broken down by phoneme. Blue circle is the distribution $p(f \in phoneme | f \in S)$ where f is a frame and S is a high-confidence segment. The red star is the distribution $p(S \in phoneme)$. The confidence threshold is $\tau = .9$.

We also compare the number of frame errors made by the baseline system (row 4 in Table 7.1) versus those made by our SS system (row 7). A mistake is a disagreement between the maximum-posterior label computed by the recognizer, for any particular frame, and the label assigned by forced alignment. The forced alignment was performed using the recognizer in row 7 of Table 7.1, since that recognizer gave the lowest WER.

When evaluating frame errors, it is possible that the system in row 7 of Table 7.1 will have an unfair advantage against the baseline, because the 'correct' frame labels were force-aligned by the same system. In any case, the phoneme transition boundaries are not important, as long the sequence is correct. We take this into account by ignoring frame mistakes due to differences of phoneme boundaries, as in Table 7.2.

We compare the frame mistakes and corrections made by row 7 to the baseline row 4 of Table 7.1. For each frame that truly belongs to phoneme p, we look at the difference of $\Delta_p = corrections_p - mistakes_p$ (see Figure 7.3). The biggest per frame improvement occurred on the 'silence' phonemes `/EOW/` and `/SIL/`, followed by the phones `/N/`, `/AY/`, `/K/` and `/AH/`. It is somewhat surprising that the short duration `/K/` phoneme was one of the biggest ben-

force-aligned frames	AAAAAB	BAAAAA
decoded frames	ABBBBB	BBBBBA

Table 7.2: The mistakes due to phoneme boundary differences between the force-aligned reference labels and labels predicted by the recognizer are ignored. The frame is marked correct if the phonemes on both sides of the boundary are predicted correctly for at least one frame. Thus, all the phonemes above are marked as correct.

eficiaries of segment replacement. The phonemes /HH/, /S/, /CH/ and /OW/ had more mistakes than corrections as a result of the segment replacement, and all other phonemes had more corrections than mistakes.

If we normalize the net improvement Δ_p by the total number of frames per phoneme N_p, so we do not penalize improvement of short phonemes, the order of the top beneficiaries of segment replacement changes (See Figure 7.4). The silence phonemes are no longer the ones most affected by segment replacement.

We further break down the normalized improvement to see what kind of phonetic confusions are corrected and which are introduced. Figure 7.5 shows a kind of confusion matrix M where the entry M_{pq} in row p column q is

$$M_{pq} = \frac{B_{pq} - W_{pq}}{N_p}$$

and where B_{pq} is the number of frames corrected (Bettered) by changing their label from phoneme q to phoneme p, and W_{pq} is the number of frames made incorrect (Worsened) by changing their label from phoneme p to phoneme q.

In Figure 7.5, the major effects of the segment replacement are concentrated in a small number of phoneme pairs. Additionally, large values in M_{pq} of the matrix usually correspond to a non-trivial value with opposite sign in M_{qp}. This suggests that the segment replacement tends to make a phoneme p sound like q in a confusable pair pq, regardless of whether it's the right thing to do, although on average more corrections than mistakes are introduced.

To focus on the largest differences introduced by the segment replacement, we highlight only those M_{pq} and M_{qp} entries where $|M_{pq}+M_{qp}| > .009$. These 42 entries are plotted in Figure 7.6, and account for 22% of $\sum_{pq} |M_{pq}|$. These phone pairs are also listed in Table 7.3.

Two phenomena describe most of the biggest improvements due to segment

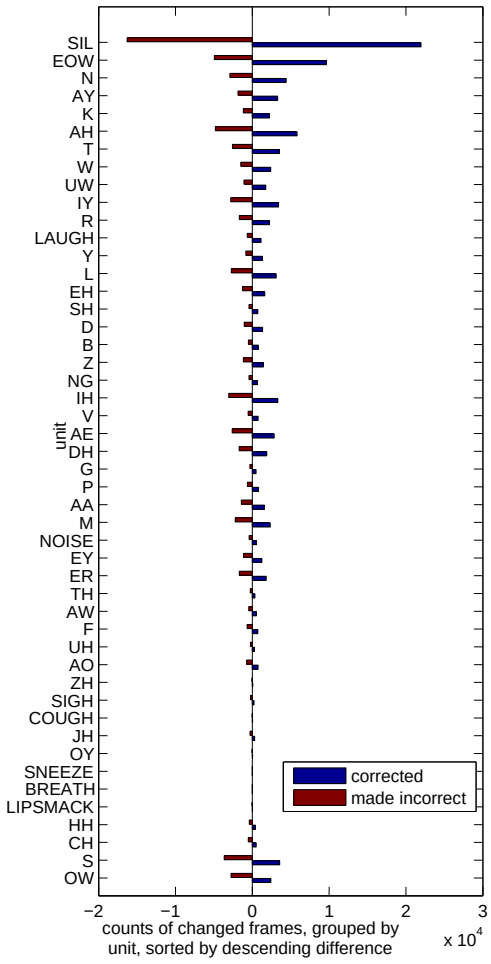

Figure 7.3: For each frame that truly belongs to phoneme p, the number of frames corrected ($corrections_p$) and made incorrect ($mistakes_p$). The phonemes are sorted in descending order of $\Delta_p = corrections_p - mistakes_p$.

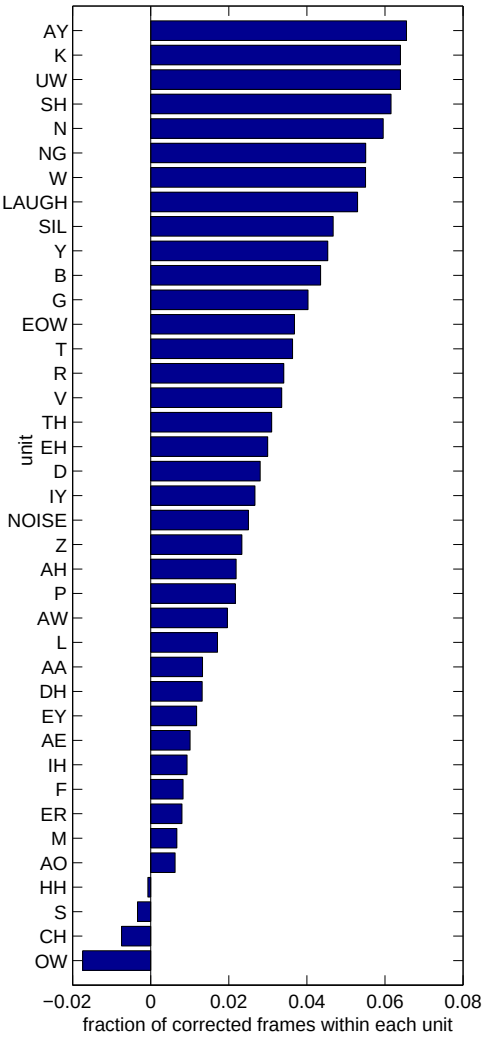

Figure 7.4: The normalized improvement $\frac{\Delta_p}{N_p}$ where Δ_p is defined as in Figure 7.3, and N_p is the total number of frames for each phoneme. The least frequent phonemes accounting for a total of 1% of the frames have been omitted.

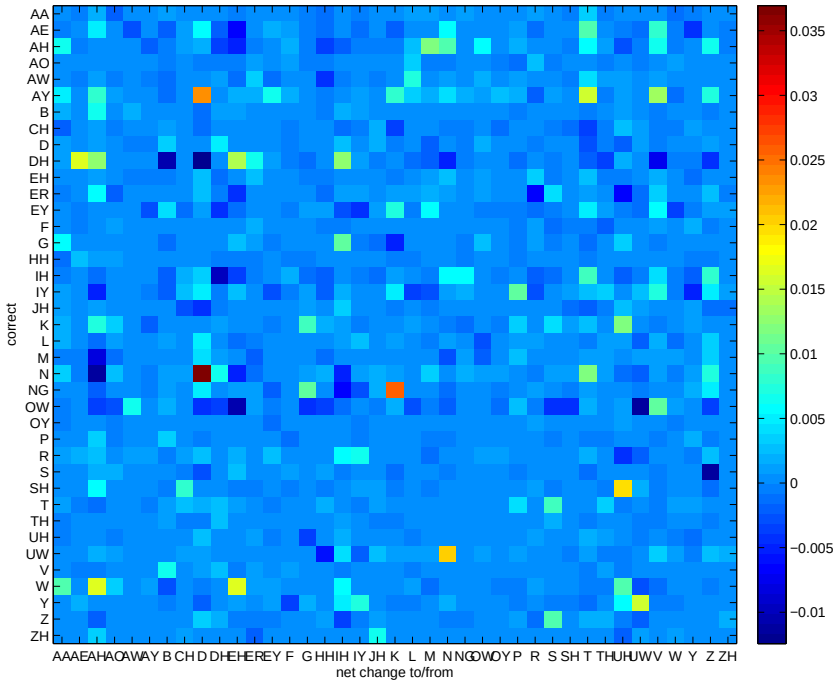

Figure 7.5: The confusion matrix M_{pq}: Normalized corrections and mistakes introduced as a result of segment replacement.

