Recognizing Sloppy Speech

Hua Yu

CMU-LTI-05-190

Language Technology Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Alex Waibel
Tanja Schultz
Richard Stern
Mari Ostendorf (University of Washington)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Abstract

As speech recognition moves from labs into the real world, the sloppy speech problem emerges as a major challenge. Sloppy speech, or conversational speech, refers to the speaking style people typically use in daily conversations. The recognition error rate for sloppy speech has been found to double that of read speech in many circumstances. Previous work on sloppy speech has focused on modeling pronunciation changes, primarily by adding pronunciation variants to the dictionary. The improvement, unfortunately, has been unsatisfactory.

To improve recognition performance on sloppy speech, we revisit pronunciation modeling issues and focus on implicit pronunciation modeling, where we keep the dictionary simple and model reductions through phonetic decision trees and other acoustic modeling mechanisms. Another front of this thesis is to alleviate known limitations of the current HMM framework, such as the frame independence assumption, which can be aggravated by sloppy speech. Three novel approaches have been explored:

- *flexible parameter tying*: We show that parameter tying is an integral part of pronunciation modeling, and introduce flexible tying to better model reductions in sloppy speech. We find that enhanced tree clustering, together with single pronunciation dictionary, improves performance significantly.

- *Gaussian transition modeling*: By modeling transitions between Gaussians in adjacent states, this alleviates the frame independence assumption and can be regarded as a pronunciation network at the Gaussian level.

- *thumbnail features*: We try to achieve segmental modeling within the HMM framework by using these segment-level features. While they improve performance significantly in initial passes, the gain becomes marginal when combined with more sophisticated acoustic modeling techniques.

We have also worked on system development on three large vocabulary tasks: Broadcast News, Switchboard and meeting transcription. By empirically improving all aspects of speech recognition, from front-ends to acoustic modeling and decoding strategies, we have achieved a 50% relative improvement on the Broadcast News task, a 38% relative improvement on the Switchboard task, and a 40% relative improvement on the meeting transcription task.

Acknowledgements

First and foremost, thanks to my advisor, Prof. Alex Waibel, for offering me an excellent environment — Interactive Systems *Laboratories* — to pursue my degree. The intriguing plural form is actually not a bluff. During these years, I have visited many wonderful towns in Germany, worked with and befriended many German students. I was fortunate enough to have a very supportive thesis committee: Dr. Tanja Schultz, for always being available and reading my entire thesis several times; Prof. Richard Stern, for feeding and inspiring many graduate students in a pizza lunch series every Thursday; Prof. Mari Ostendorf, for her expertise and insights.

I learned a great deal about speech recognition and good engineering from reading the source code of Janus, one of the best speech recognition systems in the world. Many thanks to Michael Finke, Monika Woszczyna, Klaus Ries, Gang Wang, Puming Zhan and many former ISL members for getting me started.

I have had great fun going through crunching times with Hagen Soltau, Florian Metze, Christian Fügen and Qin Jin on the Switchboard evaluation, with Thomas Schaaf, Wilson Tam, Sebastian Stüker and Mohamed Noamany on the Mandarin Broadcast News evaluation.

Thanks to Detlef, Jürgen, Kornel and Stan for proofreading my thesis, to Vicky and Celine for making a happier workplace.

Life is always sunny ;-) to have Joy around, either in the office or on the slopes. I have always enjoyed the friendship of Jie, Klaus Z., Huadong, Fei, Jerry, Ke, Rong, Yan, Stan, Kornel and many others.

For Lin, my parents and little Robbie: this thesis is dedicated to you.

The peril of having an acknowledgement is, of course, to have someone left out unintentionally. For all of you still reading and pondering, you have my deepest gratitudes.

Contents

Abbreviations

AM	Acoustic Model
ASR	Automatic Speech Recognition
BN	Broadcast News
CMN	Cepstral Mean Normalization
CMS	Cepstral Mean Subtraction
CVN	Cepstral Variance Normalization
DCT	Discrete Cosine Transform
FSA	Feature Space Adaptation
GMM	Gaussian Mixture Model
GTM	Gaussian Transition Model
HMM	Hidden Markov Model
ISL	Interactive Systems Laboratories
LDA	Linear Discriminant Analysis
LM	Language Model
LVCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel-Frequency Cepstral Coefficient
MLLR	Maximum Likelihood Linear Regression
MLLT	Maximum Likelihood Linear Transform
RT	Rich Transcription
SAT	Speaker Adaptive Training
SLPM	State-Level Pronunciation Model
SPD	Single Pronunciation Dictionary
SWB	Switchboard
VTLN	Vocal Tract Length Normalization
WER	Word Error Rate

Chapter 1

The Sloppy Speech Problem

Sloppy speech refers to the casual style of speech that people use in daily conversations, as opposed to read speech, or carefully articulated speech. It is a natural form of communication that people would hardly anticipate to become a major problem for automatic speech recognition.

In the early 1990s, Hidden Markov Models (HMMs) are working quite well on "read speech" tasks, such as the Wall Street Journal (WSJ) task, where the data is recorded from people reading newspapers in quiet rooms. As speech recognition moves from controlled lab conditions into real world scenarios, the speech style problem becomes more and more pronounced.

In the NIST Hub4 (Broadcast News) evaluation, the Broadcast News data contains a variety of different conditions, among them a significant portion of spontaneous speech. While state-of-the-art systems can achieve word error rates (WERs) of below 10% on clean, planned speech, they perform much worse on the spontaneous speech part of the same corpus. WER typically doubles.

Another major benchmark test, the Hub5 (Switchboard) evaluation, specifically targets conversational speech over telephone lines. WERs on this data are quite high, around 40% in the beginning years. While people initially suspected the telephone bandwidth and the relatively high signal-to-noise ratio, an experiment by Weintraub et al. clearly showed that speaking style is the key issue [Weintraub et al., 1996]. In the corpus later known as the SRI Multi-Register corpus, both read and spontaneous speech are collected under similar conditions. The spontaneous part was collected in the same way as in Switchboard. Speakers were asked to come back later to read the transcript of their conversation. This creates a perfect parallel corpus for comparing speaking styles. Recognition experiments show that with everything else (microphone, speaker, sentence) being equal, the WER on spontaneous speech more than doubles that on read speech in the SRI Multi-Register corpus.

While the Switchboard task has been the main focus for the LVCSR community over the past 8 years, automatic meeting transcription has gradually gained importance. It is a much harder task. Our initial experiments on internal group meeting data have a WER of over 40% [Yu et al., 1998]. We carried out a controlled exper-

iment similar to the SRI Multi-Register experiment, where we asked three meeting participants to come back to read transcripts of an earlier meeting. Recognition results show a similar trend: everything else being equal, real meeting data is much harder than the read version. WER increases from 36.7% for read speech to 54.8% for the actual meeting data.

All evidence suggests that sloppy speech is prevalent in the real world, be it broadcast news, telephone conversations, or meetings, and the conventional HMM framework is inadequate to handle it.

In this chapter, we try to define a framework for studying sloppy speech and present some linguistic analysis. Since the ultimate goal is better recognition performance, it is helpful to keep the following questions in mind: why and when people speak sloppily, and why sloppy speech does not seem to bother us as human listeners. A better understanding of these issues will guide us towards a better modeling of sloppy speech.

1.1 Defining Sloppy Speech

One of the main sources of variability in speech is speaking style. There are a plethora of descriptive terms for speaking styles: *careful / clear / formal / casual/ informal / conversational / spontaneous / scripted / unscripted / reading*, etc. Sloppy speech is loosely associated with conversational speech or spontaneous speech. But to give a more accurate definition, we will need a framework to put all speaking styles in perspective.

Defining speaking styles is not an easy task. Here we follow mostly the work of [Eskenazi, 1993], where styles are defined from the point of view of human communication:

> Style reflects the action of the environment upon the individual and the individual upon the environment. It is his perception of the various "status" levels of his listener and of the type of situation in which he finds himself. It is also a projection of himself, his background, and is a setting of the type and tone of conversation he wishes to have. All of this is a mixture of conscious and unconscious (voluntary/involuntary) effort on his part and is not always perceived in the same way it was intended.

In other words, style is a conscious or unconscious choice of the speaker, when reacting to the current situation. For example, one would use a formal tone when speaking in public, but use a casual style when chatting with friends.

Styles often change within the same conversation, so does the degree of attention to the clarity of the discourse. A change in speaking style may be caused by a change in either

- the self-image that the speaker wishes to project;

- the type of information to be communicated;
- the situation in which the conversation takes place (including background noise, arrival or departure of other persons, the dialogue context);
- the image the speaker has of the listener (can't hear well, personal background, etc.).

For example, a speaker would speak differently in a noisy environment than he/she would in a quiet room.

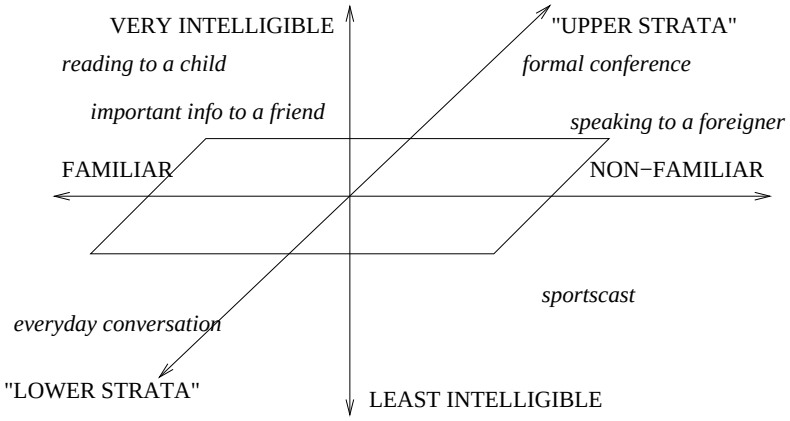


Figure 1.1: Speech Style Space

This definition leads to a characterization of styles along three dimensions, as shown in Figure 1.1.

- *Intelligibility* measures the degree of clarity the speaker intends his message to have. It varies from minimum effort to be clear, to much effort when the channel is noisy, or the listener has a problem understanding.
- *Familiarity* between the speaker and the listener plays a major role in determining styles. Extremes may go from identical twins to talking to someone from another culture and another language.
- *Social Strata* takes into account the context in which the conversation is taking place, as well as the backgrounds of the participants. This goes from a totally colloquial or "lower class" tone to a "highly cultivated" or "upper class" tone.

Figure 1.1 attempts to organize speaking styles in this three-dimensional space. For example, when reading to a child, one tries to be a good cultural vehicle and easily understood before a very familiar listener. Another example is everyday conversation, which happens between acquaintances, in a casual (less intelligible) and informal manner. Sloppy speech typically refers to this kind of speech style. As discussed

before, styles often change within the same conversation. For example, when it comes to important information, the speaking style can become more intelligible. This implies that in a corpus such as Switchboard, the degree of sloppiness can vary widely. Some part of the data can be, or come very close to clear speech.

It should be noted that the definition of speech style is not the only one available. There are other definitions. Here we are mostly interested in finding a good framework to discuss the sloppy speech problem.

1.2 Characteristics of Sloppy Speech

Sloppy speech differs from clear speech in many aspects, such as articulation, phonology, and prosody.

1.2.1 Articulation Factors

Several studies have compared the degree of attention to the articulation between sloppy speech and clear/read speech. A good review can be found in [Eskenazi, 1993]. Attention to articulation is defined to be the degree of attainment of articulatory targets, such as a given formant frequency or stop releases. In general, articulatory targets are reached much more often in clear/read speech than in sloppy speech, for both consonants and vowels. Especially for vowels, there is much evidence suggesting increased articulatory efforts in clear speech, or equivalently, decreased articulatory efforts in sloppy speech:

- Formant values tend to achieve the extremes of the "vowel triangle" in clear speech, compared to more "central" values in sloppy speech. Variability of formant values is also found to be smaller in clear speech, indicated by a smaller cluster in a plot of F1/F2 values.

- Transition rates measure the movement of the formants at the onset and the offset of a vowel. They reflect the coarticulation of the vowel with its neighbors and indicate whether articulatory targets are achieved for the vowel or not. Some authors relate this to the casualness of speech. Some studies find greater transition rates in clear speech, and more CV (consonant-vowel) coarticulation in spontaneous speech.

1.2.2 Phonology

Sloppy speech exhibits increased phonological variability. In the Switchboard Transcription Project [Greenberg, 1996], linguists manually transcribed a portion of the Switchboard corpus at the phonetic level [Greenberg et al., 1996]. It is clear that many words are not pronounced in the canonical way. Phonemes could be either deleted, or have their phonetic properties drastically changed, to such a degree that

only the barest hint of a phoneme segment can be found. Table 1.1 shows the number of variants for some of the most common words encountered in the Switchboard Transcription Project. According to Table 1.1, the word `and` has 87 different pronunciations. Some of the most frequent variants are listed in Table 1.2.

word	# occurrences	# PVs	most common pronunciation
I	649	53	ay
and	521	87	ae n
the	465	76	dh ax
you	406	68	y ix
that	328	117	dh ae
a	319	28	ax
to	288	66	tcl t uw
know	249	34	n ow
of	242	44	ax v
it	240	49	ih
yeah	203	48	y ae

Table 1.1: Pronunciation variability for the most common words in the phonetically segmented portion of the Switchboard Transcription Project (from [Greenberg, 1998]). "#PVs" is the number of pronunciation variants (distinct phonetic expressions) for each word.

Greenberg also questioned the appropriateness of the phonetic representation in this project. Portions of the data are found to be quite hard to transcribe phonetically. It was reported that 20% of the time even experienced transcribers cannot agree upon the exact surface form being spoken. The transcribing process was unexpectedly time consuming (taking on average nearly 400 times real time to complete [Greenberg, 1996]). For this reason, it was decided to transcribe only at the syllable level later on. Greenberg argues that syllables are a more stable, and therefore, a better unit for representing conversational speech. Syllables are much less likely to be deleted. Within the three constituents of a syllable, onsets are generally well preserved, nucleus exhibits more deviation from the canonical, while codas are often deleted or assimilated into the onset of the following syllable [Greenberg, 1998].

1.2.3 Prosody

When comparing clear speech and casual speech, it was found that non-uniform changes in duration make speech more comprehensible, be it for slowing down, or for speeding up speech. Some studies found clear speech to have longer durations in general. When speakers lengthen their speech to be more clear, they lengthen many parts of the speech, especially the stable parts. However, the key here is *non-uniform* changes: faster speech is not necessarily less intelligible than normal or slow speech.

# occurrences	phonetic transcription
82	ae n
63	eh n
45	ix n
35	ax n
34	en
30	n
20	ae n dcl d
17	ih n
17	q ae n
11	ae n d
...	...

Table 1.2: Pronunciation variants of the word "and" in the Switchboard Transcription Project

It was also found that fundamental frequency (F0) range is greater in clear speech than in casual speech. Some found it to be even greater in read speech than in clear speech. There is also evidence that the F0 maxima/median is higher in clear speech than in casual speech. In contrast, amplitude does not seem to be a main factor in comparing clear speech against sloppy speech. It is found that only a subpopulation of speakers speak louder when trying to be clearer.

As speakers do not have enough time to plan their speech, sloppy speech contains much more ungrammatical pauses and disfluencies than clear speech. Disfluencies are especially worth more discussion, which we describe next.

1.2.4 Disfluencies

Shriberg showed that the majority of disfluencies in spontaneous speech can be analyzed as having a three-region surface structure: reparandum, editing phase and repair [Shriberg, 1999]. Typical disfluencies and their analyses are listed in Table 1.3. The initial attempt is known as the *reparandum*, a region that will later be replaced. The end of this region is called the *interruption point*, marked with a ".", where the speaker has detected some problem. The *editing phase* may be empty, contain a silent pause, or contain editing phrases or filled pauses ("I mean", "uh", "um"). The *repair* region, which can be either a repetition or a corrected version of the reparandum, restores the fluency. These regions are contiguous, and removal of the first two (reparandum and editing phase) yields a lexically "fluent" version. Most phonetic consequences of disfluency are located in the reparandum and the editing phase. But certain effects can also be seen in the repair (such as prosodic marking), which means we cannot simply remove the first two phases and expect a perfectly fluent repair.

Disfluencies affect a variety of phonetic aspects, including segment durations, in-

Type	(Prior context)	Reparandum	Editing Phase	Repair	(Continuation)
Filled pause		.	um		we're fine
	it's	.	uh		after five
Repetition	have	the .		the	tools
Repair	to	res- .		relax	at home
False start	all	this .		this	paper
		it's .			did you?

Table 1.3: Typical disfluencies and their analyses

tonation, voice quality, vowel quality and coarticulation patterns.

Proper modeling of disfluencies is important for speech recognition. For example, filled pauses occur so frequently that most speech recognition systems define special words for them. From a word-error-rate point of view, some of the disfluencies are more troublesome than others. Broken words (or partial words) usually disrupt recognition as there are no proper HMMs to account for them. Not only will they be recognized incorrectly, neighboring words may suffer as well. As for repetition, repairs, and false starts, as long as they are well-formed word sequences, they don't pose a severe problem for acoustic modeling. They have potential language modeling consequences though, since language models are usually trained on written text, where disfluencies are rare [Schultz and Rogina, 1995].

The ability to detect disfluencies is equally important. Even when ASR can produce accurate transcriptions of conversational speech, the presence of disfluencies is still annoying since it disrupts the flow of the text. It might be helpful to automatically detect and mark disfluencies so that the text can be rendered in a desirable way.

1.3 The WH-questions of Sloppy Speech: When, Why and How?

Is sloppiness in speech caused by ignorance or apathy?
I don't know and I don't care.
Williams Safire

If we know when sloppy speech occurs and how it differs from clear/read speech, we can potentially build a better model and deploy it at the right time. The previous sections provide the necessary background to answer these questions: when, why and how people speak sloppily.

Section 1.1 provides the answer to the first question. Sloppy speech is likely to occur when the conversation parties are familiar with each other, when the social setting is informal, and the speaker does not intend to achieve high intelligibility. For

example, we can expect important information to be more intelligible than chitchat, content words more intelligible than function words, and so on. This has been verified from data analysis. Fosler-Lussier et al. showed that a word is more likely to be reduced when it is predictable, and/or it is in fast speech [Fosler-Lussier and Morgan, 1998]. Jurafsky et al. showed that words with high language model probabilities tend to be shorter, more likely to have a reduced vowel, and more likely to have a deleted final `t` or `d` [Jurafsky et al., 2001]. This suggests that we could potentially use LM perplexities or dialog states to predict when and how reductions are likely to occur. In general, however, predicting reductions is difficult, since there are many factors and some of them, such as the internal state of the speaker, are not easy to model.

Section 1.2 looks at the question how sloppy speech differs from clear/read speech. A related question is whether reductions follow certain rules, and if so, what kinds of rules. At the articulator level, I suspect that reductions can be explained, to a large extent, from a mechanical point of view. If we have a working articulatory model, many reductions can probably be modeled by "target undershooting". At a higher level, it is likely that which reduction form to use is largely determined by a speaker's pronunciation habit.

It is also useful to consider the entire communication process, from both the production side and the perception side. Humans are obviously much more comfortable with sloppy speech than current ASR systems are. The reason is that since we invented sloppy speech, it must suit our purposes well, otherwise we would not be using it. Greenberg argues that there exists an auditory basis for pronunciation variation [Greenberg, 1998]:

> The auditory system is particularly sensitive and responsive to the beginning of sounds, be they speech, music or noise. Our sense of hearing evolved under considerable selection pressure to detect and decode constituents of the acoustic signal possessing potential biological significance. Onsets, by their very nature, are typically more informative than medial or terminal elements, serving both to alert, as well as to segment the incoming acoustic stream. $\cdots$ Over the course of a lifetime, control of pronunciation is beveled so as to take advantage of the ear's (and the brain's) predilection for onsets $\cdots$

"Cue trading" can also be explained from the communication perspective. In sloppy speech, the speaker only needs to supply enough discriminative information in order to be understood. For example, when it is obvious from the context what words are going to follow, the speaker can assume that a relaxed pronunciation will be fine. Humans choose to be sloppy almost always at non-essential places, where the impact on communication is minimal: function words are more likely to be reduced than content words; our tone/volume tapers off after we expressed the main idea and just want to finish a sentence.

It could be misleading, though, to think of a speaker as an efficient encoder. While we may exploit redundancies in various ways, we don't consciously adjust our

pronunciation patterns on the fly. Our pronunciation habits are established gradually over the years and are quite stable.

1.4 Thesis Structure

The thesis has two main parts: an approaches part where novel approaches are designed to effectively handle sloppy speech, and a systems part where we focus on building better systems for three large vocabulary tasks. We start with a brief overview of the current ASR framework and a detailed analysis of pronunciation modeling. This provides motivations for the proposed new approaches: flexible parameter tying, Gaussian transition modeling and thumbnail features, described in Chapter 3,4, and 5, respectively. The second part describes our efforts in system development for three tasks: Broadcast News (Chapter 6), Switchboard (Chapter 7) and meeting transcription (Chapter 8). All of them contain sloppy speech, but the amount and the degree of sloppiness vary. Chapter 9 concludes the thesis.

Chapter 2

Background and Related Work

In this chapter, we briefly review the current ASR (Automatic Speech Recognition) framework and related work in pronunciation modeling. The goal is to motivate approaches we undertake in this thesis, namely, to address weaknesses of the current HMM framework and to implicitly model pronunciation changes.

For a detailed review of LVCSR (Large Vocabulary Continuous Speech Recognition), readers are referred to [Young, 1995] and more recently [Young, 2001]; for a detailed review of pronunciation modeling, see [Strik and Cucchiarini, 1999] and [Saraclar et al., 2000].

2.1 Overview of Automatic Speech Recognition

Automatic speech recognition has made significant progress over the last decade. A modern LVCSR system is very complex, yet the basic framework remains largely unchanged. From a modeler's point of view, it can be roughly divided into several major components: front-end, acoustic model, lexical model and language model (Figure 2.1). Typically, the front-end parameterizes the input speech waveform into a sequence of observation vectors (frames); the acoustic model computes likelihood for each frame; lexicon specifies how a word is pronounced; the language model estimates a probability for each word sequence.

Armed with the aforementioned knowledge sources, the search problem is to find the best hypothesis through a huge space of all possible sentences:

$$H^* = \arg\max_H P(H|O) = \arg\max_H P(O|H)P(H)$$

where O (observation) is an utterance, H (hypothesis) a word sequence, $P(O|H)$ is the acoustic likelihood of an hypothesis, $P(H)$ is the language model probability, and H^* is the best hypothesis found. Traditionally, sophisticated data structures and algorithms are necessary to make the search problem tractable, due to the use of cross-word acoustic models and trigram (or higher order) language models. [Mohri

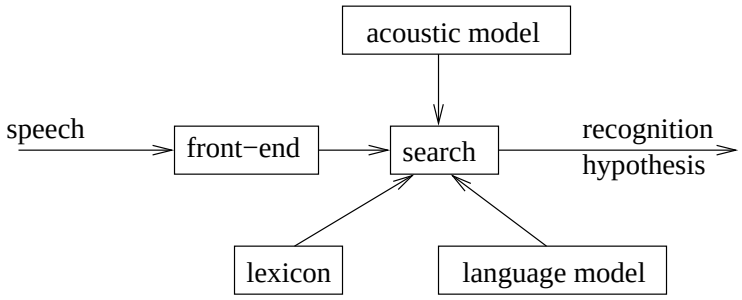


Figure 2.1: Large Vocabulary Speech Recognition

et al., 2002] showed that many of these components can be conveniently viewed as weighted finite-state transducers. Together, they define a huge transducer that converts a sequence of input frames into a sequence of words. Rather than dynamically expanding the search space, Mohri showed that it is possible to pre-compute the search network by exploiting redundancies in the transducer in a principled way.