Type	**Changes**	
Plosives repaired	/D/ → /N/	/T/ → /N/
	/D/ → /AY/	/T/ → /AY/
	/G/ → /NG/	/K/ → /NG/
	/P/ → /IY/	/T/ → /IH/
Schwa-like vowels repaired	/AH/ → /W/	/EH/ → /W/
	/UH/ → /K/	/EH/ → /DH/
	/UH/ → /SH/	
Other corrections	/N/ → /UW/	/DH/ → /AE/
	/V/ → /OW/	/V/ → /AY/
	/UW/ → /Y/	
Mistakes	/OW/ → /UW/	/OW/ → /EH/
	/B/ → /DH/	

Table 7.3: The biggest phone pair changes from Figure 7.6.

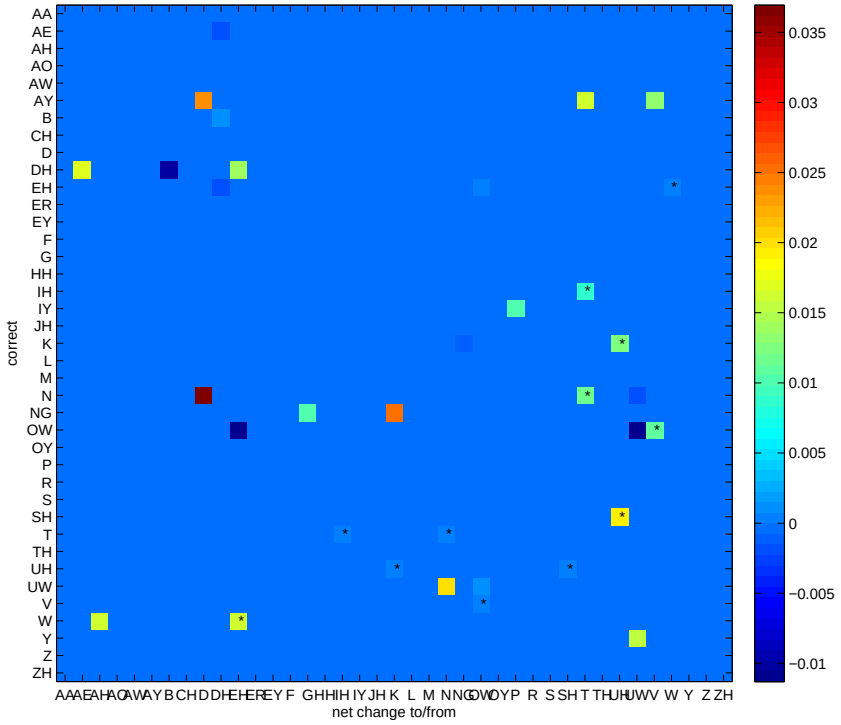

Figure 7.6: The most significant pairwise corrections and mistakes introduced by segment replacement. They are only those entries M_{pq} from Figure 7.5 where $|M_{pq} + M_{qp}| > .009$. Those M_{pq} with a $*$ in them have the additional property of pure improvement: $M_{pq} \geq 0$ and $M_{qp} \geq 0$.

replacement. The first is that wrongly detected plosives are repaired and the second is that consonants misrecognized as schwa-like vowels are repaired. There are some other corrections without an apparent pattern, as well as some mistakes introduced as a result of segment replacement.

7.2 Phone posterior smoothing

While the previous section incorporated relatively short and variable duration context into the acoustic model, in this section, we show that using long distance context can also be helpful. We show that low-pass filtering the posterior distributions of the phone given the observation frames improves the frame classification accuracy by about 7% absolute on conversational telephone speech. This is true if posterior distributions are obtained either from Gaussian mixture models (GMMs) or from multi-layer perceptrons (MLPs). The results suggest that frame classification benefits from acoustic contexts of a much longer duration (210-300ms) than what is frequently used in computing the posteriors (up to 120ms).

In the process of recognizing speech, automatic speech recognizers need to evaluate the posterior probability $p(q|o)$ of the state q given the observation frame o, for every frame in an utterance. The state q is typically a phone, a triphone, a vector of articulatory features or some other sub-word unit. $p(q|o)$ is often obtained via the Bayes rule from $p(o|q)$, where $p(o|q)$ is modeled as a set of Gaussian mixture models (GMMs), one GMM for each q. In the case of hybrid recognizers, a multi-layer perceptron (MLP) directly represents the $p(q|o)$ (See Section 3.3.6 and [50] for more details). The parameters of GMMs and MLPs are learned from labeled training observation frames.

The labels are either obtained from a human phonetic transcription, or from forced alignment of transcriptions against speech using some other model, or in case of embedded training, a distribution over labels is obtained from the unaligned transcriptions of speech. When we know the true label q^* for an observation frame o, the frame classification accuracy (FCA) of $p(q|o)$ on N frames is computed as

$$FCA = \frac{1}{N} \sum_{i=1}^{N} \left[\arg\max_{q} p(q|o_i) = q_i^* \right]$$

where $[\cdot]$ is the indicator function which equals 1 when the statement in the brackets is true, and 0 otherwise.

Although the acoustic model interacts with the language model in a complicated way, having an acoustic model with a high FCA is important for the low overall WER of the recognizer.

In these experiments we train two kinds of acoustic models: $p_{GMM}(q|o)$ which uses GMMs for $p(o|q)$, and $p_{MLP}(q|o)$ which is an MLP. We experimentally show that smoothing $p(q|o)$ with a low pass filter in the time direction improves the FCA for both $p_{GMM}(q|o)$ and $p_{MLP}(q|o)$. It is possible to further improve the FCA by choosing a different filter length for each q in $p(q|o)$.

The experiments described here are similar to TRAP features and tonotopic MLP system described in Section 2.3.1 in the sense that long duration (1 second) features are combined to predict the phone for a given frame. These long duration features combined in a tandem system with medium duration (100ms) MLP-derived features and short duration (25ms) perceptual linear predictive (PLP) features yield a significant drop in the WER on conversational telephone speech [56, 57]. The main difference between above work and our experiments lies in the kind of information integrated over the long duration. Both TRAPs and tonotopic MLPs extract long duration features from each critical band independently, and combine them with short range features in the later stages of the recognizer. In our experiments, the features are extracted from the entire spectrum of short and medium duration features and the posteriors $p(q|o)$ are combined over the long duration.

The experiments in this section are also similar to the fMPE approach in described in Section 2.3.1, where $p(q|o)$ for the target frames is a linear combination of individual Gaussians taken from all the mixtures of all the states of the target frame and the 18 surrounding frames (200ms). Different weights are learned for every q in $p(q|o)$ in a greedy, discriminative way from the training data. In other words, in fMPE a very wide QxH matrix M is learned where Q is the number of states q, and H=(number of Gaussians per state)x(number of states)x(number of neighboring frames). M is constrained to taking on only certain values. See [58] for details.

In our experiments, we also linearly map posteriors from Gaussians into $p_{GMM}(q, o)$ which is then normalized into $p_{GMM}(q|o)$ and again linearly mapped by a low-pass filter into the smoothed $p_{GMM}(q|o)$. This is equiva-

lent to normalizing first and then applying a linear operation. The difference between fMPE and our experiments is in the value of M.

The next section describes the acoustic models $p_{GMM}(q|o)$ and $p_{MLP}(q|o)$, Section 7.2.2 describes the filtering experiments and 7.2.3 reports the effects of filtering on the FCA.

7.2.1 The acoustic models

The 48 phone phoneset from which q takes its values is enumerated in Appendix A.1 and described in Section 5.4.2.

The MLP posterior phoneme probability distribution is defined as

$$p_{MLP}(q|o_i) = MLP(i)$$

$MLP(i)$ is trained exactly as in Section 3.3.6, and then normalized to sum up to one, so $p_{MLP}(q|o_i)$ is a valid distribution.