2.1.1 Modeling Sequences with HMMs

The acoustic modeling problem is concerned with computing the probability $P(O|H)$. To make the problem tractable, two important steps are taken:

- O is parameterized as a sequence of frames $(o_1, \cdots, o_T)$;
- H is split into a linear sequence of models $(m_1, \cdots, m_N)$, where m_i models one third (begin, middle, or end) of a phone. This multi-stage process is illustrated in Figure 2.2. A word sequence is first converted into a phone sequence through dictionary lookup; triphones are simply phones in contexts; each triphone is then divided into three substates: begin/middle/end; finally, each sub-triphone is mapped to a mixture model through phonetic decision trees.

The benefit of these two steps is that we can now reduce the original problem $P(O|H)$, which involves two sequences, to a series of local computations, namely, $P(o_t|s_t)$, where o_t is the tth frame and s_t is the state/model at time t. The component model $P(o_t|s_t)$ is typically the well understood Gaussian mixture model.

To cast $P(o_1, \cdots, o_T|m_1, \cdots, m_N)$ in terms of $P(o_t|s_t)$, HMMs are the most favored framework. The application of HMMs to ASR is well explained in [Rabiner, 1989]. If we further take the Viterbi approximation and ignore HMM transition probabilities — both are common practices — $P(o_1, \cdots, o_T|m_1, \cdots, m_N)$ becomes simply $\prod_{t=1}^{T} P(o_t|s_t)$, where $(s_1, \cdots, s_T)$ is the most likely state alignment. Weaknesses of the current modeling practice becomes clear in this particular form[1].

[1]Note these weaknesses exist even without the Viterbi assumption, which is used here mainly to simplify the argument.

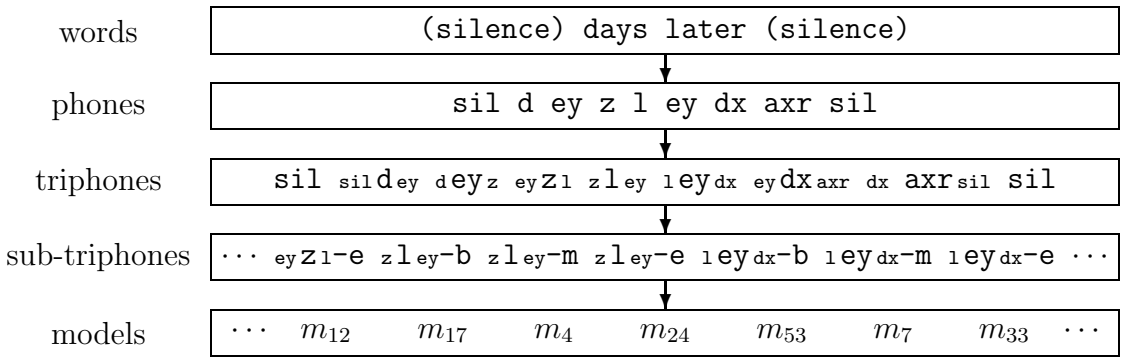


Figure 2.2: Mapping from words to models

- Each frame is assumed to be conditionally independent of any other frame given the current state. This is widely known as the frame independence assumption. Clearly there are correlations between successive speech frames since they are produced by a continuous process.

- The linear model sequence is criticized as a "beads-on-a-string" approach [Ostendorf et al., 1996b]. It ignores any higher level model structure and performs mostly template matching.

2.1.2 Modeling Variations

From a generative model point of view, acoustic modeling is all about modeling variations. Variations could come from different speakers, different speaking styles, different acoustic environments, speaking rates, and so on. Various mechanisms are deployed to deal with them: Vocal Tract Length Normalization (VTLN), Cepstral Mean Normalization (CMN), Speaker Adaptive Training (SAT), pronunciation networks, and mixture models [2]. The first three belong to the category of normalization approaches, each trying to factor out a particular cause of variation:

- By estimating speaker-specific warping factors, VTLN allows building gender-normalized models [Kamm et al., 1995, Lee and Rose, 1996, Eide and Gish, 1996, Zhan and Westphal, 1997];

- By estimating channel-specific offsets and variations, CMN makes models less sensitive to channel conditions [Acero, 1990, Westphal, 1997].

- By estimating speaker-specific transforms, SAT trains acoustic models in a speaker-normalized space [Anastasakos et al., 1996, Bacchiani, 2001].

[2] Decision-tree-based state tying is also one such mechanism. It helps to robustly model variations caused by different phonetic contexts.

Variations in pronunciations can be accommodated by adding alternative model sequences, such as pronunciation variants or pronunciation networks, which we will discuss in the next section.

Variations not captured by the aforementioned mechanisms are left to mixture models. The probability density function of a single Gaussian is unimodal. By increasing the number of components, a mixture model becomes increasingly multimodal. In theory, it can approximate any arbitrary distribution.

From this discussion, one can already see that there are several ways to model pronunciation variations. We can either add an extra pronunciation variant, or increase the number of Gaussians in a mixture model, or even use speaker adaptation techniques. It might be difficult to determine what is the best option for a particular situation, but it should be clear that the boundary between pronunciation modeling and acoustic modeling is not clearly drawn.

2.2 Pronunciation Modeling at the Phone Level

Pronunciation modeling for sloppy speech has received lots of attention since the start of the Switchboard evaluation. The majority of pronunciation modeling work has focused on the phone level, by adding pronunciation variants to the lexicon [Lamel and Adda, 1996, Finke and Waibel, 1997, Holter and Svendsen, 1997, Nock and Young, 1998, Ravishankar and Eskenazi, 1993, Sloboda and Waibel, 1996, Byrne et al., 1998]. This is the most natural choice, since phones are the de facto unit in lexical representation. Variants may be added by either enumerating alternative pronunciations or using pronunciation networks, as shown in Figure 2.3.

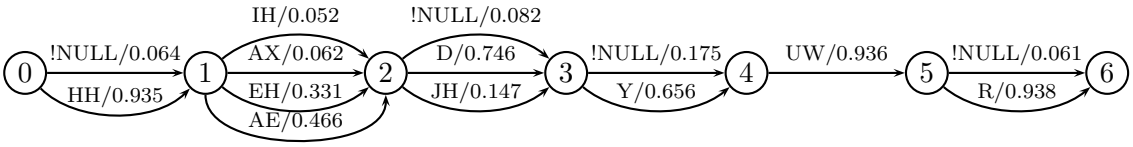


Figure 2.3: Pronunciation network for the word sequence `HAD YOUR` (!NULL indicates a deletion) (Reproduced from the WS97 pronunciation modeling group final presentation by M. Riley, et al.)

Cheating experiments have shown that WERs can be greatly reduced, when the "right" pronunciation variants are added to the dictionary [Saraclar et al., 2000, McAllaster et al., 1998].

However, finding the "right" variants proves to be a difficult task. The first problem is to obtain phonetic transcriptions for all the words. This can be done either manually or automatically. Manually editing a lexicon has the drawbacks of being both tedious, time consuming and error-prone. Automatic procedures include

- rule-based approach [Nock and Young, 1998, Ravishankar and Eskenazi, 1993], where phonological rules are used to generate alternative pronunciations. Rules

are typically obtained from linguistic studies and are therefore hand-crafted;

- phone recognizers [Nock and Young, 1998, Ravishankar and Eskenazi, 1993, Sloboda and Waibel, 1996];
- decision trees [Fosler-Lussier and Morgan, 1998, Riley et al., 1999], which map baseform pronunciations to surface forms estimated from either phonetically labeled data or automatic phone transcriptions;
- and many others.

While adding pronunciation variants increases the chance that a word can be better matched to the speech signal, it also increases acoustic confusability within the lexicon. One needs to be careful about how much to add, in order to balance between solving old errors and introducing new ones. Different criteria have been proposed to select variants, such as the frequency of occurances [Ravishankar and Eskenazi, 1993, Riley et al., 1999], maximum likelihoods [Holter and Svendsen, 1997], confidence measures [Sloboda and Waibel, 1996], or the degree of confusability between the variants [Sloboda and Waibel, 1996]. Several works find it helpful to use pronunciation probabilities with each pronunciation variant [Peskin et al., 1999, Finke et al., 1997]. Multi-words can also be added to the lexicon, in an attempt to model cross-word variation at the lexicon level [Nock and Young, 1998, Finke et al., 1997]. For example, `to` can be pronounced as `AX`, when preceded by `going`, as in:

```
GOING_TO    G AA N AX
```

Despite extensive efforts, improvements from this line of work are quite small.

2.3 Revisiting Pronunciation Modeling

Phone-level pronunciation modeling focuses on expanding dictionaries so that they contain more phone sequences (surface forms) that better match with acoustics. There are two major issues in this approach. First, dictionaries, as well as phone sets, are essentially a phonemic representation, not a phonetic representation. Second, phone sequences are only an intermediate representation, which will further be mapped to model sequences by decision trees. It is model sequences – not phone sequences – that we ultimately use for training and decoding. We will elaborate on these two issues next.

2.3.1 Issues with Dictionaries and Phone Sets

A dictionary is the knowledge source that describes how each word is pronounced, in terms of a set of phones. The acoustic identity of a word is completely determined by a dictionary. If two words have exactly the same pronunciation in the dictionary

(homophones), they would be acoustically indistinguishable. A good dictionary is therefore essential for good system performance. A phone set is equally important, since it specifies the granularity of the dictionary.

Over the years, various research groups have developed different dictionaries and phone sets, such as the Sphinx dictionary, Pronlex (or Comlex), the LIMSI dictionary, as well as our own dictionary. While they may have been derived from a limited number of sources, their differences are significant and reflect beliefs of different researchers on what constitutes a good design. Seldom can one find two dictionaries that use exactly the same phone set. The phone sets are all different for the four dictionaries mentioned above, let alone the actual pronunciation entries. To illustrate the issues in dictionary design, we quote here from the PRONLEX [PRONLEX] documentation:

> $\cdots$ Our idea is that the best current basis for speech recognition is to start with a simple and internally-consistent surface phonemic (allophonic) representation of citation forms in standard American dialect(s). Predictable variation due to dialect, reduction, or transcription uncertainty will be added in a second stage. In each such case, we have tried to define a standard transcription that will be suitable to support generation of the set of variant forms.
>
> An illustrative example: some American dialects distinguish the vowels in `sawed` and `sod`, while others do not; the ending `-ing` can be pronounced with a vowel more like `heed` or one more like `hid`, and with a final consonant like that of `sing` or like that of `sin`. This does not take account of considerable variation of actual quality in these sounds: thus some (New Yorkers) pronounce the vowel of `sawed` as a sequence of a vowel like that in `Sue` followed by one like that in `Bud`, while in less stigmatized dialects it is a single vowel (that may or may not be like that in `sod`).
>
> Combining all these variants for the transcription of the word `dogging` we would get 12 pronunciations — three versions of the first vowel, two versions of the second vowel, and two versions of the final consonant. Then someone else comes along to tell us that some Midwestern speakers not only merge the vowels in `sawed` and `sod` but also move both of them towards the front of mouth, with a sound similar (in extreme cases) to the more standard pronunciation of `sad`. Now we have $4 \times 2 \times 2 = 16$ pronunciations for the simple word `dogging` — with a comparable 16 available for `logging` and `hogging` and so forth, and plenty of variants yet to catalogue.
>
> Our approach is to give just one pronunciation in such a case. Some speech recognition researchers will want to use our lexicon to generate a network of predictable alternative transcriptions, taking account of dialect variation and reduction phenomena. Others may prefer to let statistical

> modeling of acoustic correlates handle some or all of such variation.
>
> We want to present a consistent transcription for each lexical set — so that in our example, dogging is not transcribed in one of the 16 ways while a second, different choice is made for logging, and a third one for hogging. We also want to choose a transcription that will support generation of all variants, so that distinctions made in some dialects should be made in our transcription if possible. Finally, we do want the transcription to indicate those variants that are lexically specific. Thus many cases of the prefix re- have both reduced and full variants (e.g. reduction), but many others do not (e.g. recapitalization). The difference apparently depends on how separable the prefix is from the rest of the word, but our lexicon simply has to list explicitly the cases that permit reduction. ···

This design issue is more acute for sloppy speech. People tend to introduce more phones and more variants to account for the fact that there are more variations/reductions in sloppy speech. For example, the last vowel of the word AFFECTIONATE can be pronounced in a slightly reduced way. Some dictionaries model this by using two variants: AX (as in DATA and IX (as in CREDIT).

In a sense, the central issue we are grappling with is a choice between *phonemic* representations and *phonetic* representations. Phonemes are well defined. The minimal pair test is used to decide whether a new phoneme is necessary or not. Minimal pairs are pairs of words whose pronunciation differ at only one segment, such as sheep and ship, lice and rice. Linguists introduce a new phoneme only when such a minimal pair involving that phoneme can be found. Since the definition of phonemes does not rely on sound similarity or dissimilarity, it is easy to maintain simplicity and consistency in a dictionary. In contrast, if we are to adopt a phonetic representation — if we try to faithfully capture the phonetic details — we would lose the simplicity and almost inevitably the consistency. First, the dictionary to start with is most likely a phonemic one. Second, transcribing sound with a finite set of phones is like quantization. There will be uncertainties with boundary cases, which open the door for mistakes and inconsistencies. As the PRONLEX documentation above illustrated, it is better to keep the dictionary simple and leave predictable variations, transcription uncertainties to automatic procedures.

2.3.2 Implicit Pronunciation Modeling

As indicated in Section 2.1.2, there are several mechanisms we could use to model pronunciation variations. Besides pronunciation variants or networks, we can always use the poor-man's model — adding more Gaussians to the mixture model. Phonetic decision trees are an important, yet commonly ignored, part of pronunciation modeling as well.

As shown in Figure 2.4, a given word sequence is first converted — by looking up a dictionary — into a phone sequence, which is subsequently translated into a

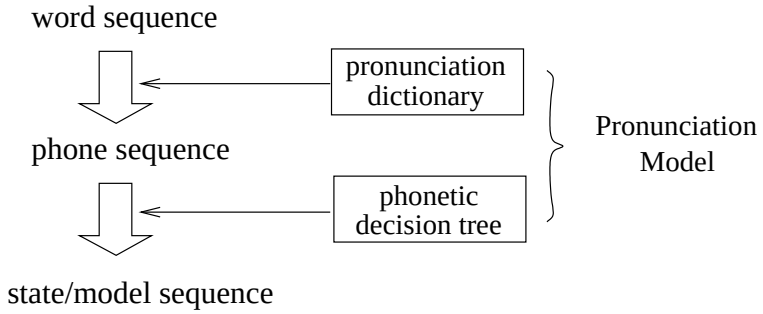


Figure 2.4: Pronunciation model as a mapping from symbolic level to model level

state/model sequence using a phonetic decision tree. The state sequence is ultimately used to align with acoustic observations, while the phone sequence is merely an intermediate representation. If we take a broader view of pronunciation modeling as the mapping from a symbolic (word) level to a model/state level, we can see that decision trees, or state tying schemes in general, are an integral part of this process.

As a historical comment, decision tree based state tying was originally introduced to model coarticulation effects. For example, the flapping of `T` in `BETTER` can be modeled by introducing an alternative lexical entry `BETTER(2)`:

```
BETTER      B EH T AXR
BETTER(2)   B EH DX AXR
```

where `DX` is the "flap" phone. But not every `T` can be flapped. One has to go through the entire dictionary to mark all such cases. This is tedious and error-prone. Since most flapping can be predicted from contexts, an alternative solution is to keep the dictionary unchanged, and instead let the decision tree to automatically determine the right model for each `T`. This example illustrates the idea of implicit pronunciation modeling. Rather than explicitly representing pronunciation changes in the dictionary, we try to model them implicitly through decision trees.

Implicit pronunciation modeling can be achieved through other means as well. In the example of `AFFECTIONATE`, some use the following two variants to allow for reductions in the final syllable:

```
AFFECTIONATE    AX F EH K SH AX N AX T
AFFECTIONATE(2) AX F EH K SH AX N IX T
```

We suspect that there is a continuous spectrum between the vowel `IX` and `AX`, all of them being a potential reduction form. Hence, just adding one variant to the dictionary won't capture all possible variations. As a matter of fact, both PRONLEX and the LIMSI dictionary use only `AX`. The phone `IX` is not used at all. In this case all the variations are captured by the Gaussian mixture model of `AX`. This is again an example for implicit pronunciation modeling.

The idea of implicit pronunciation modeling was first proposed by Hain [Hain, 2002], together with the term single pronunciation dictionary. State level pronunciation modeling [Saraclar et al., 2000] also keeps a simple dictionary, while modeling pronunciation variations at the state level. The first two approaches in this thesis are motivated by implicit pronunciation modeling. Flexible parameter tying aims at improving the traditional state tying scheme; Gaussian transition modeling is conceived to be a pronunciation network at the Gaussian level.

2.3.3 Drawbacks of Explicit Pronunciation Modeling

To emphasize, here we list the drawbacks of explicit pronunciation modeling, i.e. directly modifying a dictionary:

- Manually editing a lexicon is both labor intensive and error prone;
- Explicit pronunciation modeling focuses primarily on the phone level. The role of decision trees and the mapping to the model level are largely overlooked;
- Adding variants increases lexical confusability during recognition;
- If not performed properly, adding variants can also increase model confusability. As illustrated in Figure 2.5, when a variant replaces phone A by phone B, we are distributing to model B the data that was originally used to train model A. In cases where the variant is spurious, model B will be contaminated with data belonging to A, making A and B more confusable.

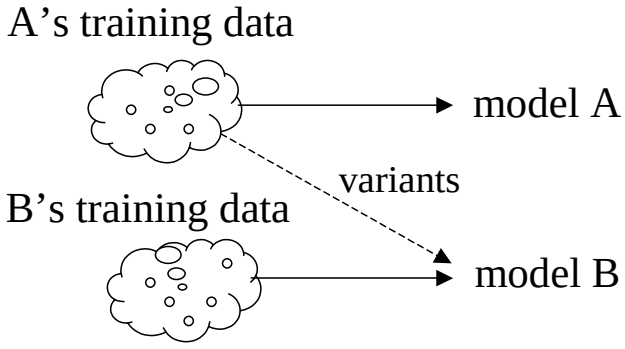


Figure 2.5: Model Contamination

In summary, explicit pronunciation modeling by adding pronunciation variants is not as simple as it seems to be. It should be exercised with great care.

2.4 Roadmap for the Proposed Approaches

We follow two main themes in this thesis to better model sloppy speech. First, we share with previous works the intuition that pronunciation modeling is important. Two of the proposed approaches are designed for this purpose: flexible parameter tying (Chapter 3) and Gaussian transition modeling (Chapter 4). Second, we try to alleviate invalid assumptions in HMMs, such as the frame independence assumption. While they may not be a serious limitation for read speech, they are now for sloppy speech due to increased coarticulation and reductions. Both Gaussian transition models and thumbnail features (Chapter 5) belong to this category.

2.5 Related Work

Many researchers have sought to move beyond the phone level. We break down these efforts into the following categories:

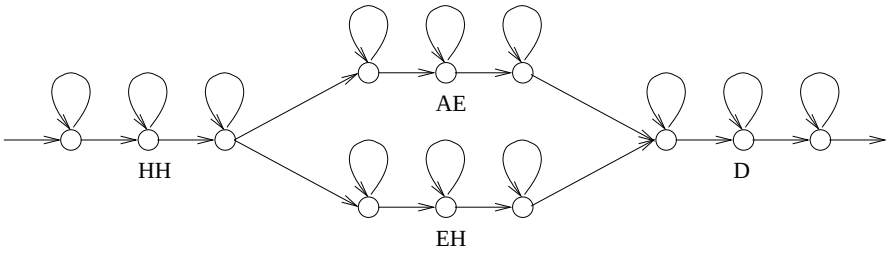
Syllable-based Modeling Greenberg [Greenberg, 1998] argued that it is better to model pronunciations at the syllable level. Syllables are much more stable than phones, in the sense that they are easily identifiable and much less likely to be deleted in sloppy speech. However, how to model reductions remains to be a key issue. Switching to the syllable level doesn't necessarily make it easier. Another issue is trainability, as the number of syllables is far greater than the number of phones. In the 1997 Johns Hopkins summer workshop, researchers tried a hybrid phone/syllable approach, where they model the most frequent syllables on top of a conventional triphone system [Ganapathiraju et al., 1997]. This approach yielded small improvements.

Articulatory Feature based Modeling Deng proposed a feature-based modeling scheme to better integrate modern phonological theory, namely, autosegmental phonology and articulatory phonology into ASR [Deng, 1997]. The basic idea is to represent each phone as a vector of articulatory features, such as voicing, lip rounding, tongue position, etc. When constructing a word model, feature spreading and overlapping are allowed to model asynchrony between different articulators. Therefore it offers a more flexible lexical representation over the conventional "beads-on-a-string" model. Ultimately, a word is represented as a fixed finite-state graph, where each state is specified by the articulatory features. Deng argues this is also a more parsimonious state space, compared to conventional triphone modeling. We feel this scheme relies critically on the soundness of the underlying linguistic theory. New developments in linguistic research on sloppy speech representation, as well as better integration with the data-driven framework, may be crucial to its success.

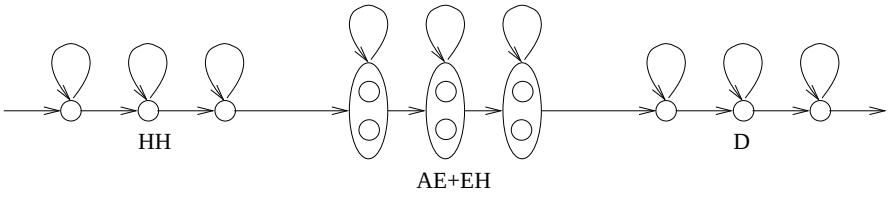
Another model for asynchrony in speech production is loosely coupled HMMs [Nock and Young, 2000]. Instead of modeling with a single HMM chain, mul-

tiple HMM chains are used, each representing an articulator. Observations are conditioned on all HMM chains, which are loosely coupled to accommodate asynchronous transitions. Training can be made tractable under certain approximations, but these approximations have serious limitations and should be revisited.

State-Level Pronunciation Modeling Based on research from the 1997 Johns Hopkins pronunciation modeling workshop, Saraclar proposed state-level pronunciation modeling [Saraclar et al., 2000]. Rather than introducing pronunciation variants at the phone level, he chose instead to model variations with more mixture components at the state level. As an example, to allow the word `HAD` (baseform `/HH AE D/` to be alternatively realized as the surface form `/HH EH D/`, he allows the model of `AE` to share Gaussians from `EH`, as shown in Figure 2.6. This approach was motivated in part by soft-tying [Luo and Jelinek, 1999]. A small improvement is observed when compared to the traditional approach of adding pronunciation variants.



(a) Phone−level pronunciation model (pronunciation variants)



(b) State−level pronunciation model

Figure 2.6: State-level pronunciation model for phone `AE` to be realized as phone `EH`, context independent models.

Implicit Pronunciation Modeling Recently, Hain proposed implicit pronunciation modeling [Hain, 2002], which shares much of the same motivation with us. As shown in Figure 2.2, the mapping from a phone sequence to a model sequence is traditionally provided by the phonetic decision tree, and is completely deterministic. Hain introduced hidden model sequence HMMs, as shown in Figure 2.7, as a replacement for the decision tree, so that a probabilistic mapping between the two sequences can be achieved.

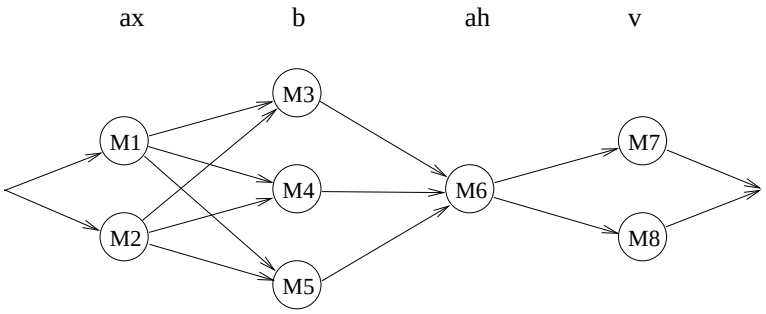


Figure 2.7: Hidden model sequence model for the word `above`. Each node is a phone level HMM. Probabilities are associated with transitions.

The new model (mapping) can be estimated using the EM (expectation maximization) algorithm. To have a consistent pronunciation dictionary to begin with, he also introduces the concept of single pronunciation dictionary, where only a single pronunciation is allowed for each word. The single pronunciation dictionary can be extracted from a multi-pronunciation dictionary using mostly a frequency-based criterion. Hain showed that the single pronunciation dictionary achieves similar or better performance over the original multi-pronunciation dictionary on both the Wall Street Journal task and the Switchboard task. Hidden sequence modeling gives an additional small improvement on top of single pronunciation dictionary.

Mode-Dependent Pronunciation Modeling This line of work attempts to actively predict pronunciation changes to curtail increased confusability [Ostendorf et al., 1996a]. The idea is that if pronunciation changes are systematic, one could potentially be able to predict the set of possible pronunciation variants in a certain context. Mode is a hidden variable that reflects the speaking style, such as sloppy, clear, or exaggerated. Mode can be conditioned on various acoustic features such as speaking rates and SNR (Signal to Noise Ratio), or linguistic features such as the syntactic structure. Along the same line, Finke proposed to weigh each variant by a factor depending on the "speaking mode" [Finke et al., 1999]. Speaking mode can be either speaker identity, speech rate, emotion, or anything that helps to predict the exact form of pronunciation. This is a powerful idea, but many details still need to be worked out, for example, how to robustly identify the speaking mode.

Joint Lexicon and Acoustic Unit Inventory Design While not specifically targeted at sloppy speech, joint lexicon and acoustic unit inventory design seeks a fully data-driven representation of the lexicon. A main motivation for this work is to improve the quality of a lexicon. The current lexicon design is purely ad hoc and error prone. Perhaps more importantly, it is likely to be inconsistent with the data-driven nature of acoustic modeling. Bacchiani et al. [Bacchiani

and Ostendorf, 1999], Holter et al. [Holter and Svendsen, 1997], and Singh et al. [Singh et al., 2002] have shown positive results on the Resource Management task. This is an ambitious line of work. As they pointed out, there are many important issues to be explored, such as the assumption of a single linear pronunciation per word and how to generalize to unseen words.

Part I

Approaches

Chapter 3

Flexible Parameter Tying

As an integral part of pronunciation modeling, parameter tying schemes, such as phonetic decision trees, determine the mapping from phones to models/states. In this chapter, we first review decision tree based state tying, then introduce two new parameter tying methods for implicit pronunciation modeling: Gaussian tying and enhanced tree clustering.

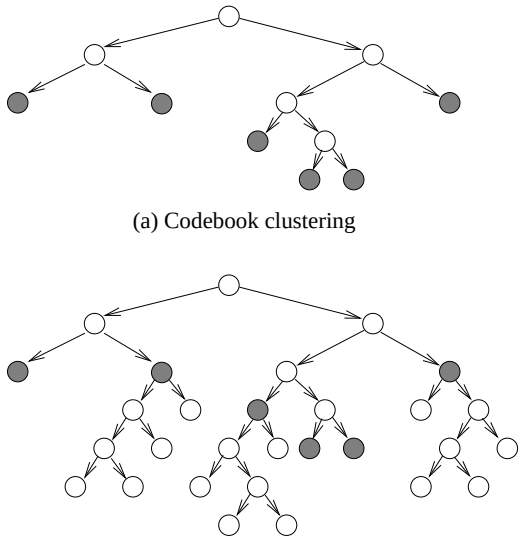
3.1 Decision-Tree-Based State Tying

State tying arises out of the need for robust context dependent modeling. Realization of a phone is heavily influenced by its neighboring phones, a phenomenon known as coarticulation. It is therefore necessary to treat each phone differently depending on its left and right contexts. A phone with a context window of 1 — the preceding phone and the following phone — is called a *triphone*. Since there are a large number of them, it is impossible to model every triphone individually. Decision trees are typically used to cluster them to achieve parameter sharing [Young et al., 1994]. As illustrated in Figure 2.4, decision trees also play an important role in the mapping from phones to models. Here we give a brief overview of the Janus tying scheme and also introduce several important Janus terms which will be used later on.

The context dependent modeling unit in Janus is called a *polyphone*. Compared to triphones, polyphones can have wider contexts. They can be either triphones, quinphones (± 2 phones), septaphones(± 3 phones), and so on.

In Janus, the parameters of a Gaussian mixture model are split into two components: a *codebook*, which corresponds to a set of Gaussians; and a *distribution*, which specifies the mixture weights. This division allows for more flexibility in parameter tying. Two polyphones can share both the distribution and the codebook, in other words, the same model; or they can share just the codebook, but have different mixture weights; or they can use completely different codebooks.

Janus uses a two-stage decision-tree-based state tying scheme [Finke and Rogina, 1997]. The same tree is used for tying both the codebooks and the distributions.



(a) Codebook clustering

(b) Distribution clustering

Figure 3.1: Two stage clustering of acoustic models. After the first stage, one codebook is created for each leaf node (shaded node). The second stage only adds more distributions.

The "standard" Janus decision tree growing scheme is shown in Figure 3.1. In the first stage, the decision tree is grown to a certain size, say, 2000 leaf nodes. A fully continuous Gaussian mixture model (a codebook and a distribution) is assigned to each leaf node. In the second stage, we continue to grow the tree to the final size of, say, 6000 leaf nodes. Only a distribution (mixture weights) is defined and trained for each leaf; the number of codebooks remains unchanged. Overall, we have 6000 tied states, which are parameterized as 6000 distributions sharing 2000 codebooks. Since the second stage introduces only more mixture weights, there is only a small increase in the overall number of parameters. This is similar in essence to soft-tying [Luo and Jelinek, 1999].

Note that the second stage can be skipped, in which case we have exactly one distribution for each codebook, a configuration known as "fully-continuous" system in some literature. Since the second stage increases modeling granularity without heavily increasing the number of parameters, two-stage clustering typically leads to better performance, compared to a fully-continuous system with the same number of parameters.

The criterion for tree splitting can be either information gain or likelihood changes. A polyphone is typically modeled by a Gaussian mixture model, i.e. a distribution over some underlying codebook (for example, context independent codebooks). We have also tried to use a single Gaussian model for each polyphone, as is the case in HTK [Young et al., 1995]. This has certain advantages in implementation: we do not need any codebooks to begin with, and parameter estimation for a single Gaussian

is much easier than updating a Gaussian mixture model. In terms of performance, these two implementations turn out to be comparable.

3.2 Flexible Parameter Tying

3.2.1 Why Flexible Parameter Tying

In the traditional decision tree based tying scheme, one decision tree is grown for each sub-state (begin/middle/end) of each phone. With 50 phonemes in the phone set, 150 separate trees are grown (Figure 3.2). Since parameter tying is only possible within the same tree, the drawback is that no parameter sharing is allowed across different phones or sub-states. However, this needs not to be the case. For example, it is reasonable to expect that neighboring states, such as the end state of one phone and the beginning state of the next phone, may have many similarities. A more flexible tying scheme would be more desirable.

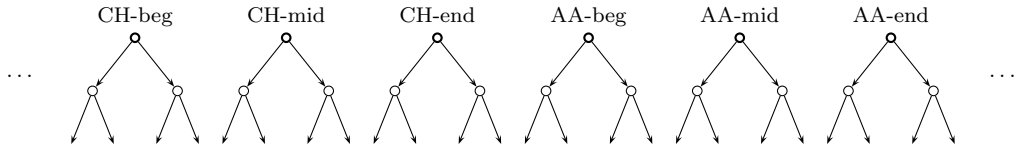


Figure 3.2: The traditional clustering approach: one tree per phone and sub-state

Flexible tying is especially needed for sloppy speech. Due to reduced pronunciation efforts, different phones (especially vowels) tend to exhibit more similarities. For example, vowel formants tend to take more central values in the "vowel triangle", as opposed to more extreme values in clear/read speech [Eskenazi, 1993]. There are also evidences in the Switchboard transcription project, where increased confusability causes difficulties in determining the phonetic identities for certain segments, which leads to slow transcription progress and low transcriber agreement [Greenberg et al., 1996].

Sharing parameters across phones also alleviates certain problems in a dictionary, namely, over-specification and inconsistencies. Examples of these include the handling of `T` and `DX`, `AX` and `IX`, as mentioned in Section 2.3.2. Some dictionaries choose to differentiate them, while others do not. In dictionaries that do, they are probably not marked consistently throughout the dictionary. By allowing parameter sharing across phones, we no longer face this tough decision: if certain phones are indistinguishable under certain contexts, they will be allowed to share the same model; if they show significant differences under certain other contexts, they will be allowed to use different models.

Next we present two approaches to achieve flexible parameter tying.

3.2.2 Gaussian Tying

Gaussian tying is a direct extension of the state level pronunciation modeling (SLPM) [Saraclar et al., 2000], introduced in Section 2.5. If a baseform phone, say, `AX`, is alternately realized as the surface form `IX` in certain contexts, SLPM augments the mixture model of `AX` with all Gaussians from the mixture model of `IX`. Gaussian tying generalizes SLPM in that it allows sharing of a selected subset of Gaussians from the `IX` model (rather than all of them), hence providing a finer level of control.

Gaussian tying is a general parameter tying framework, covering all different families of tied mixture models. Let $\mathcal{G}$, the Gaussian pool, be the complete set of Gaussians in a system. A model, m_i, is then defined by mixture weights over a subset of $\mathcal{G}$.

$$p(\cdot|m_i) = \sum_{j \in S_i} \pi_{ij} g_j(\cdot)$$

where $S_i \subset \mathcal{G}$ is the set of Gaussians used by m_i. Any acoustic modeling scheme can be completely specified by the weights matrix (Π_{ij}), where the ith row represents model m_i, the jth column the jth Gaussian in $\mathcal{G}$. Hence Gaussian tying covers a wide range of tying schemes, such as tied mixture [Kimball and Ostendorf, 1993], senones [Hwang et al., 1996], genones [Digalakis et al., 1996] and soft tying [Luo and Jelinek, 1999].

Note that most tying schemes allow only tying within the same phone and the same sub-state (begin/middle/end), corresponding to a very constrained form of the tying matrix. But as we discussed earlier, this may be undesirable.

We applied Gaussian tying as a way to fix this deficiency of an existing state-tying framework. While Gaussian tying allows more flexibility, it is not trivial to determine the optimal form of tying. We explored the following approaches:

- *similarity based tying*: Gaussians that are close together in the model space are tied. Three similarity measures are tried:
  - Euclidean distance between Gaussian means (variances are ignored)
  - KL2 (symmetric Kullback-Leibler) distance:

$$KL2(A;B) = \frac{\sigma_A^2}{\sigma_B^2} + \frac{\sigma_B^2}{\sigma_A^2} + (\mu_A - \mu_B)^2(\frac{1}{\sigma_A^2} + \frac{1}{\sigma_B^2})$$

  - Likelihood loss: this measures how much we lose in likelihood if we choose to model the data as a single Gaussian rather than two.

  With a chosen distance measure, greedy iterative bottom-up clustering is performed until a desired number of Gaussians are tied. One design issue concerns the size and quality of a cluster. Big clusters are likely to be problematic, as it causes too much smoothing and reduces discrimination between models. We

tried two solutions: simply imposing a hard limit on the cluster size, or using complete linkage clustering rather than single linkage clustering in order to create "tight" clusters. In general, the improvement is quite small, despite extensive experimentation.

- *error corrective tying*: The idea is similar to [Nakamura, 1998]. We monitor competition between models on the training data. In a perfect world, the correct model, as specified by transcripts, should achieve the best likelihood. If there is a strong competing model, we can augment the reference model with Gaussians from the competing model.

  For error corrective tying, we consider two model sequences side by side: the "correct" model sequence obtained by viterbi alignment, and the most likely model/Gaussian sequence. If a particular Gaussian g of model m' "out-votes" a correct model m frequently enough, g is added to the mixture of m.

  We conducted a cheating experiment to verify the concept. Model competition statistics is collected on the test set itself, using reference text for the correct model sequence. On our Broadcast News system, we are able to reduce WER from 19.1% to 15.1% (on 1998 Hub4e test set) by error-corrective tying. However, the gain doesn't hold when we repeat the same procedure on the training data.

The ineffectiveness of these methods might be blamed on their post-processing nature. It can be difficult to fix an existing sub-optimal tying scheme (as determined by decision trees). This leads us to enhanced tree clustering.

3.2.3 Enhanced Tree Clustering

With enhanced tree clustering, a single decision tree is grown for all sub-states of all the phones (Figure 3.3). The clustering procedure starts with all polyphones at the root. Questions can be asked regarding the identity of the center phone and its neighboring phones, plus the sub-state identity (begin/middle/end). At each node, the question that yields the highest information gain is chosen and the tree is split. This process is repeated until either the tree reaches a certain size or a minimum count threshold is reached at a leaf node. Compared to the traditional multiple-tree approach, a single tree allows more flexible sharing of parameters. Any node can potentially be shared by multiple phones, as well as sub-states.

Using a single tree for clustering triphones has been proposed in several occasions. [Paul, 1997] pointed out that single tree clustering is useful when the amount of training (or adaptation) data is very small. Successive state splitting and maximum-likelihood successive state splitting can be used for HMM topology design and state tying based on both contextual and temporal factors, which effectively achieves cross-phone/cross-substate tying [Takami and Sagayama, 1992, Ostendorf and Singer, 1997].

However, single tree clustering has never been associated with or considered in the context of pronunciation modeling. Our thesis work is novel in this aspect.

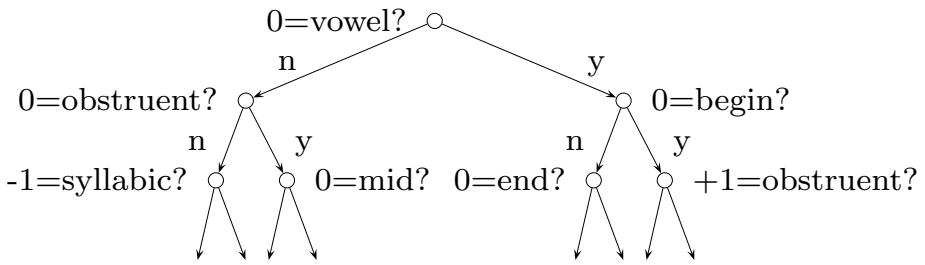


Figure 3.3: Enhanced clustering using a single tree. "−1=","0=","+1=" questions ask about the immediately left phone, the center phone, and the right phone, respectively.

Single tree clustering is quite easy to implement. The only change necessary is to expand the question set so that it includes questions about the identity of the center phone and its substate. Computational cost is the main difficulty for growing a single big tree, particularly for the first few splits. As the number of unique quinphones on the Switchboard task is around 600k, the cost of evaluating a question is quite high. The traditional approach does not have this problem, since polyphones are divided naturally according to center phones and sub-states, and each tree has only a fraction of the polyphones to work with. For this reason, we conducted two experiments to investigate the effects of cross-phone tying and cross-substate tying separately.

Experiment setup

Experiments are performed on the Switchboard (SWB) task. The test set is a 1 hour subset of the 2001 Hub5e evaluation set. The full training set includes 160 hours of SWB data and 17 hours of CallHome data. A 66 hour subset of the 160 hours SWB data is also used for fast experimentation. The front-end will be described in Section 7.1. We use a 15k vocabulary and a trigram language model trained on SWB and CallHome.

The baseline acoustic model is a quinphone model that uses a two-stage state tying scheme, with 24k distributions sharing 6k codebooks, and a total of 74k Gaussians. It has a word error rate (WER) of 34.4%. Unless otherwise stated, all results reported here are based on first-pass decoding, i.e. no adaptation or multi-pass decoding.

Cross-Substate Clustering

In this experiment, we build one tree for each phone, which covers all three sub-states. Three new questions regarding sub-state identities are added to the question set. We find these three questions to be highly important. For most phones, they are chosen

as the top two questions and the tree splits naturally into three subtrees, one for each substate. This result is therefore not any different from the traditional clustering where we explicitly use three separate trees.

This outcome is surprising. The fact that substate identity achieves the highest information gain seems to indicate that the model resolution is not sufficient along the time line. In other words, acoustic characteristics change significantly during the course of a phone, more so than any contextual effects. This probably merits further investigation.

Cross-Phone Clustering

We grow six triphone trees for cross-phone clustering: one for each of the substate of vowels and consonants. We could have built only three trees, without differentiating between vowels and consonants. The primary reason is to reduce computation. Even with the vowel/consonant division, it still takes 48 hours to grow a tree. The vowel/consonant division is chosen because we suspect that there is little parameter sharing between vowels and consonants.

As shown in Table 3.1, initial experiment on the 66 hours training subset gives a small, albeit significant, improvement (from 34.4% to 33.9%).

Single Pronunciation Dictionary

Motivated by Hain's work on single pronunciation dictionaries (SPD) [Hain, 2002], we tried to reduce the number of pronunciation variants in the dictionary. The procedure to derive a new lexicon is even simpler than Hain's. First, we count the frequency of pronunciation variants in the training data. Variants with a relative frequency of less than 20% are removed. For unobserved words, we keep only the baseform (which is more or less a random decision). Using this procedure, we reduced the dictionary from an average 2.2 variants per word to 1.1 variants per word. Unlike in [Hain, 2002], we are not strictly enforcing single pronunciation, so that we can keep the most popular variants, while pruning away spurious ones. For example, the word A has two variants in the final dictionary:

```
A       AX
A(2)    EY
```

Simply using SPD with traditional clustering gives a 0.3% improvement, which is comparable to Hain's results. More interestingly, cross-phone clustering responds quite well with SPD. Overall, we achieve a 1.3% gain by cross-phone clustering on a single pronunciation dictionary (Table 3.1).

So far we have used only the 66 hour training set and triphone clustering. The gain holds when we switch to the full 180 hour training data and quinphone clustering. Due to the high computational cost, we only compared two systems: one with multi-pronunciation lexicon and no cross-phone clustering, and the other with

Dictionary	Clustering	WER(%)	
		66 hrs, triphone	180 hrs, quinphone
multi-pronunciation	regular	34.4	33.4
	cross-phone	33.9	-
single pronunciation	regular	34.1	-
	cross-phone	33.1	31.6

Table 3.1: Cross-Phone Clustering Experiments. 66 hrs and 180 hrs denote the size of different training sets.

single-pronunciation lexicon and cross-phone clustering. WER improves from 33.4% to 31.6%, a 1.8% absolute gain.

Analysis

As the tree is grown in a purely data-driven fashion, one may wonder how much cross-phone sharing there actually is. Could it be possible that questions regarding center phones are highly important, and get asked earlier in the tree, resulting in a system which is not much different from a phonetically tied system? The top portion of the tree for the begin state of vowels is shown in Figure 3.4, which clearly shows that questions about center phone identities are not necessarily preferred over contextual questions. We also examined the leaf nodes of the six trees, and found 20% to 40% of the leaf nodes are shared by multiple phones. Consonants that are most frequently tied together are: `DX` and `HH`, `L` and `W`, `N` and `NG`. Vowels that are most frequently tied together are: `AXR` and `ER`, `AE` and `EH`, `AH` and `AX`. Table 3.2 lists phones that frequently share the same model in each tree. For each of the 6 trees, we list the top 5 most frequently shared phone tuples, together with the frequencies: how many times they end up at a same leaf node.

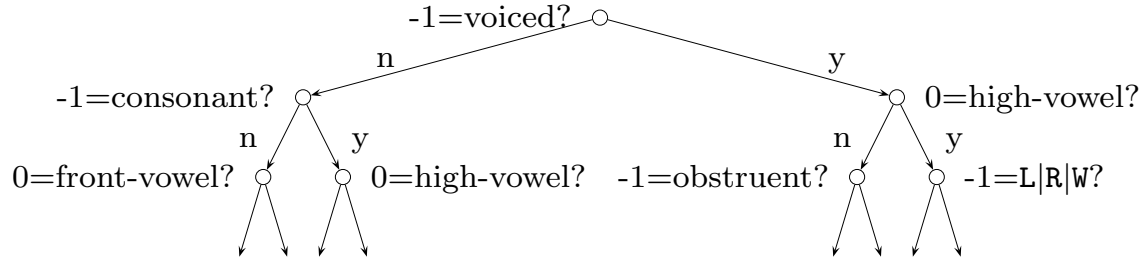


Figure 3.4: Top part of Vowel-beg tree (beginning state of vowels). "−1=" questions ask about the immediately left phone, "0=" questions ask about the center phone.

The fact that cross-phone clustering helps more with SPD (from 34.1% to 33.1%), than with a regular dictionary (from 34.4% to 33.9%), shows the advantage of implicit pronunciation modeling. Comparing to directly modifying the dictionary, it is better

vowel-beg		vowel-mid		vowel-end	
120	AE EH	114	EH IH	144	AH AX
87	IH IX UH	113	AXR ER	123	IH IX UH UW
82	IX UH	103	AX IX UW	117	AXR ER
69	IX UH UW	91	AY OY	69	AE EH
61	OW OY	63	IX UH	69	AXR EN ER
consonant-beg		consonant-mid		consonant-end	
164	N NG	166	DX HH	95	DX HH N Y
156	L W	118	DX N	84	DX HH
117	M NG	111	T TH	68	M NG
71	DX HH M N	96	L W	59	DX HH N NG Y
70	D T	85	D T	49	P T

Table 3.2: Phones that frequently share to the same model in each tree

to keep it simple and consistent, and use automatic procedures to model pronunciation variations.

3.3 Conclusion

We discussed why flexible parameter tying is important for the modeling of sloppy speech, and presented two novel techniques, of which enhanced tree clustering is found to improve recognition performance on the Switchboard task together with a simplified pronunciation dictionary.

It is worth mentioning that enhanced tree clustering could also be used for the modeling of multilingual speech and non-native speech. Phone sets are typically not well defined in those scenarios. The same IPA phones have slightly different counterparts in different languages. One may want to take advantage of their similarities to achieve robust modeling.

Chapter 4

Gaussian Transition Model

4.1 Motivation

4.1.1 Single Model Sequence or Multiple Model Sequences?

While flexible parameter tying improves the mapping from a phone sequence to a model sequence, it uses only a single model sequence for a given phone sequence. Together with a single pronunciation dictionary, only one model sequence is allowed for each word. Whereas in the pronunciation variants approach, since multiple variants are used, a word can be mapped to multiple model sequences.

Intuitively, the latter is better at capturing trajectory information. If there are indeed two distinct pronunciations, they can be correctly modeled by two model sequences. If a single model sequence is used instead, the two trajectories have to be collapsed into one, with local variations modeled by Gaussian mixture models, but the distinction between the two trajectories lost.

The reader may recall that there are arguments for the former approach as well. Indeed, as shown in Figure 2.6, state level pronunciation modeling favors merging two variants into a single model sequence and increasing the number of Gaussians in each mixture.