In the case of posterior distribution $p_{GMM}(q|o_i)$, the observation o_i for frame i is the PLP+MLP observation. It consists of the 39-dimensional PLP observation for frame i, concatenated with a suitably transformed output of the MLP classifier (see Section 3.3.6). $p_{GMM}(q|o_i)$ uses the baseline acoustic model $p(o|q_-q_jq_+)$ in its definition:

$$\begin{aligned} p_{GMM}(q|o) &= \frac{p_{GMM}(q,o)}{\sum_q p_{GMM}(q,o)} \\ p_{GMM}(q,o) &= \sum_{j=1}^{J_q} \sum_{q_-,q_+} p(o|q_-q_jq_+)p(q_-q_jq_+) \end{aligned} \tag{7.6}$$

where $p(q_-q_jq_+)$ is the sub-triphone unigram probability obtained from occurrence counts in the forced-aligned transcription.

7.2.2 Experimental methods

For each frame o, we compute $p(q|o)$ with either the $p_{MLP}(q|o)$ or the $p_{GMM}(q|o)$ model and low-pass filter the posteriors in the time direction. Throughout the experiments, we are using a Hamming filter of order N. The most direct

experiment is to filter the posteriors with a range values for N. Then we can select the N to maximize the frame classification accuracy (FCA).

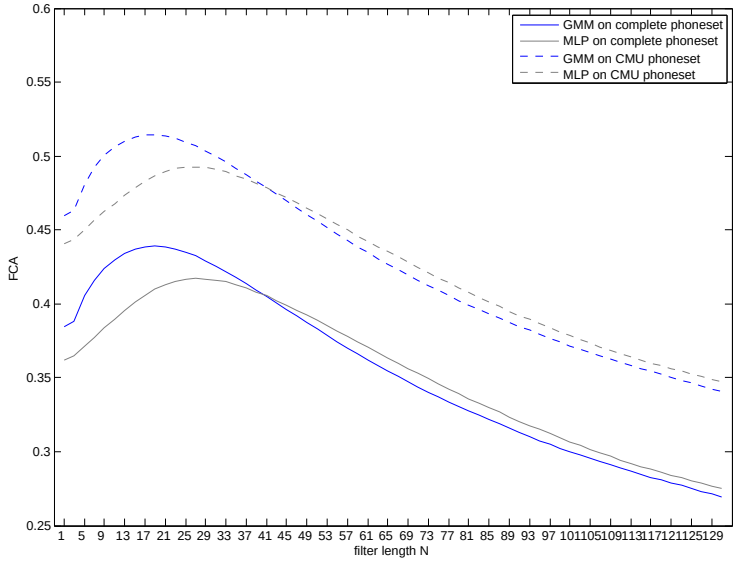

Figure 7.7: The training data frame classification accuracy after filtering the posteriors $p_{GMM}(q|o)$ and $p_{MLP}(q|o)$ with a hamming window of length N. The best FCA is reached at $N^* = 19$ for $p_{GMM}(q|o)$ and $N^* = 27$ for $p_{MLP}(q|o)$. The results are reported on the full 48 phoneset and also on the CMU phoneset, where all silence and non-speech phones are mapped to the `EOW` phone.

The tuning of filters is performed on a 2500 utterance (463286 frame) subset of the Fisher training data, and the results are reported on a 2000 utterance (755893 frame) set disjoint from the training data on which the acoustic models were trained.

Figure 7.7 shows the FCA after filtering the training data posteriors with hamming(N) with odd $N \in [1\ 131]$. $N = 1$ is the unfiltered baseline case. We let N^* be the filter length maximizing the FCA on the training data. On the training data, $p_{GMM}(q|o)$ actually outperforms $p_{MLP}(q|o)$, while the opposite is true on the test data. The optimal filter length N^* is $N^* = 19$ for $p_{GMM}(q|o)$ and $N^* = 27$ for $p_{MLP}(q|o)$.

It is possible that the FCA can be further improved by choosing a specific filter length for each phoneme posterior $p(q|o)$, instead of using the same length for all q. We perform a greedy coordinate search over the integer filter lengths, using N^* as the initial point for each phone. We call the resulting per-phone optimized filter length N^{**}. The search for N^{**} is performed

for each choice of {MLP,GMM} models, and each choice of {full, CMU} phonesets (see next section). In all model and phoneset combinations, N^{**} stays close to the initial N^*: $|N^{**} - N^*| <= 3$.

7.2.3 Results

The experiments were performed using the 48-phone phoneset. However, the non-speech phones such as `/SIGH/` occur rarely in the training data, and are typically not present in the pronunciation dictionary of a large vocabulary recognizer. The `/SIL/` phoneme occurs only at the utterance boundaries, and so its confusion with the `/EOW/` is not important either. Therefore, to have a more realistic measure of the FCA, all the non-speech phones are mapped to `/EOW/`, for both the reference labels, and the output of the classifier before calculating the FCA. The main conclusions presented here remain the same and the absolute improvement in the FCA does not change substantially, regardless of which phoneset used.

Table 7.4 reports the FCA for the baseline case of $N = 1$, as well as the N^* and phone-specific N^{**} filter lengths. The optimal $N^* = 19$ for GMM model and $N^* = 27$ for the MLP model, regardless of the phoneset used.

		full phoneset		CMU phoneset	
row	filter	GMM	MLP	GMM	MLP
1	FCA $N = 1$	42.5%	44.3%	53.1%	53.8%
2	FCA N^*	48.5%	50.1%	58.9%	60.0%
3	FCA N^{**}	49.8%	51.4%	60.0%	60.8%

Table 7.4: Test set FCA of $p_{MLP}(q|o)$ and $p_{GMM}(q|o)$ smoothed with different filters, using both the 48 phone phoneset, and the 39 phone CMU phoneset. FCA is reported for the baseline unfiltered ($N = 1$) posteriors, for the best single filter length N^* and for the per-phone filter lengths N^{**}.

Filtering benefits most phonemes and not just those with a long average duration. Filtering the $p_{MLP}(q|o)$ model with N^{**} improved the recognition accuracy for every phoneme except /EH/ and /UH/, and accuracy of /EH/ and /UH/ did not suffer substantially. The mistakes introduced on the /EH/ and /UH/ as a result of filtering accounted for 0.006% of the frames, while the improvements on all the other phones account for 7.1% of the frames.

It is also interesting to see how the frame evidence $p(o)$ interacts with

filtering. We can calculate $p(o) = \sum_q p_{GMM}(q, o)$ by using $p_{GMM}(q, o)$ in Equation 7.6.

Figure 7.8 shows that high evidence frames are mostly unaffected by filtering while low evidence frames tend to be corrected. High evidence frames tend to closely resemble something in the training data, and which can therefore be classified with reasonable accuracy. Also, their posterior $p(q|o)$ tends to be a delta function - close to 1 for some one q and close to 0 for all others. Because the low-pass filter is linear, its output is dominated by the largest values of $p(q|o)$ within $\pm N^{**}/2$ frames. The bad values of $p_{MLP}(q|o)$ or $p_{GMM}(q|o)$ in low-evidence frames therefore tend to be replaced by better estimates from neighboring high-evidence frames, while the good values in high-evidence frames are relatively unaffected.

The one exception to the above is that filtering does lower the FCA of the very high-evidence (and typically singleton) frames for $p_{GMM}(q|o)$ model. The $p_{MLP}(q|o)$ model is unaffected possibly because the unfiltered baseline already incorporates the neighboring low-evidence frames.

Figure 7.8 also shows that filtered posteriors from the MLP model are more accurate on high evidence frames ($\log(p(o) \geq -65$), while filtered posteriors from $p_{GMM}(q|o)$ are more accurate on the low evidence frames.

As an additional experiment, we also low pass filtered $p_{GMM}(q, o)$ and this approach did not work. For $p_{GMM}(q, o)$, N^* is 3 and best FCA achieved was is 43.3% and 54.0% on the 48-phone and the CMU phonesets respectively. Apparently, it is important to let each neighboring frame 'vote' on the posterior of the target frame, where the vote is weighted by the hamming filter, and not by the highly variable evidence $p(o)$ of the voting frame.