In this chapter, we introduce Gaussian transition models (GTM), which can be regarded as implicit pronunciation networks at the Gaussian level. This way, we do not need to explicitly change the dictionary, while still being able to capture different ways of pronunciation.

4.1.2 A Pronunciation Network at the Gaussian Level

To introduce the idea of GTM, consider the probability of a sequence of frames $\boldsymbol{O} = (\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T)$ given a sequence of states $\boldsymbol{S} = (\boldsymbol{s}_1, \cdots, \boldsymbol{s}_T)$, with each state modeled

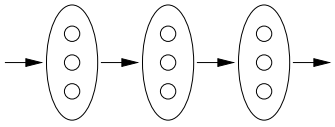
as a Gaussian mixture model:

$$p(\boldsymbol{O}|\boldsymbol{S}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{s}_t) \tag{4.1}$$

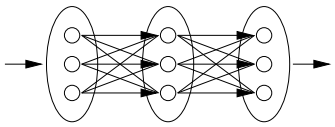
$$= \prod_{t=1}^{T} \sum_{k_t} \pi_{tk_t} g_{tk_t}(\boldsymbol{o}_t) \tag{4.2}$$

$$= \sum_{k_1} \sum_{k_2} \cdots \sum_{k_T} \prod_{t=1}^{T} \pi_{tk_t} g_{tk_t}(\boldsymbol{o}_t) \tag{4.3}$$

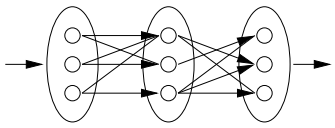
where $g_{tk}(\cdot)$ is the kth Gaussian in the tth state, π_{tk} is the mixture weight. Equation 4.1 follows from the conditional independence assumption of HMMs. Rearranging terms in Equation 4.2 leads to Equation 4.3.



(a) Linear Sequence of Mixture Models



(b) The Equivalent Full Gaussian Transition Model



(c) Pruned Gaussian Transition Model

Figure 4.1: Gaussian Transition Network

These equations have an intuitive interpretation, as shown in Figure 4.1. Figure 4.1(a) corresponds to Equation 4.1, a sequence of Gaussian mixture models. Figure 4.1(b) corresponds to Equation 4.3, a full Gaussian transition network. If we consider each Gaussian $g_{tk_t}(.)$ to be a modeling unit by itself, each $\prod_t g_{tk_t}(.)$ represents a unique trajectory, weighted by $\prod_t \pi_{tk_t}$. Hence, all possible trajectories are allowed in the traditional HMM-GMM (Equation 4.2).

GTM restricts the set of allowable trajectories by modeling transition probabilities between Gaussians in adjacent states.

$$a_{ij} = P(q_t = g_j | q_{t-1} = g_i) \tag{4.4}$$

where a_{ij} is the probability of transition from Gaussian g_i to Gaussian g_j, subject to the constraint $\sum_j a_{ij} = 1$. Figure 4.1(c) shows a GTM after pruning away unlikely transitions.

From the pronunciation modeling point of view, each trajectory $\prod_t g_{tk_t}(.)$ can be regarded as a particular pronunciation variant, $\prod_t \pi_{tk_t}$ is the corresponding pronunciation probability. Figure 4.1(c) is then a pronunciation network, similar to Figure 2.3, albeit at the Gaussian level rather than the phone level. It can potentially capture fine pronunciation changes that is too subtle to be represented as a different phone sequence.

4.1.3 The Conditional Independence Assumption

GTM also alleviates a well known problem in HMMs: the conditional independence assumption. Speech production differs from a random process in that articulators move along a low dimensional manifold. As a result, the speech trajectory is relatively smooth in the feature space. But an HMM, as a generative model, does not necessarily generate a smooth trajectory, due to the conditional independence assumption. Under this assumption, all speech frames are conditionally independent, given the hidden state sequence. This makes HMMs ineffective in modeling trajectories.

This is best illustrated by considering a gender independent HMM, using Gaussian Mixture Model (GMM) as output density function (Figure 4.2). Assuming certain mixture components are trained mostly on male speakers, while other components of the same mixture are trained on females. Sampling the HMM produces a frame sequence randomly switching between male and female at any time [1].

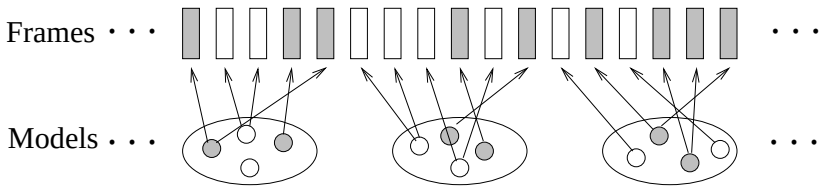


Figure 4.2: Deficiencies of HMM-GMM as a generative model (shaded area stands for male speech, non-shaded area stands for female.)

Variations in speaker, context and speaking style can all produce completely different trajectories for the same phone sequence. If modeled by a single state sequence as in a regular HMM, trajectories will be mixed up, resulting in poor discrimination between trajectories. By modeling Gaussian transitions, GTM allows us to capture trajectory information within the single state sequence. For example, in Figure 4.2, GTM can capture the difference between "legal" and "illegal" transitions: a "male" Gaussian is most likely to follow "male" Gaussians rather than "female" Gaussians.

[1]Sampling a Gaussian mixture model can be performed by first selecting a Gaussian, then sampling from the chosen Gaussian.

4.1.4 Relevant Approaches

In the case of sloppy speech, previous efforts have focused on dictionary expansion by adding alternative pronunciations. The net result of dictionary expansion is a phone level pronunciation network for each word. But many reductions are too subtle to be classified as either phoneme substitutions or deletions. Partial reduction or partial realization may actually be better modeled at a sub-phoneme level. Gaussian transition models provide a way to model such variations implicitly. It allows finer model resolution, compared to pronunciation modeling at either the phoneme level or even the state level [Saraclar et al., 2000].

Iyer et al. proposed the parallel path HMM to better model multiple trajectories [Iyer et al., 1998]. This is probably the closest proposal to GTM, and it also stays within the HMM framework. The basic idea is to use multiple paths for each phone-level HMM, therefore maintaining trajectory information and avoiding path mixing up. Paths are allowed to cross over only at phone boundaries. However, the number of parallel paths is normally quite limited (two or three). Choosing the right number of paths is also an unsolved problem.

In comparison, the GTM models multiple paths implicitly through Gaussian transitions. It is able to model a large number of trajectories. It also fits nicely into the HMM framework. However, as a first order model, GTM has a relatively short memory compared to parallel path HMMs.

Hidden model sequence modeling aims at deriving a stochastic mapping from phone sequences to HMMs automatically [Hain, 2001]. As shown in Figure 2.7, it can also model multiple trajectories for the same phone sequence. The optimization criterion is maximum likelihood, similar to that of GTM. While hidden model sequence model operates at the HMM (phone) level or the state level, GTM looks at the Gaussian level. GTM can therefore achieve a more fine-grained modeling of subtle pronunciation changes.

Segment models also attempt to exploit time-dependencies in the acoustic signal [Ostendorf et al., 1996b], by modeling trajectories either parametrically or non-parametrically. Since these approaches typically fall outside the HMM framework, it is hard to take full advantage of the efficient HMM training and recognition algorithms.

4.2 Training of the Gaussian Transition Model

The GTM models transition probabilities between Gaussians in adjacent states, as in Equation 4.4, where a_{ij} are the parameters to be estimated.

The GTM models can be readily trained using the standard Baum-Welch algorithm. Each Gaussian in a mixture will be treated as a state by itself. GTM parameters can then be estimated as regular HMM transition probabilities. Below we give the Baum-Welch formulae, following notations of [Rabiner, 1989]. The formulae

are essentially the same, only that each Gaussian is now a state by itself, i.e. i, j denote individual Gaussians rather than mixture models, and $b_j(\boldsymbol{o})$ is a single Gaussian distribution rather than a Gaussian mixture.

The forward probability:

$$\begin{aligned} \alpha_{t+1}(j) &= P(O_1 O_2 \cdots O_{t+1}, q_{t+1} = g_j) \\ &= \left(\sum_i \alpha_t(i) a_{ij} \right) b_j(\boldsymbol{o}_{t+1}) \end{aligned} \tag{4.5}$$

The backward probability:

$$\begin{aligned} \beta_t(i) &= P(O_{t+1} O_{t+2} \cdots O_T | q_t = g_i) \\ &= \sum_j a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j) \end{aligned} \tag{4.6}$$

Accumulating statistics:

$$\begin{aligned} \gamma_t(i) &= P(q_t = g_i) \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)} \end{aligned}$$

$$\begin{aligned} \xi_t(i,j) &= P(q_t = g_i, q_{t+1} = g_j) \\ &= \frac{\alpha_t(i) a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j)}{\sum_i \sum_j \alpha_t(i) a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j)} \end{aligned}$$

Updating transition probability:

$$a_{ij} = \frac{\sum_t \xi_t(i,j)}{\sum_t \gamma_t(i)}$$

Note that GTM changes the parameterization of the acoustic model. A conventional acoustic model consists of mixture weights, Gaussian means and variances, and state transition probabilities. With Gaussian transition probabilities, there are two changes.

- We have now two sets of transition probabilities, one for state transitions and another for Gaussian transitions. It is easier to think that transitions take place in two steps. First, we decide whether to stay in the same state, or transit into a different state, according to the HMM transition model. After knowing which state to transit to, we apply GTM to determine Gaussian occupancy probabilities, in other words, how likely it is to transit to a Gaussian in that state.

- Theoretically, mixture weights are no longer needed. They are replaced by GTMs. After the aforementioned two-step transition, both the forward probability and the backward probability are already determined for each Gaussian.

  Mixture weights can be regarded as a special case of GTM, where $a_{ij} = \pi_j$, in other words, the transition probability into Gaussian j is simply the mixture weight π_j, regardless of which Gaussian it comes from.

  In practice, due to data insufficiency, we can't always estimate a GTM for every state transition. In these scenarios, we naturally use mixture weights as a back-off solution. Details will be given in the next section on how to handle back-off transitions in GTM training.

4.2.1 Trainability and the Handling of Back-off Transitions

GTM can take a large number of parameters. First, a transition between two mixtures of n components each requires n^2 transition probabilities. Second, in an LVCSR system with thousands of mixture models, the potential number of transitions could be millions. Hence data sufficiency becomes a concern. In our experiments, we choose to model only frequent transitions, i.e. state transitions that receive enough counts during training. For everything else, we revert to the traditional HMM-GMM model: $a_{ij} = \pi_j$, where π_j is the mixture weight for the jth Gaussian.

This implies that we need to switch between full transition (GTM) cases and traditional GMM transitions at certain points in the forward-backward algorithm. For better understanding, it is helpful to consider a back-off transition between two GMMs as a "sum-and-distribute" operation (Figure 4.3).

Let us consider the computation of forward and backward probabilities. For forward probabilities (Equation 4.5), there are two cases:

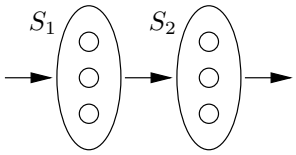
- For frequent state transitions, since they are modeled by GTMs, they are computed as is. No mixture weights are needed.

- For infrequent state transitions,

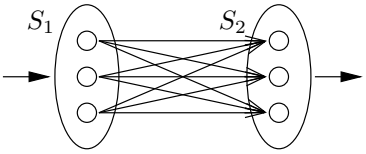
$$\left(\sum_{i \in S_1} \alpha_t(i) a_{ij}\right) b_j(\boldsymbol{o}_{t+1}) = \left(\sum_{i \in S_1} \alpha_t(i)\right) \pi_j b_j(\boldsymbol{o}_{t+1})$$

  where S_1 denotes one of the "from" states, π_j is the mixture weight for Gaussian j of the "to" state. As $\sum_{i \in S_1} \alpha_t(i)$ depends only on the "from" state, it can be stored as $\alpha_t(S_1)$ to save computation. This is the "sum" part in Figure 4.3. After we sum over all possible "from" states, each target Gaussian gets a piece of the total sum according to the corresponding mixture weight (the "distribution" step).

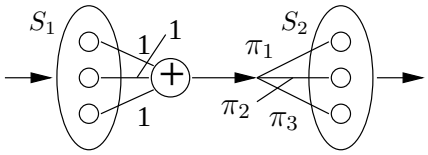
Similarly, there are two cases for backward probabilities:



(a) Transition between two mixture models



(b) The full Gaussian transition models



(c) The back−off case: sum and distribute

Figure 4.3: Handling back-off transitions in GTM training

- Frequent state transitions (with GTM models) are computed as is;
- For infrequent state transitions,

$$\sum_{j \in S_2} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j) = \sum_{j \in S_2} \pi_j b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j)$$

  where S_2 is the "to" state, π_j is the mixture weight for Gaussian j in S_2. It's obvious that this doesn't depend on the incoming state. To save computation, this quantity can be stored as $\beta_{t+1}(S_2)$.

Furthermore, it can be established that

$$\begin{aligned} \alpha_t(ik) &= \alpha_t(i) \frac{\pi_k g_{ik}(\boldsymbol{o}_t)}{\sum_m \pi_m g_{im}(\boldsymbol{o}_t)} \\ \alpha_t(i) &= \sum_k \alpha_t(ik) \\ \beta_t(ik) &= \beta_t(i), \forall k \end{aligned}$$

where i, k denotes the kth Gaussian in the ith state. $\alpha_t(i)$ and $\beta_t(i)$ are state-level (rather than Gaussian level) probabilities as defined before. Since they are used quite often, storing them separately speeds up training in the back off case.

With back-off transitions, we also need to accumulate statistics for mixture weights. When π_j is used in a back off case, we need to add

$$\begin{aligned}\sum_i \xi_t(i,j) &= \frac{\left(\sum_i \alpha_t(i)a_{ij}\right) b_j(\boldsymbol{o}_{t+1})\beta_{t+1}(j)}{P(\boldsymbol{O})} \\ &= \frac{\left(\sum_i \alpha_t(i)\right) \pi_j b_j(\boldsymbol{o}_{t+1})\beta_{t+1}(j)}{P(\boldsymbol{O})}\end{aligned}$$

to its sufficient statistics.

4.2.2 HMM-GMM as a Special Case of GTM

Based on what we have established in the previous section, if we always have $a_{ij} = \pi_j$, GTM reduces to the standard HMM-GMM. In other words, HMM-GMM is a special case of GTM where transition models — which happens to be mixture weights — are tied for all Gaussians in the same mixture.

4.2.3 Pruning

GTM, similar to HMM transition models, might suffer from the same problem that the dynamic range of their scores is too small, comparing to HMM observation probabilities. Final acoustic scores are normally dominated by observation scores. GTM, or HMM transition models have only a marginal impact on either scores or recognition accuracies. In our experience, HMM transition probabilities can often be ignored. HMM topologies, on the other hand, are quite important. As will be shown in Section 6.2, there could be a big difference by just adding or removing an extra transition to the "standard" 3-state left-to-right topology.

In terms of trajectory modeling, while GTM offers better discrimination between trajectories, all trajectories are nonetheless still permitted, including both "legal" and "illegal" ones. The difference in scores, introduced by GTM, might be overwhelmed by acoustic likelihoods.

One solution is to prune away unlikely transitions. As in Figure 4.1(c), this creates a transition network at the Gaussian level. Like removing the skipping arc in HMM topologies, this could lead to a model that is more compact and more prudent.

In reality, however, it is difficult to find a good pruning strategy. A frequency based approach is taken in this work. But it is possible that unseen trajectories will be pruned away due to a limited training set.

4.2.4 Computation

GTM training is computationally very demanding. A transition between two mixture models is now expanded to a full transition network between all Gaussians. Computational complexity is multiplied a factor of n^2, where n is the average number of Gaussians per mixture.

For back-off transitions, one can take advantage of the "sum and distribute" operation (Figure 4.3) to speed up the computation.

4.3 Experiments

Experiments are carried out on the Switchboard (SWB) task. The test set is a 1 hour subset of the 2001 Hub5e evaluation set. Acoustic training uses a 66 hours subset of the SWB data. The front-end will be described in Section 7.1. The acoustic model has roughly 6000 distributions, sharing 6000 codebooks, with a total of 86K Gaussians, on average 14 Gaussians per model. Cross-phone clustering is used together with single pronunciation dictionary, as described in Chapter 3. We use a 15k vocabulary and a trigram language model trained on SWB and CallHome.

As discussed before, we only model frequent model transitions with GTMs. Before training, we count the number of transitions for each model pair on the training data. Only state transitions with counts above a certain threshold are modeled with a GTM. Of the 6000 mixture models in the acoustic model set, a total of 40K model pairs (out of a potential 6K×6K=36M) has been observed. The most frequent 9400 model pairs are selected for GTM modeling. It turns out that most of them (6000 pairs, or 64%) are transitions within the same model (corresponding to self-loops in HMM).

Since each model/state transition is comprised of a number of Gaussian transition models — one for each Gaussian in the source model — there still may not be enough data to robustly update each individual GTM. Therefore, we also apply a minimum count criterion during training: a transition model is updated only if the Gaussian receives enough training counts, otherwise the model remains unchanged.

One iteration of Baum-Welch training gives significant improvement in term of likelihood. Log likelihood per frame improves from -50.67 to -49.18, while conventional HMM training can only improve the log likelihood by less than 0.1 on the same data. Considering the baseline acoustic model has already been highly optimized, this indicates improved acoustic modeling.

Pruning Threshold	Avg. # Transitions per Gaussian	WER (%)
baseline	14.4	34.1
1e-5	9.7	33.7
1e-3	6.6	33.7
0.01	4.6	33.6
0.05	2.7	33.9

Table 4.1: Gaussian Transition Modeling Experiments on Switchboard

We tried several different minimum count values. The best is found to be 20, at which point a majority of the transition models (86%) are updated. Within them, we

further prune Gaussian transitions if their probabilities fall below a certain threshold. Table 4.1 shows word error rates for GTM models pruned against different thresholds. N-Best rescoring is used in these experiments where N=100. The average number of transitions is measured for Gaussians in the updated transition models. The best performance, a 0.5% gain, is obtained after pruning away almost 2/3 of the transitions.

In further experiments, we tried to retrain on top of some pruned GTM models, as well as several different parameter settings, but found that the improvement is marginal at best, even worse in some cases.

4.4 Conclusion

In this chapter, we presented a new Gaussian transition model, which is motivated both as an implicit pronunciation modeling approach, and as a way to alleviate the frame independence assumption. Unfortunately, only a marginal gain has been observed in experiments.

One possible explanation is that the frame independence assumption has already been mitigated by the use of dynamic features in the front-end. By capturing dependencies between adjacent frames, dynamic features expand the coverage of the final feature vector from a single frame ($\sim$ 20 milliseconds) to a segment of typically $\sim$ 100 milliseconds. This, in a rough sense, is segmental modeling already, which makes it less effective to model dependencies at the model level.

There are several possibilities for future investigations. First, the pruning strategy needs to be revisited. Currently, the pruning threshold is set by hand, in a "trial-and-error" fashion. Also, frequency based pruning may not be well suited for structural discovery. Second, the current GTM is designed for mild variations in pronunciation, not deletions. It could be worthwhile to extend the GTM to handle deletions.

Chapter 5

Thumbnail Features

5.1 Motivation

Conventional speech analysis (front-end) provides a sequence of frames as input to the acoustic model. Due to the frame independence assumption, each triphone model can only have a very "short-sighted" view of the data, one frame at a time. Dynamic features — delta and double deltas — alleviate the problem by increasing the coverage (window size) of a frame, as will be explained in Section 6.3. This greatly improves performance. However, having a fixed window size may still be suboptimal. In Figure 5.1, if the same phoneme is realized twice in different speeds, it will end up with different number of frames. When a fixed window size is used for dynamic features, it will contain a different image each time: the entire phone the first time, but more than a phone the second time. This creates unnecessary variations that must be modeled by the underlying Gaussian mixture model.

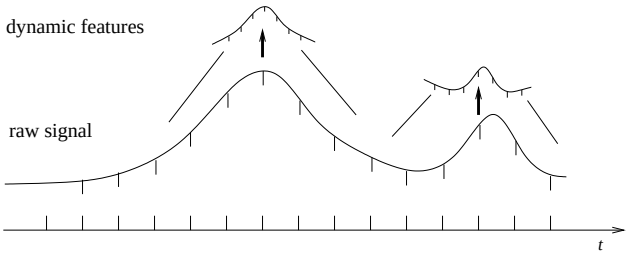


Figure 5.1: Problems with a fixed window size for dynamic features. The dynamic features shown are the results of stacking 7 adjacent frames (± 3). Due to speech rate differences, the two features have quite different shapes.

To overcome this problem, we propose the *thumbnail feature*, where we directly compute a feature vector on a segmental basis and model it as such. A segment can be either a senone (sub-phone), a syllable, a phone, two phones (diphone), etc. It is not limited to any fixed duration. In our work, we choose to subsample frames in

a segment to derive the segmental feature, which can be thought of as a thumbnail image of the entire senone/phone/syllable. In other words, we are now modeling each segment as whole, rather than a sequence of frames.

The idea of deriving segmental features by subsampling is not new. A good review of related works can be found in [Ostendorf et al., 1996b]. As discussed in [Ostendorf et al., 1996b], there are certain problems with this kind of approaches. First, segmental features are dependent on segment boundaries, which are not known a priori; second, different hypotheses are conditioned on different events, making a fair comparison between them difficult. In this chapter, we describe our particular solution to these issues. The MIT SUMMIT speech recognizer, a segment-based system, uses the so-called *anti-phone* models to cope with the score compatibility issue. We will discuss it in Section 5.4.

5.2 Computing Thumbnail Features

5.2.1 Hypothesis Driven Features

There is a salient distinction between thumbnail features and conventional features. Unlike conventional features (front-ends), thumbnail features cannot be computed beforehand and fixed during recognition. To compute thumbnail features, we need segmentation information, which is different for each hypothesis. In other words, each hypothesis has its own observation space.

We therefore face a "chicken-and-egg" problem. The features cannot be computed without a segmentation; but to find a reasonable segmentation (or the hypothesis that implies the segmentation), we need to begin with some acoustic feature. To avoid the problem of simultaneously searching for both the feature and the hypothesis, we adopt a two stage approach. A conventional system is used to find a set of hypotheses and their segmentation. Then, thumbnail features are computed and hypotheses are rescored using a second set of thumbnail models.

5.2.2 Score Compatibility

Comparing scores between hypotheses also becomes an issue. If we take the straightforward approach and compute one feature vector for each segment, it is likely that different hypotheses will have different number of feature vectors. Scores are no longer comparable. Hypotheses with fewer segments are favored because of the smaller number of probability terms.

One solution is to use a length-dependent weighting factor. Longer segments are weighted more, to approximate the traditional HMM scoring paradigm.

The solution we adopted here is to compute thumbnail features on a frame-by-frame basis. At any time point (frame), we assume there is a hypothetical segment centered around that point (frame), from which we can subsample to derive the

thumbnail feature. Some of these segments will coincide with the real segments; the others are transitions, or interpolations, between two real segments. This is illustrated in Figure 5.2, where a segment is defined to be 5 subphones. The derivation of two thumbnail features is shown. The first is located at the center (50%) of `AH-m`. 5 frames from the adjacent subphones, all at their respective 50% points, are extracted to form thumbnail. The second example is at the end of `AH-e`. Its thumbnail is formed by the last frames from ±2 subphone segments. In general, to compute a thumbnail feature at a particular time point, we first calculate its relative position in the current subphone; frames at the same relative position in neighboring subphones are extracted, and stacked together to form a super-vector, which is then projected down to a subspace using LDA.

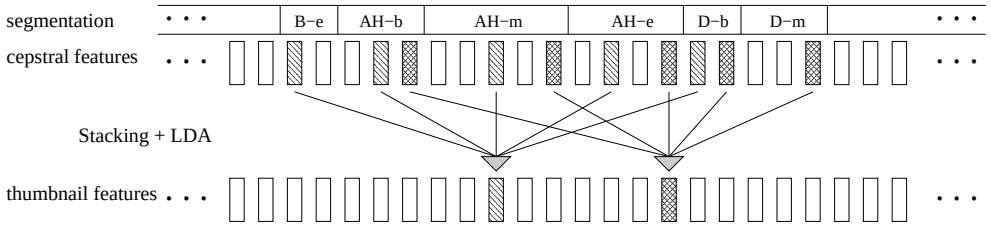


Figure 5.2: Computing thumbnail features on a frame-by-frame basis. The two examples shown are located at 50% and 100% percentage points of their respective segment. Here a thumbnail feature covers 5 subphone segments.

In this way, we can compute a thumbnail feature stream that looks exactly the same as any traditional feature stream. The number of feature vectors are guaranteed to be equal for different hypotheses. Although the observation space is still different for each hypothesis, at least we have the same number of probability terms now.

Compared to the "standard" segmental feature, which computes one feature vector per segment, our approach also achieves greater time resolution. We can now model at the segmental level without deviating too much from the frame-synchronous basis of current ASR framework.

5.2.3 Hybrid Segmental/Dynamic Features

There is a natural link between dynamic features and thumbnail features obtained by subsampling a segment.

As shown in section 6.3, dynamic features can be computed in general by stacking consecutive frames together, followed by a linear projection. If we instead choose frames from evenly spaced locations in a segment (subsampling), the set of frames is then a thumbnail sketch of the segment. The linear projection can be computed as usual, for example, by LDA.

Because of this link, we can easily derive a hybrid version from the two, by stacking frames subsampled at a segmental level, and frames immediately adjacent to the

current time point. LDA is then applied as usual. The result is a mixture between conventional dynamic features and segmental features. By varying the number of "segmental" frames vs. the number of "dynamic" frames, we can move smoothly between pure dynamic features and pure segmental features.

5.3 Experiments

5.3.1 Initial Experiments

Experiments are carried out on the Switchboard task. The test set is a 1 hour subset of the 2001 Hub5e evaluation set. A 66 hour subset of the SWB-I corpus is used for training. The front-end will be described in Section 7.1. The triphone-clustered acoustic models have 24000 distributions, sharing 6000 codebooks, with a total of 86k Gaussians. For testing, we use a 15k vocabulary and a trigram language model trained on SWB and CallHome. The baseline system has a word error rate (WER) of 34.5%. Note MLLT is not used here, so we can compare with various thumbnail models without updating MLLT each time.

Training thumbnail models is straightforward. We first generate segmentations by forced aligning reference texts on the training data using the baseline models. Then thumbnail features can be computed, from which LDA and acoustic model are estimated. For testing, the ideal would be to search through all hypotheses, each with its own features. Since this involves significant changes to the search code, we pursued two alternative methods:

- N-Best list rescoring: N-Best lists represent a reduced search space [Schwartz and Austin, 1993]. The advantage of N-Best lists is that we can accurately compute acoustic/language scores for each hypothesis. However, N-Best lists are known to be quite restrictive, compared to lattices.

- decoding with fixed thumbnail features: Since we can produce a thumbnail feature stream just like a standard front-end, we can use the baseline system to find segmentations and produce a fixed stream of thumbnail features. Then decoding can be carried out on this fixed set of features as usual.

In our initial experiments, N-Best rescoring results are worse than that of decoding on fixed thumbnail features. Subsequent experiments are therefore conducted using mostly the second approach.

As shown in Table 5.1, we mainly experimented with hybrid segmental features by adding segmental frames to the baseline dynamic features. First, we add segmental features at the senone (sub-phoneme) level. Using 11 context frames and 5 segmental frames, initial WER is 34.6%. If we recompute alignments using the thumbnail models and recompute thumbnail features, we get a small gain of 0.4%. With N-Best rescoring (N=800), the best WER we can achieve is 34.5%.

segment level	# segments	# context frames	WER	comments
-	-	11	34.5	baseline (w/o thumbnail)
senone	5	11	34.6	
			34.2	iterative feature estimation during testing
			34.5	N-Best rescoring (N=800)
phone	5	11	34.0	
			33.7	retrain after iterative feature estimation
phone	7	11	33.9	
phone	7	7	34.3	

Table 5.1: Initial Experiments with Thumbnail Features

Next, we move on to phone-level segments. Using 11 context frames and 5 frames from ± 2 phone segments, WER is 34.0%. This is done after iterative refinement of segmentation during test. If we also reestimate segmentation on the training set, WER improves to 33.7%. We also tried to vary the number of segments and the number of context frames, no further gain has been observed.

All the above experiments use only a 66 hours subset for training, no MLLT, and first pass decoding without adaptation. In the next section, we add all advanced features and present results on our RT-03 Switchboard evaluation setup, to gauge the potential improvements on the final system.

5.3.2 Experiments on RT-03 Final Stages

As will be explained in Chapter 7, the RT-03 Switchboard evaluation system utilizes several additional features, such as MLLT, feature space adaptation, MLLR, and speaker adaptive training. After incorporating MLLT and FSA-SAT in training the thumbnail models, we observed typical improvements in likelihoods on the training set.

	WER(%)
best single system (pass 15), w/o thumbnail features	24.3
best thumbnail model (MLLT, FSA-SAT, MLLR)	24.4
N-Best rescoring on 6 final hypos	24.1
system combination (final submission)	23.5

Table 5.2: Experiments on RT-03

We started testing on top of the best single system result (pass 15 in Table 7.7, which is 24.3% on the full 6 hours RT-03 test set. This set of hypotheses is used to provide initial segmentations for thumbnail features. After both FSA and MLLR, WER is 24.4% for the final thumbnail model.

As we suspect it could be suboptimal to use a fixed set of thumbnail features, we performed a "mini" N-Best rescoring experiment. Hypotheses from several final passes are collected and rescored using the thumbnail model. Six sets of hypotheses are used, with WER ranging from 24.3% to 24.7%. This way we get a 0.2% improvement, from 24.3% to 24.1%. This improvement is likely to be insignificant, especially if we are to perform the final system combination step. In the RT-03 evaluation, the final system combination (consensus network combination and ROVER) reduces WER from 24.4% to 23.5% (See Section 7.4.2 for details).

5.3.3 Analysis and Discussion

Since it is possible that the hybrid thumbnail features may degenerate into the conventional dynamic features, we examined the LDA matrix in the thumbnail models, to see whether it makes use of segmental dynamics or not. We did find large variations in the part of the LDA matrix that corresponds to segmental frames.

Thumbnail features can be interpreted as a kind of speech rate normalized feature extraction. A thumbnail feature always covers an entire "logical" segment, no matter how long or short it is. As a result, thumbnail models are more robust to changes in speaking rate. They are speech rate normalized models, similar in essence to models obtained by speaker adaptive training or VTLN. This is illustrated in Figure 5.3. During test, it is therefore important to have the correct speech rate estimation, which is implicitly represented by the initial segmentation. Otherwise, performance could be worse than models trained without normalization.

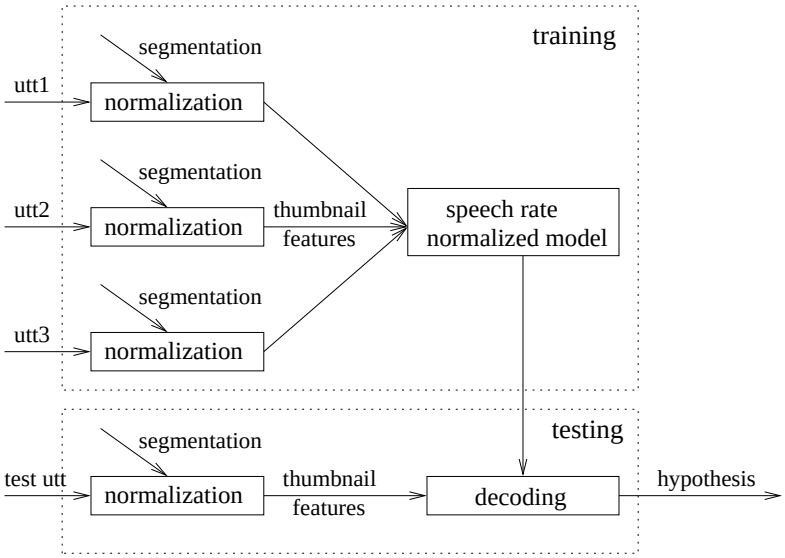


Figure 5.3: Another view of thumbnail models: speech rate normalized training

We conducted a cheating experiment to measure the impact of initial segmentation. We used the best thumbnail model in Table 5.1, which uses 5 frames from

adjacent phone segments, 11 context frames, and has a WER of 33.7%. A cheating segmentation, which is way better than the baseline segmentation, is obtained from a set of hypotheses that has a WER of 29.4%.[1] The WER is 32.5% by decoding on thumbnail features based on this segmentation. This is significantly better than 33.7%, indicating the importance of initial segmentations, but still somewhat below our expectation.

5.4 Future Work

Experiments presented so far are still preliminary. It is worth investigating why N-Best rescoring does not perform as well as decoding with fixed thumbnail features. This could be because N-Best lists are an inefficient representation of the search space. Another possible reason concerns the different observation space for different hypothesis. Although the same number of probability terms are guaranteed in our current implementation, it could still be problematic to compare hypotheses on different basis.

In the conventional ASR framework, two hypotheses are compared by

$$P(h_1|O) \quad \gtrless \quad P(h_2|O)$$

where O, the frame sequence, serves as a common ground for all comparisons. If we compute a different set of features for each hypothesis, we are comparing

$$P(h_1|O_1) \quad \gtrless \quad P(h_2|O_2)$$

The common ground is lost.

This is an important issue that affects much more than just thumbnail features. To use any hypothesis (or class) dependent feature, we will face this problem. We will need a way to normalize scores to make them comparable.

The MIT SUMMIT speech recognizer uses a segment-based framework, where feature vectors are extracted over both hypothesized phonetic segments and at their boundaries [Glass, 2003]. To ensure score compatibility, each hypothesis needs to account for the complete observation space $O = \{O_1, O_2, \cdots\}$. For example,

$$P(O|h_1) = P(O_1, O_2, \cdots|h_1) = \prod_i P(O_i|h_1)$$

$P(O_1|h_1)$ is straightforward to compute. For $i \neq 1$, $P(O_i|h_1)$ is modeled by *anti-phones*. In other words, a generic anti-phone model is used for observations(features) that are not related to the current hypothesis. As an extension, *near-miss* models are proposed as a more accurate alternative to anti-phone models.

A potentially better solution is maximum entropy models [Berger et al., 1996]. The ability to use arbitrary features is one of the hallmarks for this kind of models.

[1] It would be ideal to derive segmentation directly from reference text. However, this is not easy due to practical reasons.

There are several recent works on extending the maximum entropy principle to sequence modeling, such as maximum entropy Markov models [McCallum et al., 2000], conditional random fields [Lafferty et al., 2001]. It has also been successfully applied to many natural language processing tasks, such as shallow parsing and machine translation [Och, 2002].

Part II

Tasks

In this part we describe our system development efforts. ASR systems nowadays are very sophisticated. To build a good system, one needs to pay close attention to details in data preparation/cleanup, phone set and dictionary design, front-end design, various acoustic modeling and language modeling issues, adaptation, decoding and system combination strategies. A good amount of improvements in various DARPA evaluations, such as the Hub4 Broadcast News evaluation and the Hub5 Switchboard evaluation, actually come from such so-called "engineering" aspects. Although some improvements, for example, a better front-end, may not seem to be directly related to sloppy speech, they improve recognition performance in general and therefore are highly important in modeling sloppy speech. Besides, good systems also provide a solid baseline for evaluating new ideas.

Over the years, we have worked on three major tasks: Broadcast News (BN), Switchboard (SWB) and meeting recognition. As a brief overview, we experimented and gained a lot of experience in optimizing the front-end in the BN task, which will be described in Chapter 6. These experiences were successfully carried over to the Switchboard task, when we were preparing for the RT-03 evaluation. Both the BN and the SWB system were then applied to the meeting transcription task.

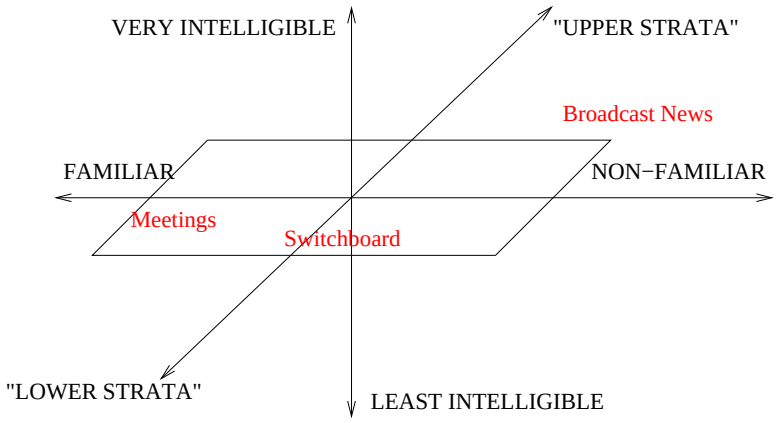


Figure 5.4: Comparing three tasks in the speech style space

In terms of speaking styles, it is interesting to compare the three tasks in the style space introduced in Section 1.1. The Broadcast News data contains both read speech and spontaneous speech. It is very formal and highly intelligible, intended for a large and unfamiliar audience. The Switchboard corpus is primarily conversational telephone speech. However, it is different from everyday conversation in that speakers did not know each other in advance, and they were asked to converse on selected topics, although the actual conversation may drift away from the assigned topic. Compared to Switchboard, meetings represent face-to-face interactions containing both verbal and non-verbal exchanges. There are a variety of different meeting types, from highly formal scenario to casual discussions. The internal group meeting data used in this thesis is mostly informal and between colleagues or classmates. Figure 5.4

shows the location of the three tasks in the style space. If we rank these tasks in terms of sloppiness, meetings are worse than Switchboard, which in turn is worse than Broadcast News. It should be emphasized that this is only a sketch and not meant to be exact.

Chapter 6

The Broadcast News Task

6.1 The Broadcast News Task

The Broadcast News task, also known as the Hub4 evaluation in the ASR community, concerns about recognition of recorded broadcasts from television networks (such as ABC, CNN and CSPAN) and radio networks (such as NPR and PRI).

The Hub4 evaluation started in 1995, as a successor to the Wall Street Journal task. While the Wall Street Journal task is in a sense artificial — professional speakers read Wall Street Journal text in quiet studios — Broadcast News is *found* speech from the real world, where one can find both native and non-native speakers, various background noise and channel conditions, as well as every possible speaking style. To facilitate the analysis of ASR system performance, the BN data is labeled according to six so-called F-conditions (Table 6.1).

F0	clean, prepared speech
F1	clean, spontaneous speech
F2	speech over telephone channels
F3	speech in the presence of background music
F4	speech under degraded acoustic conditions
F5	speech from non-native speakers
FX	all other speech

Table 6.1: F(ocus)-Conditions in Broadcast News

The BN data is recorded in 16kHz sampling rate, 16-bit, single channel format. Of all available BN data, we use the 1996 BN training corpus (LDC catalog number: LDC97S44), roughly 80 hours, for acoustic training, and the 1998 Hub4e evaluation set 1 (1.5 hours) for testing.

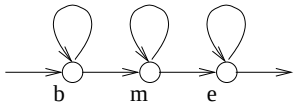
We bootstrapped the BN system from acoustic models trained on the Wall Street Journal task. The initial WER on the test set is 37%. Over time we reduced WER to 18.5%. Improvements come from a variety of sources: widening search beams,

increasing vocabulary size and language model size, introducing VTLN and CMN, using two-stage polyphone clustering, model complexity tuning, switching phone sets and dictionaries, cleanup of the dictionary and transcripts, and of course, numerous bug fixes. Among all experiments, the following two are of particular relevance to this thesis: modeling HMM topology and duration, and improving the front-end.

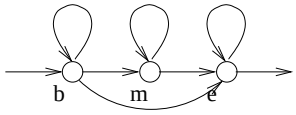
The current system uses quinphone models, with 6000 distributions sharing 2000 codebooks, and a total of 105k Gaussians. The front-end, which will be described in greater detail in Section 6.3, uses VTLN, CMN, LDA and MLLT. A trigram language model is trained with a 40k vocabulary on 130 million words broadcast news text, plus 10 times the acoustic training transcripts (800k words). The LM has 7 million bigrams and 6 million trigrams.

6.2 HMM Topology and Duration Modeling

We experimented with several different HMM topologies at an early point of our BN system development: the regular 3-state left to right topology, and the fast topology: 3-state topology with skipping arcs [Yu et al., 1999]. They are shown in Figure 6.1.



(a) regular left−to−right topology



(b) left−to−right topology with a skip arc

Figure 6.1: HMM Topologies

Training Topology	Testing Topology	WER (%)
skip	skip	34.3
skip	regular	32.7
regular	regular	32.1

Table 6.2: Topology Experiments on Broadcast News

The fast topology allows a shorter duration for each phoneme, and achieves better likelihood on the training data. However, the extra flexibility does not translate well into better WER. As in Table 6.2, using the fast topology in both training and testing is 2.2% worse than using the regular topology.

The explanation is that the more restrictive non-skipping topology enforces a certain trajectory which a phone must go through: begin, middle, end. Removing this trajectory constraint is equivalent to adding a large number of pronunciation variants to the lexicon, which causes a drastic increase in confusability. In this sense, HMM topology should be taken into account in pronunciation modeling.

Another way to enforce trajectory constraints is duration modeling. We choose to model the minimum number of frames a phoneme has to pass before transiting into the next phoneme. A context-dependent decision tree is grown to cluster all the triphone duration statistics. The minimum duration, found at the leaf level of the tree, is applied at decoding time to constrain the search. Initial experiments showed only a 0.3% absolute gain over the regular topology (32.1% $\rightarrow$ 31.8%) [Yu et al., 1999].

The same is observed on the Switchboard task. In the 1997 Switchboard system, the skip topology is also used, although only for `DX` and plosives. By completely switching back to the non-skip topology, we actually see a slight, though insignificant, improvement. Therefore, we decide to use the non-skip topology for all future experiments.

6.3 Improving the Front-End

This section details our effort in the front-end, which is an important source of improvements for the BN task.

A front-end processes raw speech signal into a sequence of feature vectors, in a form suitable for subsequent acoustic modeling. Popular front-ends include Mel-Frequency Cepstral Coefficients (MFCC) and Perceptual Linear Prediction (PLP). We will focus on optimizing the MFCC front-end here, although most of the our discussion extends to non-MFCC front-ends as well. We will first examine the traditional MFCC front-end, which can be considered as a "pipeline" consisting of various components, each designed a different purpose. Our key insights are that each of these components should not be designed in isolation, and that the ultimate goal of the front-end is to achieve better acoustic modeling. Hence, the design of any front-end component needs to be scrutinized in the context of other components, as well as the acoustic model scheme to be used. Section 6.3.2 examines and optimizes the delta and double-delta features. Section 6.3.3 looks at the overall front-end pipeline in its entirety and tries to consolidate various linear transforms, which leads to a greatly simplified front-end design. In Appendix A, we also present a novel phase space interpretation of the front-end, which explains how dynamic features are used to alleviate the frame independence assumption of HMMs.

6.3.1 The MFCC Front-End

Mel-Frequency Cepstral Coefficients (MFCC) has been a popular front-end in speech recognition for many years. It has been repeatedly shown to be quite robust for a variety of tasks. Over the years, several enhancements have been added on top of the basic MFCC design. Figure 6.2 shows a typical front-end that we use for LVCSR tasks.

Figure 6.2: A Typical MFCC Front End

Below we give a brief account of various components in Figure 6.2, with an emphasis on their connections to acoustic modeling:

1. The continuous time signal is digitized and divided into a sequence of (overlapping) frames, which will be the unit for subsequent processing. The frames have a fixed length, typically from 16 milli-seconds to 30 milli-seconds, and are evenly spaced (almost always 10 milli-seconds apart).

2. FFT is performed on each frame. The power spectrum is extracted and warped to compensate for gender/speaker differences in vocal tract lengths. This is known as Vocal Tract Length Normalization, or VTLN [Kamm et al., 1995, Lee and Rose, 1996, Eide and Gish, 1996, Zhan and Westphal, 1997]. The implementation we use is based on [Zhan and Westphal, 1997].

3. The warped spectrum is then smoothed by a series of triangular shaped filters (30 of them in our case) placed evenly along a non-linear frequency scale. Mel scale, among the most commonly used, is designed to approximate the frequency resolution of the human ear, which is more sensitive at lower frequencies.

4. Cepstral analysis is performed by applying a Discrete Cosine Transform (DCT) to the log of the filterbank output. Log is used to compress the dynamic range of the spectrum, so that the statistics are approximately Gaussian. DCT decorrelates the coefficients, so that it is more sensible to use diagonal covariance matrices later in acoustic modeling. Typically, the first 13 DCT coefficients are retained, the rest are discarded as irrelevant details.

5. Cepstral Mean Normalization (CMN) normalizes for channel variations, to make the acoustic model more robust to changes in channel conditions. CMN can be divided into two parts: cepstral mean subtraction and cepstral variance normalization.

6. Delta and double-delta features, also known as dynamic (or derivative) features, capture speech dynamics across different frames by incorporating differences between adjacent frames in feature vectors.

7. Linear Discriminant Analysis (LDA) reduces dimensionality, while retaining as much discriminative information as possible [Fukunaga, 1990]. LDA maximizes the ratio of between-class scatter and within-class scatter.

8. On top of LDA, there can be another linear transform, so that the final feature vectors fit well with the diagonal covariance assumption in acoustic modeling. This is called Maximum Likelihood Linear Transform (MLLT, [Gopinath, 1998]), which is a special case of semi-tied covariance matrices [Gales, 1998]. Recently, several extensions have been proposed to achieve more sophisticated covariance tying, by constraining the inverse covariance matrices to be a linear combination of many rank one matrices [Olsen and Gopinath, 2002], or more generally, symmetric matrices [Axelrod et al., 2002].

The term MFCC comes from the use of the Mel-scale filterbank and cepstral analysis. The basic MFCC front-end has only four stages: FFT, Mel-scale filterbank, log and DCT. This may work reasonably well for a small, controlled task, but not good enough for LVCSR. Each additional component provides some extra sizable improvement. Altogether, the difference in performance can be dramatic.

6.3.2 Optimizing the Dynamic Features

The simplest implementation of the delta and double-delta features is to subtract the preceding frame from the current frame:

$$\begin{cases} \vec{\Delta}_i &= \vec{x}_i - \vec{x}_{i-1} \\ \vec{\Delta\Delta}_i &= \vec{\Delta}_i - \vec{\Delta}_{i-1} \end{cases} \tag{6.1}$$

where $(\vec{x}_{i-2}, \vec{x}_{i-1}, \vec{x}_i, \cdots)$ is a sequence of cepstral vectors.

Furui uses the following formulae that can be regarded as a filter over the frame sequence [Furui, 1986]:

$$\begin{cases} \vec{\Delta}_i &= -3\vec{x}_{i-3} - 2\vec{x}_{i-2} - \vec{x}_{i-1} + \vec{x}_{i+1} + 2\vec{x}_{i+2} + 3\vec{x}_{i+3} \\ \vec{\Delta\Delta}_i &= -3\vec{\Delta}_{i-3} - 2\vec{\Delta}_{i-2} - \vec{\Delta}_{i-1} + \vec{\Delta}_{i+1} + 2\vec{\Delta}_{i+2} + 3\vec{\Delta}_{i+3} \end{cases} \tag{6.2}$$

This performs better than Equation 6.1 in general, and is used by many researchers. We will refer to this as the *traditional* delta and double delta feature henceforth.