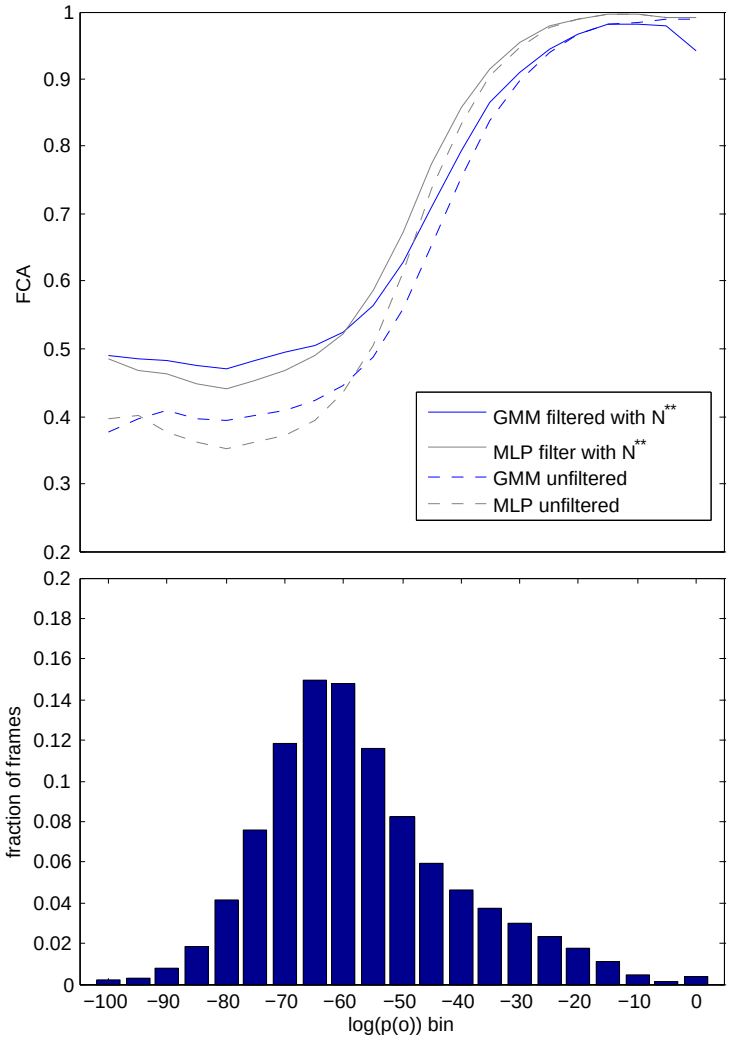

Figure 7.8: The upper plot shows the FCA as function of $\log(p(o))$ for unfiltered $p_{GMM}(q|o)$ and $p_{MLP}(q|o)$ posteriors as well for posteriors filtered with the per-phone optimized filter lengths N^{**}. The frames are binned by $\log(p(o))$, and the FCA is calculated for each bin. The lower plot shows the histogram of the normalized number of frames in each bin. The bin for $\log(p(o)) = 0$ also includes all the frames where $\log(p(o)) > 0$.

Chapter 8

DISCUSSION

This chapter reviews the key findings, summarizing the experiments that worked and experiments that should have worked but didn't. For the later ones, possible explanations are provided. Directions for future work are also discussed.

This thesis explored pronunciation modeling in a broad sense, encompassing implicit pronunciation modeling which takes place within the acoustic models and explicit pronunciation modeling which manipulates the set of pronunciation units and the pronunciation lexicon.

From a birds eye view, pronunciation variability is an important problem because it is one of the major causes of poor LVCSR performance. Therefore, having an accurate explicit pronunciation model is an important goal. This goal is very challenging, but it is being pursued by a number of researchers from a variety of points of view [95, 96].

It is also known that the acoustic model quality and the overall difficulty of the recognition task have an important effect on the pronunciation model's ability to model pronunciation variability [11, 12, 13]. The same pronunciation modeling approach can perform very differently when paired with different acoustic models or when applied to recognition tasks of differing difficulty. Therefore, it is important to consider pronunciation modeling in conjunction with a particular acoustic model and within the context of a particular recognition task.

This thesis explores explicit pronunciation modeling in context - with a particular acoustic model and on the task of conversational telephone speech recognition. Instead of building an accurate and complete pronunciation model (a daunting task), it attempts to characterize the pronunciation models along one dimension, namely the pronunciation perplexity. Because conversational speech recognition is a difficult task and the acoustic model is not very accurate, the explicit pronunciation model in this case is designed

with minimal pronunciation perplexity.

Our explicit pronunciation models were evaluated against a baseline having the same number of model parameters. For a particular parameter budget, the explicit pronunciation modeling experiments allow us to state three things:

1. Growing the model through traditional methods (such as increasing the number of Gaussians per mixture) lowers the WER.

2. Growing the model through more sub-word units, such as the units designed in Chapter 6, also lowers the WER.

3. Compared to a baseline with a good balance between GPMs and the number of triphone clusters, similar performance levels can be reached with fewer model parameters, when error-based sub-word units are added to the model.

The first two statements are not new and the two approaches have been proposed exactly because they lower the WER. With the third statement, it is not surprising that adding sub-word units which depend on longer context will improve the accuracy. But it is perhaps interesting that including models with longer context is beneficial long before traditional approaches have been exhausted.

The experiments with error-driven sub-word units presented here show no improvement in WER over baseline, and only a small reduction in the model size. But the ideas described here could still be beneficial for a state-of-the-art speech recognizer because this error-driven unit selection provides a principled way of adding context-dependent units that unifies a variety of successful ad-hoc methods for incorporating context.

This kind of context-dependent unit selection spans the range from short units with short context, such as triphones, through syllable models and whole word models, all the way to the long context effects captured by multi-word models. All these variable-length units with variable length contexts are derived automatically and coexist within the same decoder, and the longer-length models are inserted only if the existing recognizer breaks down. The 'no perplexity added' rule from Section 2.2 is followed, since the new units only replace existing pronunciations and do not provide alternative pronunciations.

Additionally, this approach can be used to diagnose the pronunciation dictionary, as it will suggest new models for words that either have unrealistic pronunciations, or are inherently variable. For example, Appendix A.2 shows that a whole word model replaces the `MHM` ⇒ `AH M HH AH M` definition from the CMU dictionary, which is probably rarely pronounced in this way. The inclusion of error-driven units makes the dictionary accuracy less important, although it's still important for initialization.

Explicit and implicit pronunciation modeling approaches are complimentary, and in fact are typically used together in the same recognizer. In Chapter 7, this thesis presented two new ways of making longer-duration context available to the acoustic model.

One is a very simple segmental model which uses a single high-confidence frame to represent an entire segment. The acoustic model already considers the dynamics of the observations, either through velocity and acceleration, or through tandem PLP+MLP observations. Therefore summarizing frame segments with a single frame makes longer duration acoustic context available to the acoustic model.

Training the acoustic models on higher quality representative frames improves the WER of a tandem triphone recognizer on conversational telephone speech. Using the tandem PLP+MLP observations is essential: the WER of the segmental model is actually higher than the baseline if only the PLP observations are used. This suggests promise in using tandem features in segmental models.

The WER improvement is a result of better silence detection, as well as better modeling of most phonemes. In particular, the segmental model makes fewer mistakes such as misrecognizing nasals as plosives and misrecognizing consonants as schwa-like vowels.

In segment summarization, even the high-confidence segments span phoneme transitions often. Therefore, the presented segmental model cannot be used to reduce the number of frames in an utterance during decoding without detriment to the WER. However, the frames can safely be dropped during training, thereby reducing the time needed to train a model. The number of dropped frames is large when the utterances include a lot of silence.

Our second implicit pronunciation experiment shows that the frame classification accuracy can be improved by about 7% absolute by low-pass filtering the phoneme posteriors with a filter of a relatively long length (200-300ms).

Smoothing benefited the frame classification accuracy (FCA) of almost all phonemes, and not just those with long average durations.

The phoneme posterior smoothing experiments confirm that a filter length of about 200ms is a good choice when the posteriors are computed with Gaussian mixture model $p_{GMM}(q|o)$. The same filter length is used in fMPE training, which uses something similar to GMM models to compute the phone posteriors [58] and uses a linear combination of the posteriors as the HMM observations. However if fMPE was used with tandem models, according to our experiments, a good filter length to try is about 290ms (27 frames). The positive FCA results suggest that this approach should be helpful in full speech recognition, which is an obvious direction for future work.

8.1 Future work

The experiments described in this thesis point the way to additional WER improvements, through both implicit and explicit pronunciation modeling. The implicit pronunciation modeling appears more successful, but the explicit pronunciation modeling ideas remain tantalizing. The directions for future work are described for both in turn.