If we express both in the matrix form, Equation 6.1 becomes

$$\begin{pmatrix} \vec{x}_i \\ \vec{\Delta}_i \\ \vec{\Delta\Delta}_i \end{pmatrix} = \begin{pmatrix} \vec{x}_i \\ \vec{x}_i - \vec{x}_{i-1} \\ \vec{x}_i - 2\vec{x}_{i-1} + \vec{x}_{i-2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} \vec{x}_i \\ \vec{x}_{i-1} \\ \vec{x}_{i-2} \end{pmatrix} \tag{6.3}$$

Equation 6.2 can be written as

$$\begin{pmatrix} \vec{x}_i \\ \vec{\Delta}_i \\ \vec{\Delta\Delta}_i \end{pmatrix} = \begin{pmatrix} & & & & 1 & & & \\ -3 & -2 & -1 & 0 & 1 & 2 & 3 & \\ \multicolumn{8}{c}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots} \end{pmatrix} \begin{pmatrix} \vec{x}_{i-6} \\ \vdots \\ \vec{x}_i \\ \vdots \\ \vec{x}_{i+6} \end{pmatrix} \tag{6.4}$$

The matrix form makes it easy to understand the nature of dynamic features. There are two components: stacking of several adjacent cepstral frames, followed by a linear projection. The linear projection could potentially reduce dimensionality, as is the case in Equation 6.4. In terms of information flow, the stacking operation increases the amount of information in each frame, while the linear projection reduces the amount of information per frame.

There are two main differences between Equation 6.1 and Equation 6.2: context window size (measured by the number of cepstral frames used to form the extended vector), and the exact form of the transformation.

As shown in Equation 6.3, the former has a context window of 3 frames. The latter has a window size of 13: $\vec{\Delta}_i$ takes into account 7 adjacent frames; $\vec{\Delta\Delta}_i$, when written out in terms of $\vec{x}$, uses 13 adjacent frames. The wider the context window, the more information there will be in the final feature vector that a classifier can make use of. Note the necessity to pack more information into every feature vector comes from the frame independence limitation in acoustic modeling. If an acoustic model could capture dynamics across frames, there would be no need for the dynamic features.

As for the form of the transformation, Equation 6.2 uses a seemingly more elaborate filter: $(-3, -2, -1, 0, 1, 2, 3)$, compared to the simple step function in Equation 6.1. However, both are hand-crafted, and there is no guarantee about their optimality. The key effect of the transformation is dimensionality reduction. The filter in Equation 6.2 is equivalent to a projection from a high dimensional space (13 frames $\times$ 13 cepstral coefficients = 169) into a 39-dimensional subspace. Choosing the subspace by hand is sub-optimal. A better approach, adopted here and at some other research groups [Haeb-Umbach and Ney, 1992], is to explicitly construct the extended vector by stacking adjacent cepstral vectors together, skip any hand-crafted transformation, and use LDA to choose the optimal subspace in a data-driven manner. The only decision to make here is the context window size. This has indeed led to significant improvements on the BN task.

Table 6.3 compares different front-ends on the Broadcast News task. All numbers reported are first pass decoding results. The traditional front-end uses Equation 6.2 to compute dynamic features. As discussed before, this corresponds to a nominal context window size of 13 frames. Together with power, delta-power and double-delta-power, the final feature vector has 42 dimensions. For data-driven derivation of the dynamic feature, we tried several different context window sizes. To be fair in the comparison,

Δ,ΔΔ Style	Context Window Size (# frames)	WER (%)	
		no MLLT	MLLT
traditional	13	21.6	21.2
data-driven	7	20.8	19.2
data-driven	13	20.1	19.0
data-driven	15	-	18.5

Table 6.3: Word error rates on 1998 Hub4e (Broadcast News) eval set 1. LDA is used throughout all experiments to reduce the final dimensionality to 42.

VTLN and CMN are always applied, and the final feature dimensionalities are always 42. LDA is used and re-estimated for all experiments. In the case of traditional dynamic features, LDA does not reduce dimensionality at all. It simply chooses the most discriminative dimensions within the same space. In our experience, this makes a 5% to 10% relative WER reduction compared to the case when LDA is not applied.

The following conclusions can be drawn from Table 6.3:

- Although the traditional dynamic feature has a nominal 13 frame context window, it is outperformed by data-driven projection using only a 7-frame context window. This indicates that the linear projection in Equation 6.2 makes a poor decision in choosing the subspace. LDA, on the other hand, chooses the most discriminative subspace.

  The difference is even more pronouncing after MLLT is applied. For the traditional dynamic feature, MLLT gives little gain (21.6% → 21.2%), comparing to almost 8% relative in the data-driven case (21.2% → 19.0%).

- The size of the context window also matters. By doubling the number of frames (from 7 to 15) used to derive the final feature vector, we reduced WER by 0.7% absolute. It remains unclear what the optimal context window size is. It may have some correlation with the context width in context dependent acoustic modeling. For a context independent system, we may not need to look far beyond the current frame to decide its identity. But as we go from triphone to quinphone, it makes sense to use more and more adjacent frames for accurate modeling.

  This also coincides with the TRAPs idea [Hermansky and Sharma, 1999]. TRAPs examine the temporal pattern of critical band energies over a long time span (about one second). Hermansky et al. argue that for phone classification, temporal relationships are as important as short time spectral correlations. In one extreme, they extracted feature vectors from the temporal evolution of spectral energy at a single critical band.

  In the data-driven formulation of the dynamic features, we can easily increase the window size to cover a longer time span. Both temporal and spectral pat-

terns are captured, which can be regarded as a joint time-freqency filtering (which will be illustrated later in Figure 6.5). The only concern is increased computational cost in estimating the LDA matrix. Experimentally, a window of 15 frames (about 170 milli-seconds) is usually good enough for LVCSR tasks.

6.3.3 Streamlining the Front-End

Many of the stages in Figure 6.2 are linear transformations. As a matter of fact, except for FFT[1], VTLN and log, everything else is a linear transformation. Mel-scale filtering, for example, can be considered as a matrix multiplication on FFT coefficients. Delta and double delta can be casted as a linear transform over the extended cepstral vector.

As a combination of linear transforms is also linear, we can simplify the MFCC front end. There are two potential benefits:

- Unnecessary computation can be eliminated;
- Different components in the MFCC front-end are designed with different motivations. Their interaction with each other is typically overlooked. When putting them together, the overall optimality of the front-end is not guaranteed. Here, we try to consolidate various components of the front-end to achieve better overall performance.

Combining two linear transforms per se is no problem, since a matrix multiplication is all we need. The real question is whether we can completely eliminate a linear transform from both training and decoding, without adversely affecting system performance. For example, if we eliminate the DCT step completely from Figure 6.2 and retrain acoustic models, we could get a different set of model parameters. Will it give the same word error rate? The answer turns out to be yes in most cases. Next, we present a formal analysis on how linear transforms in the feature space affect recognition performance. We first examine the effect of full (dimensionality preserving) linear transforms, then discuss the effect of dimensionality reduction.

Invariance Properties of Speech Recognizers

As a simple example, consider adding a linear transform to Figure 6.2, after the LDA step. The linear transform is diagonal, i.e. it stretches or compresses each dimension of the final feature vector independently. This gives rise to a different feature stream as input for acoustic training and decoding. One can expect the new acoustic model to be different, but decoding result to remain unchanged, since the new linear transform does not affect discrimination at all. It is, after all, a simple scaling of coordinates. In fact, this is why CMN schemes need not worry about whether to normalize the variance to 1, or 0.5, or any other constant, as long as it is consistently normalized.

[1]FFT is linear by itself, but taking the magnitude of the spectrum is not.

We try to determine for the following cases, whether recognition performance will remain the same upon adding (or removing) a linear transform in the front-end. We consider here only non-singular linear transforms and we assume that the acoustic model parameters will be reestimated.

1. A linear transform immediately before LDA

   LDA was initially introduced as a dimensional reduction technique that tries to retain as much discrimination information as possible in a reduced space.

   One criterion for choosing the LDA matrix is

   $$\arg\max_B \frac{|B\Sigma_b B^T|}{|B\Sigma_w B^T|}$$

   where Σ_b is the between-class scatter matrix, Σ_w is the within-class scatter matrix.

   Note that the LDA solution is not unique. If B is found to maximize this criterion, it is easy to verify that AB also maximize the criterion, where A is square and non-singular. In other words, LDA specifies only the optimal subspace, but not the exact set of coordinates. This is exactly why it would matter to have an additional MLLT transform on top of LDA. However, given a particular LDA implementation — such as simultaneous diagonalization — which usually finds a single solution, the uniqueness of the transform could be empirically guaranteed.

   Under this assumption, if we add an extra non-singular linear transform A before LDA, the combination of the newly found LDA matrix B' and A will be equal to the original LDA matrix B:

   $$\begin{aligned} B' &= \arg\max_B \frac{|BA\Sigma_b A^T B^T|}{|BA\Sigma_w A^T B^T|} \\ \Longrightarrow \quad B'A &= \arg\max_B \frac{|B\Sigma_b B^T|}{|B\Sigma_w B^T|} \end{aligned}$$

   This means that the final feature vectors, as well as the model parameters, will remain unchanged.

   If the uniqueness of the LDA solution is not guaranteed, we can only establish that the new subspace (after the new LDA) will be the same as the original subspace. Equivalently, the new subspace is a linear transform of the original subspace. To determine whether recognition performance will remain invariant, we need to consider the following scenario (case 2).

2. A linear transform immediately after LDA

If a random variable $X \in R^d$ is modeled by a Gaussian $N(\mu, \Sigma)$, a linear transform of X, $Y = AX$, can be modeled by $N(A\mu, A\Sigma A^T)$. It can be shown that their acoustic scores are related by:

$$\begin{aligned} p(x|\mu,\Sigma) &= |A| * p(Ax|A\mu, A\Sigma A^T) \\ \log p(x|\mu,\Sigma) &= \log|A| + \log p(Ax|A\mu, A\Sigma A^T) \end{aligned} \tag{6.5}$$

- If full-covariance matrices are used in acoustic modeling, Equation 6.5 indicates that the additional linear transform A causes a constant shift in log-likelihood. This will not affect discrimination between models. Therefore, recognition performance will remain the same, although we may need to adjust some parameters such as the search beam size and the language model weight.
- If semi-tied covariance matrices or MLLT is used, Σ is represented as two components in the acoustic model: H and Σ_d.

  $$\Sigma = H\Sigma_d H^T$$

  where Σ_d is a diagonal matrix, defined for each Gaussian, H is tied between certain Gaussians. Since

  $$X \sim N(\mu, H\Sigma_d H^T) \quad \Longrightarrow \quad AX \sim N(A\mu, AH\Sigma_d H^T A^T)$$

  the additional linear transform A can be absorbed into the non-diagonal part of the covariance matrices, or the MLLT matrix. Recognition performance will stay unchanged, as in the full-covariance case.
- If diagonal covariance matrices are used, recognition performance will stay the same if A is diagonal too.

  $$X \sim N(\mu, \sigma^2) \quad \Longrightarrow \quad aX \sim N(a\mu, (a\sigma)^2)$$

  where X is a particular feature dimension, a is a scaling factor. Any non-diagonal transformation, such as rotation or affine transformation, may result in a performance difference.

To summarize, LDA reestimation will most likely take care of any non-singular linear transform immediately before it; acoustic model reestimation will absorb any non-singular linear transform on the final features, with the exception that under the diagonal covariance assumptions, only diagonal transforms are allowed.

Dimensionality Reduction

Dimensionality reduction happens at many places in the MFCC front-end. Some are easy to recognize, such as spectrum smoothing using the Mel-scale filterbank,

truncation of the cepstrum after DCT, and LDA. Some are less obvious, such as the delta and double delta stage. As mentioned before, it is equal to two operations: stacking several adjacent cepstral vectors, followed by a linear projection. Stacking effectively increases the dimensionality of the feature vector, while linear projection reduces dimensionality.

While it makes training more feasible, dimensionality reduction can potentially lose useful information, and therefore, should be treated with great care. For example, as shown in Section 6.3.2, the linear projection in the traditional dynamic features make a poor choice in the subspace, which seriously degrades system performance. Our practice is to adopt a data-driven approach, typically LDA, to retain as much discriminative information as possible in the subspace.

Some of the dimensionality reduction stages mentioned above seem to be well motivated. Mel-scale filterbank assigns lower resolution at high frequencies to simulate the sensitivity of human ear; DCT retains only the top 13 coefficients to capture the spectral envelope. However, these intuitions may not correspond well with a data-driven criterion like LDA. As it indicates by our previous experiments in optimizing dynamic features, there could be potential improvements by streamlining these operations as well.

Experiments

In the traditional MFCC front end (Figure 6.2), there is a chain of linear transforms after the log stage. Following the analysis developed in the previous sections, we can eliminate unnecessary linear transforms in the front end. We expect no loss in recognition accuracy, and even potential improvements.

We tried two simplified front-end schemes. The first consolidates DCT, the linear projection in the delta and double-delta step, and LDA. This effectively eliminates DCT from the front-end. The second scheme goes one step further to eliminate the Mel-scale filterbank as well.

Before we begin, some special consideration is needed from CMN. CMN is typically performed on an utterance (or speaker) basis. It is therefore linear within each utterance (or speaker), but not globally linear. For this reason it can't be absorbed into the single linear transform described above. However, it can be shown that in a series of linear transforms, it doesn't matter where exactly mean subtraction takes place. Since the net effect of CMS (Cepstral Mean Subtraction) is to center feature vectors around 0, we can move the CMS step anywhere we want. The resulted feature vector will remain the same.

Variance normalization is a little different. Since it is typically applied for each dimension separately, variance normalization at a different stage produces a different set of feature vectors. However, this may not be a major problem, as shown in the following experiments, since variances are still normalized somewhat.

Figure 6.3: Simplified MFCC Front End

Eliminating DCT DCT is an integral part of the original MFCC front-end. It serves to decorrelate cepstral coefficients to make the dimensional independence assumption more valid.

In Figure 6.3, we try to eliminate the DCT step in the simplified MFCC front end. Now, we no longer need to choose the form of DCT or the form of the linear projection for dynamic features. They are implicitly chosen in a data-driven fashion by LDA. In addition, instead of reducing dimensionality at several stages, LDA now handles all the dimensionality reduction. Both could lead to improved performance.

	Avg	F0	F1	F2	F3	F4	FX
with DCT	21.6	10.2	21.8	31.2	34.3	16.5	31.7
without DCT	21.6	10.2	20.7	30.8	36.6	16.3	32.6

Table 6.4: WER(%) on Hub4e98 Set1, comparing two front-ends with or without DCT. Both use data-driven dynamic features with a 7-frame context window and no MLLT.

As shown in Table 6.4, the front-end without DCT achieves the same performance as the one that uses DCT. These two front-ends are still different, as can be seen from the WER breakdown into different focus-conditions. However, the overall difference is not significant. Two conclusions can be drawn. First, DCT, as well as the truncation of the cepstrum afterwards, is well motivated, since the data-driven transform couldn't find a better alternative. On the other hand, DCT is dispensable. One can leave it out without hurting system performance.

Eliminating Mel-scale filterbank Going one step further, since the Mel-scale filterbank is just another linear transform that reduces dimensionality, one would naturally question its optimality too. After all, it's motivated perceptually, and is not necessarily consistent with the overall statistical framework.

Due to the nonlinearity of the log step, it's not straightforward how to optimize this stage directly. Instead, we tried to leave out this stage completely, since the function it serves, namely to smooth the spectra and to reduce dimensionality, can be well captured in the linear transform after the log step. This leads to an even simpler and highly unconventional front end, which we call the emphLLT[2] front end

[2]LLT is mainly an internal acronym, reflecting the fact that the front-end has two parts: log and a linear transform.

(Figure 6.4). Alternatively, one can think of the LLT front end as the result of switching the order of log and the Mel-scale filterbank, after which the Mel-scale filterbank can be integrated with the other linear transforms.

Figure 6.4: The LLT Front End

Front-End	Context Window Size (# frames)	WER (%)	
		w/o MLLT	w/ MLLT
MFCC	7	20.8	19.2
LLT	7	20.4	19.0

Table 6.5: Effect of the LLT Front-End on the BN task. Data-driven dynamic features are used in both cases.

Table 6.5 shows the performance of the LLT front-end on the BN task. LLT gives roughly the same performance as the MFCC front-end, if not better. This indicates that, first, the Mel-scale filterbank is well designed; second, it is possible to leave out the Mel-scale filterbank and let LDA decide how to smooth the spectrum. The LLT transform has significantly more parameters, though. To project 7 frames of FFT coefficients (129 per frame) into a 42 dimensional feature space, the size of the LLT matrix is 903 by 42.

In Figure 6.5, the LLT matrix in the BN system is visualized. We remind the reader that the LLT matrix here is a combination of LDA and MLLT; interpretation of these plots should be drawn with this in mind. Since each row of the 42×903 matrix computes one final feature dimension from 7 adjacent frames of 129 log-FFT coefficients, it can be considered a joint time-frequency filter. Figure 6.5 shows the first 4 dimensions of the LLT transform in the BN system. For clarity, only coefficients for the first (lowest) 30 FFT indices are shown. The higher frequency part is relatively flat comparing to the lower frequency region. It is interesting that the third dimension is mostly a frequency-domain filter; little is happening along the time axis. But for most other dimensions, the filter is undoubtedly operating on the joint time-frequency domain. It is also worth noting that most of the activities happen at the low frequency region. This supports the traditional wisdom behind the Mel-scale filter, which has higher resolution on low frequency regions.

We speculate the following reasons why LLT does not outperform MFCC:

- Cepstral mean and variance are 13 dimensional for MFCC, while they are 129-dimensional for LLT. Their estimation could be more susceptible to the data sparseness problem.

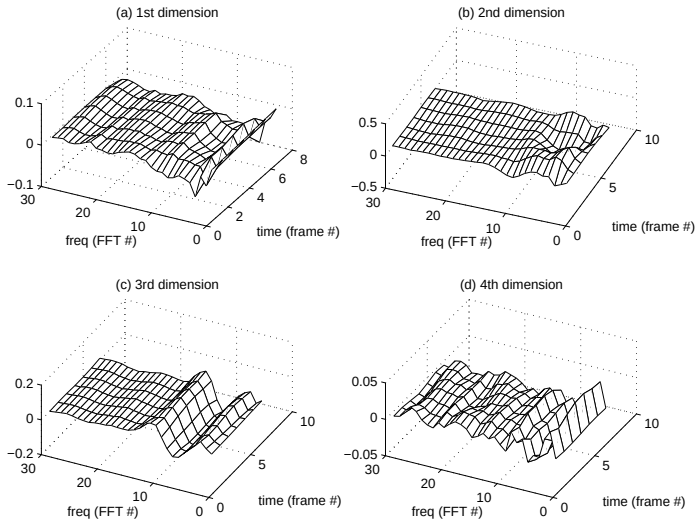


Figure 6.5: Visualizing the LLT projection as Joint Time-Frequency Filtering

- The LLT matrix has far more parameters than the LDA matrix in the MFCC front-end. It could overfit on the training data. Smoothing techniques may be needed.

Although LLT is conceptually simpler than MFCC, it is computationally more expensive. LLT delays dimensionality reduction until the very last step, at the cost of using a big LDA matrix. In the MFCC front-end, Mel-scale filterbank and cepstral truncating cut dimensionalities at an earlier stage, which keep the cost low.

6.4 Conclusion

Overall, we achieved a 50% relative improvement (37% to 18.5%) on the BN task, of which 14% comes from front-end optimization. An important side product of this process is a better understanding of the modern MFCC front-end:

- We showed that dynamic features should be optimized in a data-driven fashion;
- We empirically verified the validity of DCT and the Mel-scale filterbank;
- We proposed a conceptually simplified front-end which leaves out both DCT and the Mel-scale filterbank, yet achieves a comparable performance.

Chapter 7

The Switchboard Task

This chapter describes experiments on the RT-03 (Rich Transcription) Switchboard experiments, where we integrate techniques such as the optimized front-end, single pronunciation dictionary and enhanced tree clustering into the final system.

7.1 The Switchboard Task

The Switchboard corpus is a large collection of conversational telephone speech [Godfrey et al., 1992]. The original SWB corpus, SWB-1 Release 2, contains roughly 2500 conversations from 500 U.S. speakers. The speakers are previously unknown to each other, and are asked to converse on a certain topic over the phone. The data is recorded in two channels using a 4-wire setup, 8kHz sampling rate, and 8-bit u-law encoding. An echo cancellation algorithm is applied for all the data.

	#conv collected	#conv transcribed	#sides used	#hours used
SWB-1 Release 2	2400	all	4876	317
CallHome English	120	all	200	17
SWB-2 Phase I	3638	0	0	0
SWB-2 Phase II	4472	473	944	33
SWB-2 Phase III	2728	0	0	0
SWB-Cellular	1309	728	1456	63
Total	-	-	7476	430

Table 7.1: Composition of the SWB Training Set. *conv* is short for conversation. Each conversation has two sides.

The collection and transcription of the SWB data is a long and still ongoing effort. Switchboard-1 is the "original" SWB corpus. It has been released two times. There are certain differences between the two. Release 2 is the most commonly used, with over 240 hours of speech. There are two major versions of transcripts

for SWB-1: the original LDC (Linguistic Data Consortium) transcript and the ISIP transcript. The ISIP transcript is an effort, conducted by ISIP (Institute for Signal and Information Processing) at Mississippi State University, to fix certain problems in the original LDC release. We experimented with both versions of transcript and didn't find major differences in performance. SWB-2 is collected somewhat later, and largely untranscribed. SWB-Cellular aims at capturing cellular data, but it contains a portion of landline calls as well. Transcription of the SWB-2 data and a large chunk of the cellular data (the CTRAN data) was completed recently using a quick transcription approach, as opposed to the traditional careful transcription approach. The objective is to produce large volumes of transcribed speech at a reasonable quality and relatively low budget.

From 1996 to 2001, NIST held an (almost) annual evaluation on the Switchboard task, known as the Hub5 evaluation. Starting in 2002, the Rich Transcription (RT) project continues to evaluate system performance on conversational telephone speech (CTS), with an emphasis towards using more the so-called Fisher data, which is collected in a way that closely resembles Switchboard.

Table 7.1 gives a dissection of the entire SWB training set at the time of the RT-03 evaluation. The entire training set we have contains 7476 sides, roughly 430 hours of speech. The RT-03 evaluation set is 6-hours long, half from Switchboard and half from the Fisher data. For most of our SWB experiments, we use a 66 hour training subset with 1088 sides, all from SWB-1 Release 2, and a 1-hour development set extracted from the 2001 Hub5e evaluation data.

7.2 The Switchboard Baseline System

Our Switchboard baseline system is the 1997 ISL Hub5e evaluation system [Finke et al., 1997]. We give here a brief overview of this system.

For pre-processing, 13 MFCC coefficients, together with power, and their first and second derivatives form a 42-dimensional vector. LDA is used to map it into a final 32-dimensional space. Maximum likelihood vocal tract length normalization is performed to build a gender-independent acoustic model.

For acoustic modeling, continuous mixture density models are used, together with two-stage quinphone clustering. There is a total of 24000 distributions, defined on top of 6000 codebooks, with 16 Gaussians in each codebook. 161 hours of Switchboard-I data and 17 hours of CallHome data are used for acoustic training.

The phone set is composed of 44 regular phones, 1 silence, 6 noise phones, 4 interjections (to model filled pauses such as `uh`, `oh`, `um`, `uh-huh`, `mm-hm`), and a mumble phone. The vocabulary size is 15k, including the most frequent words from the Switchboard and CallHome text. The baseline dictionary is expanded by applying a number of hand-chosen rules. The final dictionary has about 30k pronunciation entries (2 variants per word on average), after pruning away unlikely candidates.

For language modeling, the best result comes from interpolating a plain trigram Switchboard LM, a class-based Switchboard LM, and a 4-gram Broadcast News LM.

During decoding, VTLN factors are first estimated and refined, followed by iterative MLLR estimation and adapted decoding.

This system achieved a top ranking in the 1997 Hub5e evaluation. With exactly the same setup (and multi-pass decoding), the WER is 35.1% on the 1 hour development set defined above. In the following section, we describe our efforts in improving the system for the RT-03 evaluation. Our final RT-03 evaluation system is quite sophisticated, which uses multiple sets of acoustic models and language models. Our RT-03 evaluation is a joint effort by members of the ISL at both University of Karlsruhe and Carnegie Mellon University [Soltau et al., 2004]. Most experiments described here are conducted by the author, unless indicated otherwise.

7.3 Training Experiments

7.3.1 Front-End

Based on the experiments on the BN task, we performed a series of front-end optimizations for the SWB system. Results are summarized in Table 7.2. The baseline system, the 1997 evaluation system, is trained on 180 hours of data, while the others are trained on a 66 hours subset. The baseline front-end uses the traditional delta and double delta setup, similar to the original BN front-end. The baseline also performs cepstral mean subtraction per conversation side, but no variance normalization, nor MLLT.

We gained 1% absolute, from 39.8% to 38.9%, by switching the delta and double-delta feature to data-driven projection, with a context window of 11 frames. If we take into account the difference in the size of the training set, the improvement is actually larger. The difference in WER between the 180 hours training data and the 66 hours subset is estimated to be 1% to 2% absolute.

System	WER(%)
baseline	39.8
data-driven Δ,$\Delta\Delta$	38.9
+ plain CVN	39.7
+ speech-based CMN	37.8
+ MLLT	35.6

Table 7.2: Experiments on Switchboard. Context window size is 11 frames except for the baseline.

In the first two experiments, only cepstral mean subtraction is used. We then added cepstral variance normalization (CVN). While channel distortion causes a shift in cepstral mean, additive noise has two major effects on the distribution of MFCC

parameters, reduced variance and shift in the means [Jain, 2001]. CMS therefore compensates for both channel distortion and part of the noise effect, while CVN tries to restore the reduced variance. However, as many have noted, variance compensation is quite sensitive to the amount of silence in a utterance [Jain, 2001, Woodland et al., 2001]. In case of mismatched training / testing scenario, this can cause significant degradation in WER. In our Switchboard experiments, we have verified that a straightforward CVN can indeed hurt performance (38.9% to 39.7%). As suggested in [Westphal, 1997], speech and silence regions need different treatment in the joint compensation situation. A simple solution is Speech-based Cepstral Mean Normalization (SCMN). The idea is to use a power based speech detector to mark speech region in every utterance, and estimate cepstral mean and variance only on the speech region. SCMN helps about 1% absolute on the Switchboard task. MLLT contributes another 2.2% extra improvement.

Front-End	Context Window Size (# frames)	WER (%)	
		w/o MLLT	w/ MLLT
MFCC	11	37.8	35.6
LLT	7	37.9	35.5

Table 7.3: Effects of the LLT Front-End on SWB. Data-driven dynamic features are used in both cases.

We also experimented with the LLT front-end as illustrated in Figure 6.4. Results are shown in Table 7.3. Even with a smaller context window size, the performance is comparable to that of the MFCC front-end. For computational efficiency reasons, we choose the MFCC front-end for all subsequent experiments.

7.3.2 Acoustic Modeling

Most of the acoustic models use VTLN, MLLT, and FSA-SAT. During training, we iteratively estimate VTLN parameters and the acoustic model. VTLN parameters typically converge after a couple of iterations. Then we estimate MLLT and add it to the front-end.

FSA-SAT is similar in essence to constrained model space SAT [Bacchiani, 2001]. A linear transform is computed for each conversation side during both training and decoding, and acoustic models are estimated in the normalized feature space. This improves recognition accuracy with only a fraction of the computational cost of model space SAT.

As shown in Table 7.4, FSA-SAT reduces WER by 1.3% absolute. The gain is smaller when MLLR is used, but still in the 1% range.

Feature space adaptation can be used to adapt a non-SAT model as well. The gain is typically much smaller comparing to MLLR, but combining the two, FSA gives an additional $\sim 0.3\%$ improvement on top of MLLR.

Setup	WER(%)
non FSA-SAT model	31.6
FSA-SAT model	30.3

Table 7.4: The Effect of FSA-SAT (w/o MLLR) on the full training set

We also get improvements from single pronunciation dictionary and enhanced tree clustering, as described in Section 3.2.3.

7.3.3 Language Modeling

We use an interpolated language model on a 40k vocabulary[1]. The baseline LM is

LM	WER(%)
3gram SWB	31.4
+ 5gram class SWB	31.0
+ 4gram BN	30.3

Table 7.5: Language Modeling

a trigram model, trained on the Switchboard corpus. It is interpolated first with a 5-gram class-based Switchboard LM, and then a 4-gram LM trained on the Broadcast News corpus. The overall improvement from the 3-fold interpolation is $\sim 1\%$.

7.4 RT-03 Switchboard Decoding Experiments

7.4.1 Model Profiles

	sys-id	description
AM	base	non-SAT model, quinphone, 168k Gaussians
	SAT	FSA-SAT model, quinphone, 168k Gaussians
	MMIE	MMIE, FSA-SAT, septaphone, 288k Gaussians
LM	3g small	3gram SWB LM on 15k vocabulary
	3g big	3gram SWB LM on 40k vocabulary
	full	3-fold interpolated SWB LM

Table 7.6: Model Profiles

The final models used in decoding are listed in Table 7.6. All acoustic models are trained on roughly 430 hours of the SWB training data. The MMIE models are

[1]The LM experiments are performed by Christian Fügen.

estimated using maximum mutual information criterion, while the other two uses the conventional maximum likelihood criterion. The MMIE models are also trained with a different lexicon that has an average of 2 pronunciations per word, while the other two uses single pronunciation lexicon described inSection 3.2.3.[2]

A scaled-down trigram LM is used in early decoding passes. It uses a smaller vocabulary and, therefore, is much faster to load.

7.4.2 Decoding Experiments

Table 7.7 lists the decoding steps on the RT-03 Switchboard evaluation set.

step #	decode	AM	LM	adapt	WER	comments
1	x	base	3g small			warp=1.0
2	x	base	3g small			reestimate warp
3		base	3g small	MLLR		reestimate warp
4	x	SAT	3g small	FSA	32.7	reestimate warp
5		base	3g small	MLLR	30	
6		SAT	3g small	both	29.4	
7	x	SAT	full	both	27.6	
8		base	full	MLLR	27.5	
9		SAT	3g big	both	27.7	
10		SAT	full	both	26.4	
11		base	full	MLLR	26.5	
12	x	MMIE	full	both	24.7	
13	x	base	full	both	24.9	
14		SAT	full	both	25	adapt on hypo13
14b		SAT	full	both	24.7	adapt on hypo12
15	x 8ms	MMIE	full	both	24.3	adapt on hypo14
16		base	full	both	24.7	
16b		SAT	full	both	24.4	
16c	8ms	SAT	full	both	24.4	

Table 7.7: Decoding Steps on RT-03 Switchboard Eval Set. Decoding is performed only on steps marked with "x". Acoustic lattice rescoring is used for the rest. 8ms means frame shift is changed to 8 milli-seconds (125 frames per second). The default is 10ms. Adaptation mode "both" means adapting with both FSA and MLLR.

The first 4 steps are used to establish reasonable VTLN warping factors. To save time, only the first minute from each conversation side (5 minutes long on average) is decoded in the first 3 steps, since we only need 30 seconds of voiced speech to reliably

[2]The MMIE models are trained by Hagen Soltau, Florian Metze and Christian Fügen. Details about the MMIE training can be found in [Soltau et al., 2004].

estimate a warping factor. This also prevents us from computing WERs for these steps.

After establishing VTLN parameters, we progressively apply more sophisticated acoustic models and language models. The organization of these steps is based on decoding experiments on the RT-03 dry-run set. Major improvements come from SAT and MMIE models, larger vocabulary, LM interpolation and system combination. In particular, the following items are worth mentioning:

Cross-adaptation By adapting one system using hypotheses produced by another system, we can achieve a certain degree of system combination [Peskin et al., 1999]. As is evident from Table 7.7, cross-adaptation is used extensively, where we switch either the acoustic model, the language model, or both. Step 9, for example, deliberately switches back to a simpler LM get some cross-adaptation effect. We observed that when WER seems to get stuck, we can drive it further down by cross-adaptation using hypotheses in the same WER range.

Different frame rates Changing frame shift to 8ms is another way to generate some "jittering" effect. Feature vectors are computed as usual, only that more frames are generated.

Decoding vs. lattice rescoring We don't carry out full decoding every time. Acoustic lattice rescoring is an effective technique we use to approximate full decoding. It uses information in a lattice to greatly cut down the search space. Typically we get a speedup of at least 10 fold and still very accurate WER estimation.

Speed vs. accuracy It is possible to reduce the number of steps. The particular decoding strategy in Table 7.7 is what we found to be optimal on the dry-run data. Some steps can be eliminated with only a marginal loss on accuracy.

System combination The numbers shown here are roughly comparable to the results in the official ISL submission to the RT-03 evaluation, except for the omission of the final step: confusion network combination [Mangu et al., 1999] and ROVER [Fiscus, 1997]. That step reduces WER from 24.4% to 23.5%.

Overall, our RT-03 Switchboard system reduces WER by 38% relative (35.1% to 23.5%) from the baseline Switchboard system we started a year ago [Soltau et al., 2004].

Chapter 8

The Meeting Transcription Task

8.1 The Meeting Transcription Task

Meeting transcription is a new task that attracts more and more interest. Over the years a number of internal group meetings have been recorded here at the Interactive Systems Laboratories at Carnegie Mellon University [Burger et al., 2002]. To minimize interference with normal styles of speech, we used clip-on lapel microphones rather than close-talking microphones. While lapel microphones are not as intrusive as close-talking microphones, they cause degraded sound quality. A variety of noises, such as crosstalk, laughter, electric humming and paper scratching noise are picked up during recording. For simultaneous recording of multiple speakers, a 8-channel sound card is used. This simplifies the recording setup and also eliminates the need for synchronizing multiple channels.

Meeting transcription is more challenging than Switchboard in the following aspects.

- Meetings contain a significant amount of crosstalk, where people speak simultaneously. Recognition is virtually impossible on regions of severe crosstalk. While crosstalk is also present in telephone conversations, it is quite rare and of a mild nature.
- Compared to Switchboard, meetings represent multi-party face to face interactions between familiar parties. The sloppy speech style issue only becomes more severe.
- Meeting data is very noisy, partly due to the use of lapel microphones, as described before.

A total of 14 meetings are used for our experiments. All of them are internal group meetings, where people discuss various projects and research issues. Each meeting lasts about 1 hour, with an average of 5-6 participants. The test set consists of randomly selected segments from six meetings (roughly 1 hour, 11,000 words).

8.2 The Effect of Speaking Styles

As mentioned in Chapter 1, we conducted an experiment similar to the SRI Multi-Register experiment [Weintraub et al., 1996] to assess the effect of sloppy speaking styles.

For one of the meetings, we asked three of the meeting participants to come back to read the transcript of the meeting. They were recorded using the same lapel microphone in the same meeting room. They were first asked to read the transcript in a clear voice, which is denoted as "read" speech in Table 8.1. Then each of them were asked to read the transcript, but this time "act out" his/her part in the real meeting, in other words, to simulate the spontaneous speech style. This is denoted as "acted".

condition	read	acted	spontaneous
speaker1	36.8	36.7	46.1
speaker2	35.2	61.0	63.3
speaker3	37.0	61.3	73.6
Overall	36.7	48.2	54.8

Table 8.1: WER(%) with different speaking styles

Recognition was carried out using the Broadcast News system on all three conditions. The overall WER increases from 36.7% for read speech, to 48.2% for acted speech, and finally to 54.8% for the real meeting. This clearly shows that the sloppy speech style is a major concern for meetings.

8.3 Meeting Recognition Experiments

8.3.1 Early Experiments

Since we did not have enough data to develop a meeting specific system, we experimented with various other systems in the beginning, including the ESST (English Spontaneous Scheduling Task) system, the WSJ system and the 1997 Switchboard system. [Yu et al., 1998, 1999, 2000, Waibel et al., 2001].

Different system matches the meeting task in different aspects:

- The ESST system is an in-house system developed for an English spontaneous scheduling task [Waibel et al., 2000]. It has 16kHz sampling rate and is trained on 26.5 hours of spontaneous dialogs in a travel reservation domain.

- The WSJ system is trained on 83 hours of wideband read speech for the Wall Street Journal task.

- The Switchboard system is the 1997 ISL Switchboard evaluation system. It matches the meeting task on speaking styles, but not on bandwidth. Before decoding, we need to first downsample the meeting data from 16kHz to 8kHz, under the risk of losing information in the higher frequency regions.

Table 8.2 shows recognition results using various existing systems on the meeting task[1]. Unsupervised MLLR adaptation was performed, leading to significant reduction in WER. Overall, the SWB system is significantly better, which is likely due to the match in speaking styles. While the ESST system is also spontaneous, it has a fairly limited amount of training data and is highly specialized for the travel domain.

	bandwidth (Hz)	speaking style	adaptation iterations		
			0	1	2
ESST	16k	spontaneous	67.4	57.5	55.2
WSJ	16k	read	54.8	49.6	49.9
SWB	8k	conversational	47.0	42.3	41.6

Table 8.2: WER(%) of various systems on meetings

8.3.2 Experiments with the BN System

At a later stage, we developed the BN system, which achieves a first pass WER of 18.5% on the 1998 hub4e eval set 1, as described in Chapter 6. The BN system is an attractive choice for meetings for the following reasons.

- It matches the meeting task in terms of bandwidth (16kHz wideband).
- The availability of large amounts of training data and the presence of various acoustic conditions make the models more robust acoustically. Notice the ESST system, the WSJ system and the SWB system are all trained on fairly clean data.
- In terms of speaking styles, the BN training data also contains a fair amount of spontaneous speech.

As shown in Table 8.3, the ESST system has a WER of 54.1% on this test set. The BN baseline system achieves a significantly lower WER of 44.2%. After acquiring more meeting data, we tried to adapt the BN system to the meeting domain. Maximum a-posteriori (MAP) adaptation of the BN system on 10 hours of meeting data improves the WER to 40.4%. By further interpolating the BN LM with a meeting specific LM trained on 14 meetings, we reduce WER to 38.7%.

[1]These are early experiments performed on a 1 hour long meeting.

	WER (%)
Baseline ESST system	54.1
Baseline BN system	44.2
+ acoustic MAP Adaptation (10h meeting data)	40.4
+ language model interpolation (14 meetings)	38.7

Table 8.3: Adapting the BN system to meetings

8.3.3 Experiments with the Switchboard System

More recently, encouraged by the results on the RT-03 Switchboard task, we applied the improved SWB system on the meeting transcription task.

pass #	decode	AM	LM	adapt	WER	comments
1	x	base	3g small			warp=1.0
2	x	base	3g small		41.2	reestimate warp
3		SAT	3g small	FSA	38.7	
4		base	3g small	both	38.1	
5	x	base	4g BN	MLLR	34.6	
6		SAT	4g BN	both	33.5	
7		SAT	full	both	32.6	

Table 8.4: Decoding the meeting data using the Switchboard system. "4g BN" denotes the 4-gram BN component of the full SWB LM.

The new Switchboard system uses multiple sets of acoustic models and language models, as described in Section 7.4. Compared to the RT-03 SWB experiments in Table 7.7, a simpler decoding strategy is taken (Table 8.4). The first two passes are to establish reasonable warping factors. With better acoustic models and language models, WER gradually drops to the final 32.6%. A more elaborate decoding setup may yield even lower WERs. Comparing to our previous best results, we achieved a 15% relative reduction (from 38.7% to 32.6%) in WER, without using any meeting specific data or tuning. The reason, we suspect, is that the Switchboard system better matches the meeting task in terms of the sloppy speaking style, which outweighs a better matching in bandwidth and acoustic conditions as the BN system does.

The overall improvement since the onset of the meeting transcript project is 40% relative (54.1% to 32.6%).

Chapter 9

Conclusions

9.1 Summary

Sloppy speech is a challenging problem for the current generation of ASR systems. This thesis explored several fronts to improve the modeling of sloppy speech.

One important difference of sloppy speech from read speech is the deviation from standard pronunciations. Most previous work has focused on explicit pronunciation modeling. After examining the lexicon design problem and the overall pronunciation modeling strategy, we explored two implicit pronunciation modeling methods: flexible parameter tying and Gaussian transition modeling. Together with a single pronunciation dictionary, enhanced tree clustering — a flexible tying method — has led to significant improvements (5% relative) on the Switchboard task.

Sloppy speech also exacerbates the known weaknesses of the HMM framework, such as the frame independence assumption and/or the "beads-on-a-string" modeling approach. Another front of the thesis is to alleviate these problems. Gaussian transition modeling introduces explicit dependencies between Gaussians in successive states. Thumbnail features are essentially segmental level features that capture higher level events. We haven't been able to achieve significant improvements with Gaussian transition models. Thumbnail features improve performance in initial passes, but the improvement becomes marginal when combined with advanced acoustic modeling and adaptation techniques.

One may argue that all three methods are not really specific to sloppy speech. They can be applied as general speech recognition techniques to other types of speech as well. In a sense, this stems from the fact that the boundary between sloppy speech and other types of speech is not clearly drawn, especially since styles can often change within the same conversation, as discussed in Section 1.1. It is therefore difficult to imagine a technique that works only on sloppy speech, but not on non-sloppy speech. From a historic point of view, sloppy speech is largely a "found" problem — people didn't even realize that sloppy speech was an issue in the earlier days. If ASR technologies were more accurate, we may not even need to worry about

the sloppy speech issue. Hence, it is possible that advances in generic modeling techniques can make the sloppy speech problem largely obsolete. On the other hand, all three approaches are designed with sloppy speech in mind. Their utility might be greatly diminished if applied to read speech. For example, the author believes that single tree clustering is necessary only when there is reason for cross-phone/substate parameter sharing, such as increased pronunciation reduction in sloppy speech. While this technique was proposed quite early [Takami and Sagayama, 1992, Ostendorf and Singer, 1997, Paul, 1997], there was little interest at that time in applying it to read speech.

In addition to exploring new ideas, we also focus on empirically improving system performances. Overall, we have achieved significant word error rate reduction on three tasks: 50% relative on Broadcast News, 38% relative on Switchboard (which is a team effort) and 40% relative on meetings.

Although the three tasks differ in many aspects and it may not be appropriate to compare WERs between them, it is still interesting that there seems to exist a direct relationship between the amount/degree of sloppiness and the recognition performance: our best WERs for BN, SWB, and meetings are 18.5%[1], 23.5% and 32.6%, respectively. The degree of sloppiness seems to be a good indicator of recognition performance. On the other hand, this suggests that sloppy speech is still a major problem for automatic speech recognition.

9.2 Future Work

Despite the progress, the sloppy speech problem is far from being solved. We think there is ample room for research in the following areas:

Richer acoustic models beyond HMMs There have been many proposals to go beyond the HMM framework. One would like to have, for example, the ability to use higher level features, such as prosodic features or segmental features. Maximum entropy model is one of the most promising proposals. It is designed to take advantage of arbitrary features, without requiring these features to be independent. When used in a conditional setting, it is a discriminative model by design. It has also been extended to sequence modeling, such as maximum entropy Markov models [McCallum et al., 2000] and conditional random fields [Lafferty et al., 2001]. There has been some recent work in using them for acoustic modeling as well [Likhodedov and Gao, 2002], [Macherey and Ney, 2003].

Language modeling While it has been very hard to improve upon the plain trigram language model, most researchers believe the full potential of language modeling

[1]We haven't actively worked on the BN task since 2001. This number should be significantly lower if we incorporate the latest technology improvements.

is yet to be unleashed. The challenge could be either to improve language models per se, or to find a better integration with acoustic models.

Prosodic modeling We haven't really tapped into prosody features in this thesis. However, things like stress and intonation are known to be very important in human communication. In addition to reliably tracking prosodic features, a major difficulty in prosodic modeling is also how to integrate with acoustic models and language models.

Appendix A

Phase Space Interpretation of Dynamic Features

Typically, a front-end is designed with the following objectives in mind: dimensionality reduction, for removing irrelevant details from subsequent modeling; discrimination, to keep as much class-specific information as possible; data transformation, to better fit any particular acoustic modeling scheme or assumptions, such as Gaussian mixture models with diagonal covariance matrices

In this section, we present a different view of the front-end, which is to recover the speech generation process. If the goal of ASR is to decipher words spoken, it would be desirable to first recover the physical process that produces the acoustic signal. If one could directly measure articulatory movements in a non-intrusive way, ASR could be a lot more successful nowadays. At least, channel distortion and additive noise would not be an issue. Since articulatory measurements are not readily available, the challenge is to recover information about the production process directly from speech signal. In this section, we first introduce the concept of phase space as a way to represent a dynamic process; then discuss the general idea of phase space reconstruction; and show how the current dynamic features are, in a way, a phase space representation.

A.1 Phase Space

In dynamical system research, the dynamics of a physical system can be described mathematically in a *phase space* or a *state space*. Each dimension of the space represents an independent (state) variable of the system, such as position or velocity. Each point in the phase space corresponds to a unique state of the system. The evolution of a system over time produces a *phase portrait* in the phase space. Much can be learned about the system dynamics from its phase portrait.

Common in physics text, a simple mechanical system can be described in a phase space of two dimensions: position versus velocity. A complex system with many

degrees of freedom needs a high dimensional phase space.

An example is shown in Figure A.1, where articulatory movements are measured while a person is producing the syllable /ba/ repeatedly [Kelso et al., 1985]. The left panel shows the traditional time domain measurements of jaw and lower lip movements; the right panel shows the corresponding phase portraits for the two articulators, plotted in a plane of position vs. instantaneous velocity.

Certain aspects about speech production become clear in the phase portraits. The most visible is the repetitive syllable pattern. Each circle represents an instance of /ba/, where the half denoted as `CLOSED` corresponds to /b/, `OPEN` for /a/. Inter-syllable events, such as stress, can be seen as alternating patterns of larger and smaller circles. It is also clear that the motion of the articulators is less variable during the production of the consonant (`CLOSED`) than of the vowel (`OPEN`). In addition, inter-articulator timing (articulatory synchrony/asynchrony) can be studied if we plot a phase space that covers multiple articulators (not shown here).