With the phone posteriors smoothing approach, $p_{GMM}(q|o)$ posteriors are more accurate on low evidence frames, while the $p_{MLP}(q|o)$ posteriors are more accurate on the high evidence frames. It may be possible to increase the FCA further by combining the two models $p_{GMM}(q|o)$ and $p_{MLP}(q|o)$.

Another idea is to use the smoothed posteriors as an initialization for proven discriminative learning techniques. For example, the Hamming filter could be the initialization point for the feature transformation matrix M in fMPE training.

The low-pass filtering of the $p_{MLP}(q|o)$ posteriors can be seen as a large, sparse MLP with two hidden layers. The input layer and the hidden layers would be the independent 9-frame MLPs. The mapping from from the second hidden layer to the output layer consists of the non-linearity where the activations of the 9-frame MLPs are normalized to sum up to 1, followed by the hamming-weighted sum between the nodes for each q. This would be similar to using a group of neural networks as statistical experts as described in [50]. Given the smoothing filter and the weights of $MLP(i)$, the weights

of this two-hidden-layer MLP are completely specified. These can serve as an initialization, and the network (or parts of it) can then further be trained with back-propagation.

8.1.1 Future work on dictionary generation

Even though the letter-to-phone mapping (described in Chapter 5) played a minor role in the construction of the recognizers described here, it may be useful in other contexts, so the ideas for future work on lexicon generation are described here as well.

There are a number of relatively easy ways the letter-to-phone mapping could further be improved. Using a trigram "language model" over phones with better smoothing instead of the bigram with +1 smoothing will probably be beneficial.

As mentioned earlier, some of the errors in our system occur where a single letter is pronounced with multiple phonemes and there are more phones than observed letters (e.g. ABUSES gets pronounced as `AE B AH S AH Z` instead of `AH B Y UW S AH Z`). Around 1.8% of the words fall into this category. In English, there are relatively few letters (e.g. U and X) causing such problems and splitting those letters into two parts is an easy way to address this.

Using quinphones instead of triphones would not only sharpen the observation distributions of the HMMs, but will also reduce the ratio of adjacent tee'd models, further improving model quality.

Another interesting experiment would be to combine the approaches in [87, 88] with the HMM model described here in a way similar to a Hybrid HMM. A distribution for

$$p(phoneSequence|letter_{i-n}, ..., letter_i, ..., letter_{i+n}) \tag{8.1}$$

can be estimated as in [87, 88]. Most of the time, *phoneSequence* consists of only a single phone and in this case it can be provided as virtual evidence to the triphone (or quinphone) system presented here. In the rare cases where *phoneSequence* with multiple phones has significant probability, an alternate pronunciation with more phones can be hypothesized, and weighed by the probability given by Equation 8.1. This kind of top-down and bottom-up hybrid approach handles interdependencies between neighboring phones and

between neighboring letters, and can potentially give even better accuracy than either approach alone.

8.1.2 Future work on explicit pronunciation modeling

Despite the promise of the explicit pronunciation modeling approach described here, there are a variety of difficulties that still need to be overcome.

A practical problem is that for each unit selection experiment, the full triphone models are trained from scratch. This time-consuming process is necessary because the additional unit selection takes place before any EM training is done. Each point in Figure 6.3 is about a week of compute time on our compute cluster. A possible solution would be to use less training data. An added advantage of this is that the WER improvement due to growing GPMs and the number of DT clusters would bottom out on smaller models, and we could see the improvement from additional units more clearly.

Another way to speed up training is to train the acoustic models for the new units only on the examples in which they occur. There are at most tens of thousands of examples for a given mistake, so this approach should be much faster than retraining all units on the entire dataset. The disadvantage of this is that the original triphones will not be modified to reflect the missing tokens which were moved to the new units. Development testing could also be performed just on confusable token pairs which were discovered during the mistake token search.

Additionally, virtually every design choice made in the selection of new units could be improved and we describe some of these improvements here.

The obvious improvements would be to use methods which have been proven useful in triphone systems: discriminative training and decision-tree based clustering of unit contexts. The former is particularly important in this case, because new units may have a tendency to model the highly reduced portions of speech, and so the new models may actually become more confusable. The decision-tree based clustering offers the standard advantages, such as ensuring that models have sufficient training data, and making good use of the parameter budget, while allowing longer contexts.

Our experiments so far have been only with whole-word and sub-word units, while a substantial fraction (28%) of the mistakes seen in the data span

word boundaries. Experiments with multi-word units should be promising. They should be compared against systems with dictionaries which include phonetic definitions for multi-words, which have been very successful [21].

Another possible improvement lies in the definition of the mistake token. It's possible that only a subset of the long string of phonemes making up the mistake is responsible for the misrecognition. Detecting and separately modeling only the responsible segments instead of the entire mistake can reduce the number of additional model parameters.

In [37], the authors describe a more realistic measure of the anticipated classification error change as a result of introducing a new pronunciation model. This measure can be adapted to replace our crude estimate in Eqn. 6.1.

A fourth improvement lies in the way the set $\mathcal{M}$ of non-conflicting candidate units is selected. Two candidate mistake-in-context units conflict if their mistake portions overlap and their contexts are compatible. As described in Section 6.2, we pick a non-conflicting set greedily by choosing the units with longest mistakes and longest contexts, which do not conflict with already chosen units. Instead, we can form a graph with weighted vertices, with units as vertices weighted by the expected improvement due to selecting the unit, and inserting edges between units if the units conflict. The problem of choosing the set of minimally conflicting candidate units reduces to the well studied maximum weighted independent set problem. It's the same problem used to select tee models in dictionary generation in Section 5.2. As mentioned earlier, the problem is NP-complete, but a number of approximate algorithms have been proposed and analyzed [97], and we can use one of the more sophisticated ones for our problem as well.

Recently, a discriminative approach has been introduced which grows the pronunciation model while taking into account the added perplexity between lexical items [37]. Also recently, methods were described for joint discriminative training of the language and acoustic models [35, 36, 34]. It is encouraging to see more research considering how the pronunciation model interacts with confusability in the acoustic model. Even given the recent progress, a systematic study of the interaction of the acoustic and explicit pronunciation models would further advance the state of the art of ASR.

Appendix A

A.1 The Fisher phoneset

The fisher phoneset is derived from the CMU phoneset http://www.speech.cs.cmu.edu/cgi-bin/cmudict and is given in the following table.

Phone	substates	example	pronunciation
AA	3	odd	AA D
AE	3	at	AE T
AH	3	hut	HH AH T
AO	3	ought	AO T
AW	2	cow	K AW
AY	2	hide	HH AY D
B	3	be	B IY
CH	3	cheese	CH IY Z
D	3	dee	D IY
DH	3	thee	DH IY
EH	3	Ed	EH D
ER	3	hurt	HH ER T
EY	2	ate	EY T
F	3	fee	F IY
G	3	green	G R IY N
HH	3	he	HH IY
IH	3	it	IH T
IY	3	eat	IY T
JH	3	gee	JH IY
K	3	key	K IY

Continued on next page

Table A.1 – continued from previous page

Phone	substates	example	pronunciation
L	3	lee	L IY
M	3	me	M IY
N	3	knee	N IY
NG	3	ping	P IH NG
OW	2	oat	OW T
OY	2	toy	T OY
P	3	pee	P IY
R	3	read	R IY D
S	3	sea	S IY
SH	3	she	SH IY
T	3	tea	T IY
TH	3	theta	TH EY T AH
UH	3	hood	HH UH D
UW	3	two	T UW
V	3	vee	V IY
W	3	we	W IY
Y	3	yield	Y IY L D
Z	3	zee	Z IY
ZH	3	seizure	S IY ZH ER
SIL	3		
NOISE	3		
LAUGH	3		
BREATH	3		
COUGH	3		
LIPSMACK	3		
SIGH	3		
SNEEZE	3		
EOW	1		

Table A.1: The Fisher phoneset and the number of sub-phone states for each phone.