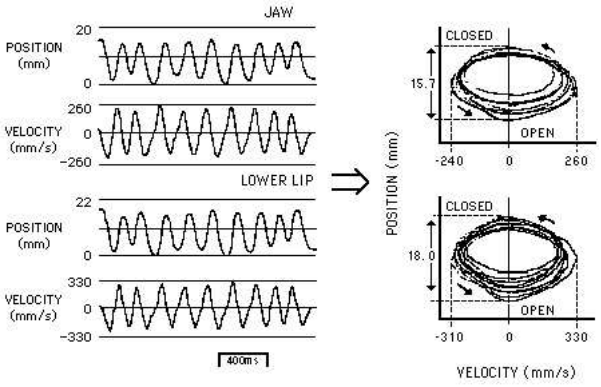


Figure A.1: Phase portraits of two articulators during production of reiterant /ba/. (From Kelso et al. 1985, ©1985 Acoustical Society of America)

The phase space of the speech production process is roughly the configuration of the human vocal tract, which, in turn, depends on the position of various speech articulators, such as tongue, lips, jaw, velum, and larynx. It is the behavior of the articulators over time that produces continually varying acoustics. A recurrent belief among speech researchers is that what the listener extracts from the speech signal might be information about the speech production process itself [Rubin and Vatikiotis-Bateson, 1998].

If measurements of various articulators could be made easily and accurately, it would be an inherently superior representation than the one based on acoustics. It gives a more direct access to the information source, and besides, there is less contamination by noise or channel distortion. Generally, however, only the speech signal is available to an ASR system. Therefore, it would be desirable to reconstruct the phase space from acoustics.

A.2 Phase Space Reconstruction using Time-Delayed Embedding

To study the dynamics of a system, all we need is the phase portrait. But in many cases, the system is not fully observable. We may only get a scalar measurement one at a time, denoted by $\{s_n\}$. Vectors in a new space, the embedding space, are formed from time-delayed values of the scalar measurements:

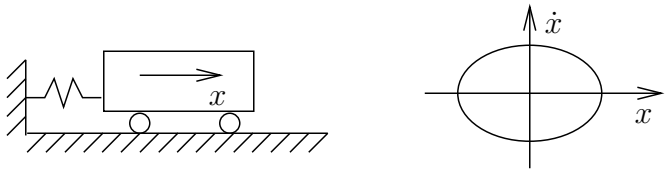
$$\vec{\boldsymbol{s}}_n = (s_{n-(m-1)\tau}, s_{n-(m-2)\tau}, \cdots, s_n)$$

The number of samples m is called the *embedding dimension*, the time τ is called *delay* or *lag*. The celebrated reconstruction theorem by Takens states that under fairly general assumptions, time-delayed embedding $\{\vec{\boldsymbol{s}}_n\}$ provides a one-to-one image of the original phase portrait, provided m is large enough [Takens, 1981].

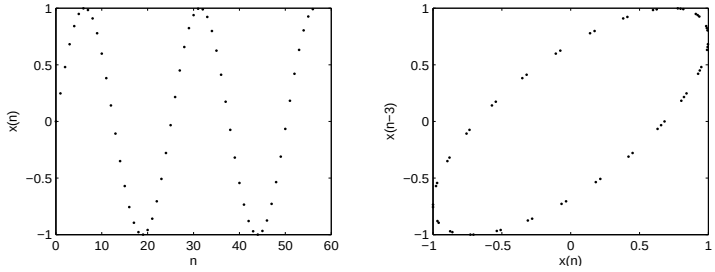
Time-delayed embedding is a fundamental tool for studying the chaotic behavior of nonlinear systems. For a detailed discussion, as well as how to choose the right value for m and τ, readers are referred to [Kantz and Schreiber, 1997].

A.2.1 A Linear Oscillator Example

For simplicity, we use a linear system here to illustrate the idea of phase space and phase space reconstruction.



(a) (b) the original phase space



(c) time series observation (d) reconstructed phase space

Figure A.2: A linear oscillator, its phase portrait and reconstructed phase space from time series observation

Consider a linear oscillator consisting of a mass attached to a linear elastic spring (Figure A.2(a)). According to Newton's law of motion, the acceleration of the object

is the total force acting on the object divided by the mass.

$$\ddot{x} = \frac{f}{m}$$

Assuming no friction, the spring force f is proportional to the amount that the spring has been compressed, which is equal to the amount that the object has been displaced.

$$f = -kx$$

Combining the two, the system dynamics can be uniquely described by

$$\ddot{x} = -\frac{k}{m}x \tag{A.1}$$

Solving this differential equation, we have

$$x = a\sin(wt + b)$$

where $w^2 = \frac{k}{m}$, the values of a and b depend on the initial condition.

The phase space for such a system is typically $(x, \dot{x})$. The system moves along a closed ellipse periodically (Figure A.2(b)). When friction is taken into account, the phase portrait becomes an inward spiral, since the system will gradually lose velocity.

Now, suppose we only observe a time series $\{x_n\}$, under a certain sampling rate (Figure A.2(c)). We can reconstruct the phase space, as shown in Figure A.2(d), where the embedding dimension m is set to 2, delay τ equals 3. Clearly the reconstructed phase portrait has the same structure as the original system, although a strong correlation exists between the delayed coordinates.

A.2.2 Chaotic Systems

Time-delayed embedding is used extensively in the study of chaotic systems, which have been found to be quite common in daily life. Speech, among other things, has been shown to be chaotic. The phase portrait of a chaotic system is very complex, with the existence of strange attractors being its hallmark. Examples of chaotic systems, their phase portraits, and reconstruction of their dynamics can be found in many books and websites, for example, [Diks, 1999, Kantz and Schreiber, 1997].

A.3 Phase Space Reconstruction for Speech Recognition

In recent years, there has been a growing interest in applying phase space reconstruction to speech recognition. In the classic source-filter model, the speech signal is the combined outcome of a sound source (excitation) modulated by a transfer (filter) function determined by the shape of the supralaryngeal vocal tract. This model is based

on the linear system theory, so are most traditional speech parameterizations, such as the linear predictive coding. It has been argued that phase space reconstruction, as a nonlinear time series analysis technique, fits better with the nonlinear nature of speech. Using delayed embedding directly on the time domain signal, various chaotic features (such as correlation dimension and Lyapunov exponents) are extracted as the basis for recognition. It is reported that although the new chaotic feature does not outperform the traditional MFCC (Mel-Frequency Cepstral Coefficients) feature, a combination of the two tends to improve recognition accuracy [Pitsikalis and Maragos, 2002, Lindgren et al., 2003].

While these works all use time-delayed embedding directly on the time domain signal, we argue that it is more appropriate to embed in the cepstral domain. As a result, the delta and double delta features are actually a phase space representation.

A.3.1 Why Embedding in the Cepstral Domain

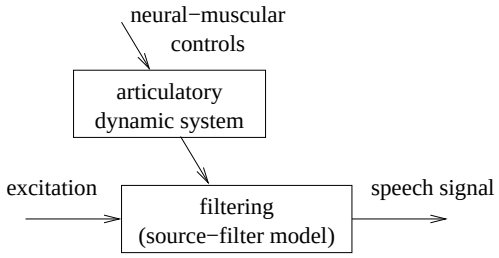


Figure A.3: Two Sub-systems in Speech production

Upon closer examination, there are really two systems involved in speech production (Figure A.3): the filtering system (as in source-filter model) and the articulatory system. The coordinated motion of various articulators determines the shape of the vocal tract, which then filters the sound source, producing the speech signal. Since the ultimate goal of ASR is to infer the phase space of the articulatory system, it is more appropriate to start from a representation of the instantaneous vocal tract shape, rather than directly from the speech signal.

According to the traditional theory, cepstral coefficients are designed to capture the spectral envelope, which is largely determined by the shape of the vocal tract [Rabiner and Juang, 1993]. In other words, cepstrum is a fairly good representation of the vocal tract characteristics. It gives a reasonable source/vocal tract separation. Working in the cepstral domain allows us to focus on the (nonlinear) dynamics of the articulatory system, whereas the dynamics reconstructed from the time domain contains the compounding effects of both systems.

A.3.2 Delta and Double-delta Features

As noted before, delta and double delta features can be computed by stacking several adjacent frames of cepstral vectors together, then projecting down to a lower dimension space by a linear transform.

It should be clear now that modulo the linear transform, dynamic features are exactly time-delayed embedding in the cepstral domain. This leads to the revelation that dynamic features have a fundamental meaning, which is to recover the phase space of the speech production system, i.e. the time-varying articulatory configuration.

There are several caveats, though. First, we are embedding a vector series, not a scalar time series. This is equivalent to taking simultaneous measurements of multiple variables of a system, and, therefore, not a problem at all. Second, speech production is not deterministic. The existence of measurement noise (environmental noise and channel distortion) further complicates the picture of the reconstructed dynamics. Some of the issues are discussed in [Kantz and Schreiber, 1997, Diks, 1999].

One may argue that after all, delayed embedding is only a different representation of the data, without introducing any new information. In the case of speech recognition, we need to justify any changes at the feature level with respect to the underlying modeling framework. The next section will show why time-delayed embedding is essential for HMMs.

A.4 Time-delayed Embedding and HMMs

As many researchers have pointed out, HMMs fail to capture speech dynamics accurately, due to the conditional independence assumption: each frame is conditionally independent of each other, given the state sequence. Several alternative approaches have been proposed to compensate for this weakness, such as segmental models [Ostendorf et al., 1996b], parallel-path HMMs [Iyer et al., 1998], and Gaussian Transition Models [Yu and Schultz, 2003]. Unfortunately, these sophisticated models have yet to show improvements over the seemingly simple HMMs.

Part of the reason is due to the use of time-delayed embedding, i.e. delta and double-delta features. By changing the feature representation, each feature vector now covers a window of consecutive frames, rather than a single frame. Hence, the entity being modeled with HMMs is an entire segment, typically around 100 milliseconds in duration, rather than a single frame of $\sim$20 milliseconds. In a sense, this is segmental modeling in disguise.

The effect of time-delayed embedding on the underlying model can be more formally established in the following scenarios.

A.4.1 Deterministic Dynamic Systems

It is well known that for dynamic systems that can be described by differential equations, a set of first order differential equations is general enough, even for second or higher order systems.

In the above example of a second order linear oscillator, it is easy to convert the system equation to a set of first order differential equations. By introducing a new variable $y = \dot{x}$, equation A.1 can be rewritten as

$$\begin{cases} \dot{x} &= y \\ \dot{y} &= -\frac{k}{m}x \end{cases}$$

In the phase space of (x, y), this is a first order system. In the same spirit, first order Markov models can be elevated to a higher-order model by phase space reconstruction.

A.4.2 Markov Models

A Markov model of order m is a model where the probability at time t depends only on the last m steps. These last m steps define the state of the system. Hence, using time-delayed embedding of the past m samples, the state of the system can be accurately determined.

If the data indeed comes from an m-th order Markov source, we need m-dimensional embedding to model it properly with a first order HMM, since now the probability of the next state (or observation) depends only on the current state (or observation).

A.4.3 Hidden Markov Models

Markov models can be thought of as a special case of HMMs where there is a one-to-one correspondence between states and observations, i.e. states are not hidden. For HMMs, we can no longer strictly prove that a first order HMM can model an mth order source using delayed embedding of order m. It may be a little difficult here to think of HMMs as a generative model in this context. Nonetheless, from a discriminative point of view, each delay vector contains more information about the identity of the HMM state than a single frame.

This is also related to the *false nearest neighbor* method, commonly used in nonlinear time series analysis to determine the minimal sufficient embedding dimension [Kantz and Schreiber, 1997]. If the embedding dimension m is less than the dimensionality of the original system, the reconstructed dynamics won't be a one-to-one image of the original attractor. Instead, "folding" will occur: points are projected into neighborhoods of other points to which they don't belong to. False nearest neighbor can therefore be used as a test for insufficient embedding dimension.

Similarly, with no embedding or insufficient embedding dimension, the feature vector in ASR doesn't carry enough information to accurately determine the state of

the articulatory system. Hence, embedding empowers a first order HMM by increasing the mutual information between feature vectors and their class labels.

A.5 Linear Transformation of the Phase Space

A linear transformation of the phase space does not change the validity of the embedding theorem. It can actually lead to a better representation of the data. As shown in Figure A.2(d), a strong correlation exists between the delayed measurements, which is irrelevant to the structure of the system dynamics. Derivative coordinates (similar to delta and double delta) and principal component analysis have been proposed as alternatives to delayed coordinates [Kantz and Schreiber, 1997]. Both are linear transforms of the original phase space. In the case of ASR, it is clearly worthwhile to apply LDA and MLLT on dynamic features.

Bibliography

A. Acero. *Acoustic and Environmental Robustness in Automatic Speech Recognition*. PhD thesis, Carnegie Mellon University, 1990.

T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul. A compact model for speaker-adaptive training. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1996.

S. Axelrod, R. Gopinath, and P. Olsen. Modeling with a subspace constraint on inverse covariances. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2002.

M. Bacchiani. Automatic transcription of voicemail at AT&T. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2001.

M. Bacchiani and M. Ostendorf. Joint lexicon, acoustic unit inventory and model design. *Speech Communication*, 29(2-4), 1999.

A. Berger, S. Della Pietra, and V. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71, 1996.

S. Burger, V. MacLaren, and H. Yu. The ISL meeting corpus: The impact of meeting type on speech style. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2002.

W. Byrne, M. Finke, S. Khudanpur, J. McDonough, H. Nock, M. Riley, M. Saraclar, C. Wooters, and G. Zavaliagkos. Pronunciation modelling using a hand-labelled corpus for conversational speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 313–316, Seattle, WA, USA, 1998.

L. Deng. Speech recognition using autosegmental representation of phonological units with interface to the trended HMM. *Free Speech Journal*, (2), 1997. http://www.cse.ogi.edu/CSLU/fsj/home.html.

V. Digalakis, P. Monaco, and H. Murveit. Genones: Generalized mixture tying in continuous hidden markov model-based speech recognizers. *IEEE Transactions on Speech and Audio Processing*, pages 281–289, July 1996.

C. Diks. *Nonlinear Time Series Analysis: Methods and Applications*, volume 4 of *Nonlinear Time Series and Chaos*. World Scientific Publishing, 1999.

E. Eide and H. Gish. A parametric approach to vocal tract length normalization. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1996.

M. Eskenazi. Trends in speaking styles research. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, 1993.

M. Finke, J. Fritsch, P. Geutner, K. Ries, and T. Zeppenfeld. The JanusRTk Switchboard/Callhome 1997 evaluation system. In *Proceedings of LVCSR Hub5-e Workshop*, 1997.

M. Finke, J. Fritsch, D. Koll, and A. Waibel. Modeling and efficient decoding of large vocabulary conversational speech. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, 1999.

M. Finke and I. Rogina. Wide context acoustic modeling in read vs. spontaneous speech. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1743–1746, 1997.

M. Finke and A. Waibel. Speaking mode dependent pronunciation modeling in large vocabulary conversational speech recognition. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, 1997.

J. Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proceedings of LVCSR Hub5-e Workshop*, 1997.

E. Fosler-Lussier and N. Morgan. Effects of speaking rate and word predictability on conversational pronunciations. In *Proceedings of the ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition*, May 1998.

K. Fukunaga. *Introduction to Statistical Pattern Recognition (Second Edition)*. New York: Academic Press, 1990.

S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Audio, Signal and Speech Processing*, 34 (1):52–59, 1986.

M. J. F. Gales. Semi-tied covariance matrices. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998.

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 *IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings (ASRU)*, 1997.

J. Glass. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17:137–152, 2003.

J. Godfrey, E. Holliman, and J. McDaniel. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1992.

R. Gopinath. Maximum likelihood modeling with gaussian distributions for classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998.

S. Greenberg. Switchboard Transcription Project. 1996 LVCSR summer workshop technical reports, Center for Language and Speech Processing, Johns Hopkins University, 1996. http://www.icsi.berkeley.edu/real/stp.

S. Greenberg. Speaking in shorthand - a syllable-centric perspective for understanding pronunciation variation. In *Proceedings of the ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition*, 1998.

S. Greenberg, J. Hollenback, and D. Ellis. Insights into spoken language gleaned from phonetic transcription of the switchboard corpus. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1996.

R. Haeb-Umbach and H. Ney. Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1992.

T. Hain. *Hidden Model Sequence Models for Automatic Speech Recognition*. PhD thesis, Cambridge University, 2001.

T. Hain. Implicit pronunciation modelling in ASR. In *ISCA Pronunciation Modeling Workshop*, 2002.

H. Hermansky and S. Sharma. Temporal patterns (traps) in asr of noisy speech. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1999.

T. Holter and T. Svendsen. Combined optimisation of baseforms and model parameters in speech recognition based on acoustic subword units. In *IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings (ASRU)*, 1997.

M. Hwang, X. Huang, and F. Alleva. Predicting unseen triphones with senones. *IEEE Transactions on Speech and Audio Processing*, pages 412–419, November 1996.

R. Iyer, H. Gish, M. Siu, G. Zavaliagkos, and S. Matsoukas. Hidden markov models for trajectory modeling. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1998.

P. Jain. Improved online mean and variance normalization for robust distributive speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2001.

D. Jurafsky, A. Bell, M. Gregory, and W. Raymond. The effect of language model probability on pronunciation reduction. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2001.

T. Kamm, G. Andreou, and J. Cohen. Vocal tract normalization in speech recognition: Compensating for systematic speaker variability. In *Proc. of the 15th Annual Speech Research Symposium*, pages 161–167. CLSP, Johns Hopkins University, 1995.

H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*, volume 7 of *Cambridge Nonlinear Science Series*. Cambridge University Press, 1997.

J.A.S. Kelso, E. Vatikiotis-Bateson, EL Saltzman, and B. Kay. A qualitative dynamic analysis of reiterant speech production: Phase portraits, kinematics, and dynamic modeling. *Journal Acoustical Society of America*, 77:266–280, 1985.

O. Kimball and M. Ostendorf. On the use of tied-mixture distributions. In *Proceedings of ARPA Human Language Technology Workshop*, 1993.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

L. Lamel and G. Adda. On designing pronunciation lexicons for large vocabulary continuous speech recognition. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1996.

L. Lee and R. Rose. Speaker normalization using efficient frequency warping procedures. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1996.

A. Likhodedov and Y. Gao. Direct models for phoneme recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2002.

A. Lindgren, M. Johnson, and R. Povinelli. Speech recognition using reconstructed phase space features. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2003.

X. Luo and F. Jelinek. Probablistic classification of HMM states for large vocabulary continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1999.

W. Macherey and H. Ney. A comparative study on maximum entropy and discriminative training for acoustic modeling in automatic speech recognition. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, 2003.

L. Mangu, E. Brill, and A. Stolcke. Finding consensus among words: Lattice-based word error rate minimization. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, 1999.

D. McAllaster, L. Gillick, F. Scattone, and M. Newman. Fabricating conversational speech data with acoustic models: a program to examine model-data mismatch. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1998.

A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning*, 2000.

M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16:69–88, 2002.

A. Nakamura. Restructuring gaussian mixture density functions in speaker-independent acoustic models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998.

H. Nock and S. Young. Detecting and correcting poor pronunciations for multiword units. In *Proceedings of the ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition*, 1998.

H. Nock and S. Young. Loosely coupled HMMs for ASR. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2000.

F. Och. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2002.

P. Olsen and R. Gopinath. Modeling inverse covariance matrices by basis expansion. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2002.

M. Ostendorf, M. Bacchiani, M. Finke, A. Gunawardana, K. Ross, S. Roweis, E. Shriberg, D. Talkin, A. Waibel, B. Wheatley, and T. Zeppenfeld. Modeling systematic variations in pronunciation via a language-dependent hidden speaking mode. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1996a.

M. Ostendorf, V. Digilakis, and O. Kimball. From HMMs to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(5):360–378, 1996b.

M. Ostendorf and R. Singer. Hmm topology design using maximum likelihood successive state splitting. *Computer Speech and Language*, 11:17–41, 1997.

D. Paul. Extensions to phone-state decision-tree clustering: Single tree and tagged clustering. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1997.

B. Peskin, M. Newman, D. McAllaster, V. Nagesha, H. Richards, S. Wegmann, M. Hunt, and L. Gillick. Improvements in recognition of conversational telephone speech. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1999.

V. Pitsikalis and P. Maragos. Speech analysis and feature extraction using chaotic models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2002.

PRONLEX. COMLEX English Pronunciation Lexicon Release 0.3, LDC Catalog No.: LDC98L21, 1997.

L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. IEEE, 77 (2), 257-286.*, 1989.

L. Rabiner and B-H Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.

M. Ravishankar and M. Eskenazi. Automatic generation of context-dependent pronunciation. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, 1993.

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 Communication*, 29(2-4):209–224, November 1999.

P. Rubin and E. Vatikiotis-Bateson. Measuring and modeling speech production. In S.L. Hopp, M.J. Owren, and C.S. Evans, editors, *Animal Acoustic Communication*. Springer-Verlag, 1998.

M. Saraclar, H. Nock, and S. Khudanpur. Pronunciation modeling by sharing gaussian densities across phonetic models. *Computer Speech and Language*, 14(2):137–160, April 2000.

T. Schultz and I. Rogina. Acoustic and language modeling of human and nonhuman noises for human-to-human spontaneous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1995.

R. Schwartz and S. Austin. A comparison of several approximate algorithms for finding multiple (n-best) sentence hypotheses. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1993.

E. Shriberg. Phonetic consequences of speech disfluency. In *Proceedings of the International Congress of Phonetic Sciences*, 1999.

R. Singh, B. Raj, and R. M. Stern. Automatic generation of sub-word units for speech recognition systems. *IEEE Transactions on Speech and Audio Processing*, 10(2), 2002.

T. Sloboda and A. Waibel. Dictionary learning for spontaneous speech recognition. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1996.

H. Soltau, H. Yu, F. Metze, C. Fügen, Q. Jin, and S. Jou. The ISL transcription system for conversational telephony speech. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2004.

H. Strik and C. Cucchiarini. Modeling pronunciation variation for ASR: a survey of the literature. *Speech Communication*, 29:225–246, 1999.

J. Takami and S. Sagayama. A successive state splitting algorithm for efficient allophone modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1992.

F. Takens. *Detecting Strange Attractors in Turbulence*, volume 898 of *Lecture Notes in Math*. Springer, New York, 1981.

A. Waibel, H. Soltau, T. Schultz, T. Schaaf, and F. Metze. Multilingual speech recognition. In *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 33–45. Springer-Verlag, 2000.

A. Waibel, H. Yu, M. Westphal, H. Soltau, T. Schultz, T. Schaaf, Y. Pan, F. Metze, and M. Bett. Advances in meeting recognition. In *Proceedings of the Human Language Technology Conference (HLT)*, 2001.

M. Weintraub, K. Taussig, K. Hunicke-Smith, and A. Snodgrass. Effect of speaking style on LVCSR performance. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 1996.

M. Westphal. The use of cepstral means in conversational speech recognition. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, 1997.

P. Woodland, T. Hain, G. Evermann, and D. Povey. CU-HTK March 2001 Hub5 System. In *Hub5 2001 Workshop Presentation*, 2001.

S. Young. Large vocabulary continuous speech recognition: a review. In *IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings (ASRU)*, 1995.

S. Young. Statistical modelling in continuous speech recognition. In *Conference on Uncertainty in Artificial Intelligence*, 2001.

S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. *The HTK Book*. Cambridge University, 1995.

S. Young, J. Odell, and P. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of ARPA Human Language Technology Workshop*, 1994.

H. Yu, C. Clark, R. Malkin, and A. Waibel. Experiments in automatic meeting transcription using JRTk. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998.

H. Yu, M. Finke, and A. Waibel. Progress in automatic meeting transcription. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, 1999.

H. Yu and T. Schultz. Implicit trajectory modeling through gaussian transition models for speech recognition. In *Proceedings of the Human Language Technology Conference (HLT)*, 2003.

H. Yu, T. Tomokiyo, Z. Wang, and A. Waibel. New developments in automatic meeting transcription. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2000.

P. Zhan and M. Westphal. Speaker normalization based on frequency warping. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1997.