A.2 Mistakes-in-context added to baseline experiment

The new sub-word units added into the model chosen by analyzing the mistakes made by the baseline recognizer in the first 100,000 utterances of the training data. The mistakes-in-context are restricted to be word internal. There are 69 such mistakes, and of these 66 are whole word mistakes. The first three columns are the left context, the modeled sequence, and the right context respectively. The next two columns are the statistics for the number of hypothesized phone starts within the frame sequence spanned by the reference token. The tokens are grouped by whether the hypothesis was correct, or incorrect, and for each group, the mean and the variance is reported. The "errors seen" are the tokens seen that were recognized at least partially incorrectly, and the "error ratio" column is the "errors seen" tokens divided by the total tokens for that mistake-in-context.

mistake-in-context			**phone starts mean(var)**		**errors**	**error**
Left	**Mistake**	**Right**	**correct**	**wrong**	**seen**	**ratio**
EOW	AH	EOW	1.26(0.76)	0.93(0.69)	15350	0.725734
EOW	DH_AH	EOW	2.03(0.68)	1.54(1.03)	14636	0.719037
EOW	T_IH	EOW	1.98(0.68)	1.48(1.05)	11218	0.679838
EOW	AH_V	EOW	2.17(0.72)	1.51(0.99)	9348	0.731055
EOW	IH_T	EOW	2.17(0.86)	1.56(1.15)	8676	0.722097
EOW	IH_N	EOW	2.27(1.05)	1.83(1.41)	6561	0.728919
EOW	AH_M_HH_AH_M	EOW	4.90(0.47)	4.19(2.44)	3278	0.816032
EOW	ER	EOW	0.93(0.81)	1.09(0.81)	5307	0.637324
EOW	IH_Z	EOW	2.24(1.04)	1.73(1.31)	4402	0.653213
EOW	AE_N_D	EOW	2.97(0.74)	1.98(1.49)	11721	0.543343
EOW	W_AH_Z	EOW	3.17(0.89)	2.19(1.74)	3897	0.653092
EOW	IH_F	EOW	2.24(0.86)	1.93(1.45)	2504	0.702779
EOW	AE_T	EOW	2.15(0.76)	1.56(0.96)	1930	0.752730
EOW	K_AY_N_D	EOW	4.11(0.88)	2.74(1.92)	1764	0.776750
EOW	EH_Z	EOW	2.08(0.71)	1.93(1.30)	1635	0.799511
EOW	W_AH_T	EOW	3.15(0.80)	2.34(1.59)	2916	0.630759
EOW	G_OW_IH_N	EOW	4.46(1.06)	2.78(2.22)	1600	0.771828
EOW	IH_T_S	EOW	3.32(1.04)	2.23(1.77)	4339	0.566671
EOW	K_AH_N	EOW	3.33(0.78)	2.78(1.76)	1421	0.742812
EOW	DH_EH_N	EOW	3.38(0.96)	2.73(1.78)	1513	0.696914
EOW	AH_N	EOW	2.52(0.96)	1.56(1.17)	1029	0.842062
EOW	DH_AH_M	EOW	3.36(0.82)	2.57(1.71)	1256	0.737522

Continued on next page

Table A.2 – continued from previous page

mistake-in-context			phone starts mean(var)		errors	error
Left	**Mistake**	**Right**	**correct**	**wrong**	**seen**	**ratio**
EOW	AW_T	EOW	2.14(0.86)	1.70(1.19)	1254	0.690529
EOW	D_UW	EOW	2.27(0.98)	1.68(1.35)	2771	0.571222
EOW	W_EH_N	EOW	3.18(0.83)	2.35(1.68)	1359	0.668799
EOW	W_EH_R	EOW	3.07(0.62)	2.25(1.42)	1114	0.712276
EOW	D_OW_N	EOW	3.21(0.95)	2.04(2.32)	3370	0.554094
EOW	AY_M	EOW	2.29(1.04)	1.78(1.21)	2643	0.568754
EOW	HH_AE_D	EOW	3.14(1.00)	2.50(1.66)	1233	0.671935
EOW	Y_UW_Z	EOW	3.20(1.15)	2.07(1.51)	738	0.840547
EOW	W_AA_N_T	EOW	4.20(1.03)	3.01(1.80)	826	0.782938
EOW	S_AO_R_T	EOW	4.15(0.97)	2.59(1.81)	770	0.786517
EOW	DH_IH_S	EOW	3.33(0.88)	2.75(2.22)	1937	0.583785
EOW	HH_UW	EOW	2.36(1.22)	1.64(1.05)	964	0.701601
EOW	D_IH_D	EOW	3.37(0.76)	2.63(2.23)	865	0.706699
EOW	D_IH_N_T	EOW	4.43(0.86)	3.34(2.92)	746	0.748996
EOW	HH_AE_Z	EOW	3.18(1.03)	2.69(1.71)	681	0.731472
EOW	HH_IY	EOW	1.98(0.84)	1.58(1.20)	1179	0.609302
EOW	W_AH_L	EOW	3.02(0.95)	2.29(2.11)	495	0.860870
EOW	AH_P	EOW	2.26(0.92)	2.13(1.23)	1116	0.612178
EOW	W_ER	EOW	2.18(0.97)	1.65(1.25)	1468	0.577498
EOW	HH_AH	EOW	1.66(0.77)	1.28(1.12)	556	0.760602
EOW	JH_IH_S_T	EOW	4.15(0.95)	2.97(2.67)	3046	0.533263
EOW	AA_N	EOW	1.92(0.85)	1.73(1.08)	2052	0.550282
EOW	IH_M	EOW	2.56(1.26)	1.97(1.42)	428	0.886128
EOW	OW_N_L_IY	EOW	4.12(0.71)	2.96(2.12)	491	0.802288
EOW	DH_AH_N	EOW	3.47(0.97)	2.88(2.04)	592	0.727273
EOW	B_IH_N	EOW	3.28(0.99)	2.62(1.53)	853	0.635618
EOW	K_AA_Z	EOW	3.38(0.73)	2.75(1.53)	366	0.958115
EOW	DH_OW	EOW	2.78(1.03)	1.93(1.27)	404	0.865096
EOW	AY_V	EOW	2.14(0.85)	1.90(1.14)	852	0.618287
EOW	HH_AW	EOW	2.25(1.11)	1.73(1.47)	1261	0.571882
EOW	W_IH_TH	EOW	3.33(1.07)	2.52(1.91)	1754	0.547954
EOW	EH	EOW	1.50(1.00)	0.99(0.77)	301	0.974110
EOW	HH_AH_M	EOW	4.00(0.00)	2.44(1.48)	284	0.986111
EOW	Y_IH_R	EOW	3.03(0.94)	2.41(1.24)	486	0.689362
EOW	AY_D	EOW	2.41(1.22)	2.20(1.13)	344	0.807512
EOW	W_AH_N	EOW	3.19(1.00)	2.56(1.78)	1330	0.552096
EOW	AE_M	EOW	2.78(2.42)	1.97(1.32)	292	0.856305
EOW	N_UW	EOW	2.25(1.54)	1.77(1.61)	443	0.683642

Continued on next page

Table A.2 – continued from previous page

mistake-in-context			phone starts mean(var)		errors	error
Left	**Mistake**	**Right**	**correct**	**wrong**	**seen**	**ratio**
EOW	AY_L	EOW	2.09(0.43)	1.70(1.27)	260	0.918728
EOW	HH_ER	EOW	2.20(1.05)	1.52(1.28)	383	0.701465
EOW_Y	EY	EOW	0.00(0.00)	0.47(0.45)	221	0.995495
EOW	B_AY	EOW	2.30(1.22)	2.39(1.22)	502	0.638677
EOW	AA	EOW	0.92(0.77)	1.25(1.02)	1336	0.538059
EOW	IY_T	EOW	2.16(0.82)	1.75(1.48)	225	0.855513
EOW	EH	N_D_EOW	0.83(0.63)	0.76(0.71)	221	0.772727
EOW	OW_N	EOW	2.52(0.86)	1.97(1.31)	261	0.681462
EOW_K	OW	L_D_EOW	1.23(0.95)	0.40(0.29)	189	0.707865

Table A.2: Word internal mistakes-in-context.

References

[1] "Nuance Communications website," http://www.nuance.com/.

[2] S. Greenberg, J. Hollenback, and D. Ellis, "Insights into spoken language gleaned from phonetic transcription of the Switchboard corpus," in *ICSLP*, 1996.

[3] M. E. Beckman, "Timing models for prosody and cross-word coarticulation in connected speech," in *HLT '89: Proceedings of the workshop on Speech and Natural Language*. Morristown, NJ, USA: Association for Computational Linguistics, 1989, pp. 12–21.

[4] K. Chen, M. Hasegawa-Johnson, A. Cohen, S. Borys, S.-S. Kim, J. Cole, and J.-Y. Choi, "Prosody dependent speech recognition on radio news," *IEEE Transactions On Audio, Speech, And Language Processing*, 2006.

[5] J. Zheng, H. Franco, and A. Stolcke, "Rate-of-speech modeling for large vocabulary conversational speech recognition," in *Speech Transcription Workshop*, 2000.

[6] J. E. Fosler-Lussier, "Dynamic pronunciation models for automatic speech recognition," Ph.D. dissertation, University of California, Berkeley, 1999.

[7] Y. Zheng, R. Sproat, L. Gu, I. Shafran, H. Zhou, Y. Su, D. Jurafsky, R. Starr, and S.-Y. Yoon, "Accent detection and speech recognition for Shanghai-accented Mandarin," in *Interspeech*, 2005.

[8] J.-C. Junqua, "The Lombard effect: A reflex to better communicate with others in noise," in *ICASSP*, 1999.

[9] T. Athanaselis, S. Bakamidis, I. Dologlou, R. Cowie, E. Douglas-Cowie, and C. Cox, "ASR for emotional speech: clarifying the issues and enhancing performance," *Neural Networks*, vol. 18, no. 4, pp. 437–444, 2005.

[10] T. Polzin and A. Waibel, "Pronunciation variations in emotional speech," in *Proc. of the ESCA Workshop*, 1998.

[11] M. Weintraub, K. Taussig, K. Hunicke-smith, and A. Snodgrass, "Effect of speaking style on LVCSR performance," in *ICSLP*, 1996, pp. 16–19.

[12] D. McAllaster, L. Gillick, F. Scattone, and M. Newman, "Fabricating conversational speech data with acoustic models: A program to examine model-data mismatch," in *ICSLP*, vol. 5, 1998.

[13] M. Saralar, H. Nock, and S. Khudanpur, "Pronunciation modeling by sharing gaussian densities across phonetic models," *Computer Speech and Language*, vol. 14, pp. 137–160, 2000.

[14] T. Hain, "Implicit modelling of pronunciation variation in automatic speech recognition," *Speech Communication*, vol. 42, no. 2, 2005.

[15] "Winners of nist speech recognition competitions." NIST, Tech. Rep., 2009, http://www.itl.nist.gov/iad/mig/publications/ASRhistory/index.html.

[16] D. Pallett, "A look at NIST's benchmark ASR tests: past, present, and future," in *ASRU*, 2003.

[17] H. Strik, "Pronunciation adaptation at the lexical level," in *ITRW on Adaptation Methods for Speech Recognition*, 2001.

[18] M. Riley and A. Ljolje, "Automatic generation of detailed pronunciation lexicons," in *Automatic Speech and Speaker Recognition: Advanced Topics*, C.-H. Lee, F. K. Soong, and K. K. Paliwal, Eds. Norwell, MA, USA: Kluwer Academic Publishers, 1995.

[19] E. Fosler, M. Weintraub, S. Wegmann, Y.-H. Kao, S. Khudanpur, C. Galles, and M. Saraclar, "Automatic learning of word pronunciation from data," in *ICSLP*, 1996.

[20] M. Riley, W. Byrne, M. Finke, S. Khudanpur, A. Ljolje, J. McDonough, H. Nock, M. Saraclar, C. Wooters, and G. Zavaliagkos, "Stochastic pronunciation modelling from hand-labelled phonetic corpora," *Speech Commun.*, vol. 29, no. 2-4, pp. 209–224, 1999.

[21] A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. R. R. Gadde, M. Plauch, C. Richey, E. Shriberg, K. Sönmez, F. Weng, and J. Zheng, "The SRI march 2000 Hub-5 conversational speech transcription system," in *In Proceedings of the NIST Speech Transcription Workshop*, 2000.

[22] S. Greenberg, "The origins of speech intelligibility in the real world," in *ESCA Workshop on Robust Speech Recognition for Unknown Channels*, 1997.

[23] H. Pfitzinger, "Local speech rate as a combination of syllable and phone rate," in *ASRU*, 1998.

[24] H. Nam and E. Saltzman, "A competitive, coupled oscillator of syllable structure," in *Proceedings of the XIIth International Congress of Phonetic Sciences*, 2003.

[25] S. Greenberg, "Speaking in shorthand – a syllable-centric perspective for understanding pronunciation variation," in *ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition*, 1998.

[26] O. Fujimura, "Syllable as a unit of speech recognition," *IEEE Trans. on Accoustics Speech and Signal Processing*, 1975.

[27] A. Ganapathiraju, J. Hamaker, J. Picone, M. Ordowski, and G. R. Doddington, "Syllable-based large vocabulary continuous speech recognition," *IEEE Transactions On Speech and Audio Processing*, 2001.

[28] A. Ganapathiraju, V. Goel, J. Picone, A. Corrada, G. Doddington, K. Kirchhoff, M. Ordowski, and B. Wheatley, "Syllable – a promising recognition unit for LVCSR," in *ASRU*, 1997.

[29] A. Sethy, B. Ramabhadran, and S. Narayanan, "Improvements in English ASR for the MALACH project using syllable-centric models," in *ASRU*, 2003.

[30] A. Sethy and S. Narayanan, "Split-lexicon based hierarchical recognition of speech using syllable and word level acoustic units," in *ICASSP*, 2003.

[31] A. Hämäläinen, L. ten Bosch, and L. Boves, "Modelling pronunciation variation using multi-path HMMs for syllables," in *ICASSP*, 2007.

[32] A. Hämäläinen, L. Boves, J. de Veth, and L. ten Bosch, "On the utility of syllable-based acoustic models for pronunciation variation modelling," *EURASIP Journal on Audio, Speech, and Music Processing*, 2007.

[33] S. Borys, M. Hasegawa-Johnson, K. Chen, and A. Cohen, "Modeling and recognition of phonetic and prosodic factors for improvements to acoustic speech recognition models," in *Interspeech*, 2004.

[34] H. Printz and P. Olsen, "Theory and practice of acoustic confusability," *Computer Speech and Language*, pp. 77–84, 2002.

[35] J.-T. Chien and C.-H. Chueh, "Joint acoustic and language modeling for speech recognition," *Speech Communication*, vol. 52, no. 3, pp. 223 – 235, 2010.

[36] M. Lehr and I. Shafran, "Discriminatively estimated joint acoustic, duration, and language model for speech recognition," in *ICASSP*, 2010.

[37] O. Vinyals, L. Deng, D. Yu, and A. Acero, "Discriminative pronunciation learning using phonetic decoder and minimum-classification-error criterion," in *ICASSP*, 2009.

[38] K. Livescu and J. Glass, "Feature-based pronunciation modeling with trainable asynchrony probabilities," in *ICSLP*, 2004.

[39] E. Ristad and P. Yianilos, "A surficial pronunciation model," in *Proc. of the ESCA Workshop 'Modeling Pronunciation Variation for Automatic Speech Recognition'*, 1998.

[40] K. Livescu, O. Çetin, M. Hasegawa-Johnson, S. King, C. Bartels, N. Borges, A. Kantor, P. Lal, L. Yung, A. Bezman, S. Dawson-Haggerty, B. Woods, J. Frankel, M. Magimai-Doss, and K. Saenko, "Articulatory Feature-based Methods for Acoustic and Audio-Visual Speech Recognition: 2006 JHU Summer Workshop Final Report," John Hopkins Univerity, Tech. Rep., 2007.

[41] T. Hain, P. Woodland, G. Evermann, M. Gales, X. Liu, G. Moore, D. Povey, and L. Wang, "Automatic transcription of conversational telephone speech," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 6, pp. 1173–1185, Nov. 2005.

[42] T. Holter and T. Svendsen, "Maximum likelihood modelling of pronunciation variation," in *Proc. of the ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition*, 1998.

[43] K. Livescu, O. Çetin, M. Hasegawa-Johnson, S. King, C. Bartels, N. Borges, A. Kantor, P. Lal, L. Yung, A. Bezman, S. Dawson-Haggerty, B. Woods, J. Frankel, M. Magimai-Doss, and K. Saenko, "Articulatory feature-based methods for acoustic and audio-visual speech recognition: Summary from the 2006 JHU summer workshop," in *ICASSP*, 2007.

[44] S. King, C. Bartels, and J. Bilmes, "Svitchboard 1: Small vocabulary tasks from Switchboard 1," in *Interspeech*, 2005.

[45] N. Oostdijk, W. Goedetier, F. V. Eynde, L. Boves, J. Martens, M. Moortgat, and H. Baayen, "Experiences from the spoken Dutch corpus project," in *Proceedings of LREC*, 2002.

[46] J. Bilmes, "Graphical models and automatic speech recognition," University of Washington, Department of Electrical Engineering, Tech. Rep. UWEETR-2001-0005, 2001.

[47] D. Jurafsky, W. Ward, Z. Jianping, K. Herold, Y. Xiuyang, and Z. Sen, "What kind of pronunciation variation is hard for triphones to model?" in *ICASSP*, 2001.

[48] A. Ljolje, "Pronunciation dependent language models," in *Interspeech*, 2006.

[49] K. Chen, M. Hasegawa-Johnson, and J. Cole, "A factored language model for prosody-dependent speech recognition," *Advanced Robotic Systems*, 2007.

[50] N. Morgan and H. Bourlard, "Continuous speech recognition: An introduction to the hybrid HMM/connectionist approach," *IEEE Signal Processing Magazine*, vol. 12, no. 3, pp. 25–42, 1995.

[51] S. Sharma, D. Ellis, S. Kajarekar, P. Jain, and H. Hermansky, "Feature extraction using non-linear transformation for robust speech recognition on the Aurora database," in *ICASSP*, 2000.

[52] Q. Zhu, B. Chen, N. Morgan, and A. Stolcke, "On using MLP features in LVCSR," in *Proc. ICSLP, Jeju, Korea*, 2004, pp. 921–924.

[53] O. Çetin, A. Kantor, S. King, C. Bartels, M. Magimai-Doss, J. Frankel, S. King, and K. Livescu, "An articulatory feature-based tandem approach and factored tandem observation modeling," in *ICASSP*, 2007.

[54] H. Hermansky and S. Sharma, "Traps – classifiers of temporal patterns," in *IN Proc. ICSLP*, 1998, pp. 1003–1006.

[55] H. Hermansky, S. Sharma, and P. Jain, "Data-derived nonlinear mapping for feature extraction in HMM," in *ASRU*, 1999.

[56] N. Morgan, B. Y. Chen, Q. Zhu, and A. Stolcke, "TRAPping conversational speech: Extending TRAP/tandem approaches to conversational telephone speech recognition," in *in Proc. ICASSP*, 2004, pp. 536–539.

[57] B. Y. Chen, Q. Zhu, and N. Morgan, "Tonotopic multi-layered perceptron: A neural network for learning long-term temporal features for speech recognition," in *ICASSP Proc.*, 2005.

[58] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, and G. Zweig, "FMPE: Discriminatively trained features for speech recognition," in *ICASSP Proc.*, 2005.

[59] M. Ostendorf, V. Digalakis, and O. Kimball, "From HMMs to segment models: a unified view of stochastic modeling for speech recognition," *IEEE Transactions On Audio, Speech, And Language Processing*, 1996.

[60] F. Jelinek, "Continuous speech recognition by statistical methods," *Proc. of IEEE*, vol. 64, 1976.

[61] M. Jordan and Y. Weiss, "Graphical models: probabilistic inference," in *Handbook of Neural Networks and Brain Theory*. MIT Press, 2002.

[62] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[63] J. Bilmes, "A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," ICSI, Tech. Rep., 1998.

[64] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. of IEEE*, vol. 77, 1989.

[65] L. E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology," *Bull. Amer. Math. Soc.*, vol. 73, pp. 360–363, 1967.

[66] J. Bilmes and G. Zweig, "The graphical models toolkit: An open source software system for speech and time-series processing," in *ICASSP*, 2002.

[67] G. G. Zweig, "Speech recognition with dynamic bayesian networks," Ph.D. dissertation, 1998.

[68] C. Cieri, D. Miller, and K. Walker, "The Fisher corpus: a resource for the next generations of speech-to-text," in *International Conference On Language Resources And Evaluation*, 2004.

[69] T. Hain, P. Woodland, P. C. Woodl, T. Niesler, and E. Whittacker, "The 1998 HTK system for transcription of conversational telephone speech," in *ICASSP*, 1999, pp. 57–60.

[70] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *Acoustical Society of America Journal*, vol. 87, pp. 1738–1752, Apr. 1990.

[71] C.-P. C. Karim, C. ping Chen, K. Filali, and J. A. Bilmes, "Frontend post-processing and backend model enhancement on the Aurora 2.0/3.0 databases," in *in International Conference on Speech and Language Processing*, 2002, pp. 241–244.

[72] C.-P. Chen, J. Bilmes, and D. Ellis, "Speech feature smoothing for robust ASR," *ICASSP*, vol. 1, pp. 525–528, March 18-23, 2005.

[73] J. Frankel, M. Magimai-Doss, S. King, K. Livescu, and Özgür Çetin, "Articulatory feature classifiers trained on 2000 hours of telephone speech," in *Interspeech*, 2007.

[74] J. J. Odell, "The use of context in large vocabulary speech recognition," Ph.D. dissertation, University of Cambridge, Berkeley, 1995.

[75] "The complete RT-02 evaluation plan, v3, 4/19/2002," NIST, Tech. Rep., April 2002, http://www.itl.nist.gov/iad/mig//tests/rt/2002/docs/rt02_eval_plan_v3.pdf.

[76] "Transcription normalization rules for the 2004 NIST STT competition," NIST, Tech. Rep., 2004, http://itl.nist.gov/iad/mig/tests/rt/2004-spring/devset/transcriptions/en20040402.glm.

[77] L. Gillick and S. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *ICASSP*, May 1989, pp. 532–535 vol.1.

[78] D. Klakow and J. Peters, "Testing the correlation of word error rate and perplexity," *Speech Communication*, vol. 38, no. 1-2, pp. 19 – 28, 2002.

[79] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proc. of ACL*, 1996, pp. 310–318.

[80] J. T. Goodman, "A bit of progress in language modeling extended version," Microsoft, Tech. Rep., 2001.

[81] D. Yuret and A. Stolcke, "SRILM manual," http://www.speech.sri.com/projects/srilm/manpages/ngram-discount.7.html.

[82] A. Stolcke, "Entropy-based pruning of backoff language models," in *In Proc. DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 270–274.

[83] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *In Proc. ICSLP*, 2002, pp. 901–904.

[84] A. Kantor, "Language model experiments on the fisher corpus," http://mickey.ifp.uiuc.edu/wiki/Fisher_Language_Model/.

[85] V. Siivola, T. Hirsimaki, and S. Virpioja, "On growing and pruning Kneser-Ney smoothed n-gram models," *IEEE Transactions on, Audio, Speech, and Language Processing*, vol. 15, no. 5, pp. 1617–1624, July 2007.

[86] C. Chelba, T. Brants, W. Neveitt, and P. Xu, "Study on interaction between entropy pruning and Kneser-Ney smoothing," in *Interspeech*, 2010.

[87] W. M. Fisher, "A statistical text-to-phone function using ngrams and rules," in *in ICASSP*, 1999, pp. 649–652.

[88] K. Toutanova and R. C. Moore, "Pronunciation modeling for improved spelling correction," in *Proceedings of the Association for Computational Linguistics*, 2002, pp. 144–151.

[89] J. Odell, D. Ollason, P. Woodland, S. Young, and J. Jansen, *The HTK Book for HTK V2.0*. Cambridge University Press, Cambridge, UK, 1995.

[90] A. Kantor, "Letter-to-Phone String Mapping Tool," http://mickey.ifp.uiuc.edu/speech/webpronounce/.

[91] "Carnegie-Mellon Pronouncing Dictionary," http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[92] J. R. Glass, "A probabilistic framework for segment-based speech recognition," *Computer Speech & Language*, vol. 17, no. 2-3, pp. 137–152, 2003.

[93] S. Chang, S. Greenberg, and M. Wester, "An elitist approach to articulatory-acoustic feature classication," in *Eurospeech*, 2001.

[94] Z.-H. Tan and B. Lindberg, "Low-complexity variable frame rate analysis for speech recognition and voice activity detection," *IEEE Journal of Selected Topics in Signal Processing*, 2010.

[95] X. Zhuang, H. Nam, M. Hasegawa-johnson, E. Saltzman, and L. Goldstein, "Articulatory phonological code for word classification," in *Interspeech*, 2009.

[96] K. Livescu, "Feature-based pronunciation modeling for automatic speech recognition," Ph.D. dissertation, 2005.

[97] A. Kako, T. Ono, T. Hirata, and M. M. Halldórsson, "Approximation algorithms for the weighted independent set problem," in *Graph-Theoretic Concepts In Computer Science, 31st International Workshop*, 2005, pp. 341–350.