ADA 275217

Accesion For	
NTIS CRA&I	☑
DTIC TAB	☐
[illegible]	☐
[illegible]	
A-1	

Subphonetic Acoustic Modeling for Speaker-Independent Continuous Speech Recognition

Mei-Yuh Hwang

December 17, 1993

CMU-CS-93-230

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Thesis Committee:
Raj Reddy, Chair
Xuedong Huang, Co-chair
Richard Stern
Tom Mitchell
Kai-Fu Lee, Apple Computer

This research was sponsored by the Department of the Navy, Naval Research Laboratory under Grant No. N00014-93-1-2005.

Keywords: Hidden Markov models, generalized triphones, shared-distribution models, senones, senonic baseforms, classification and regression trees, senonic decision trees.

Carnegie Mellon

School of Computer Science

DOCTORAL THESIS
in the field of
Computer Science

Subphonetic Acoustic Modeling for Speaker-Independent Continuous Speech Recognition

MEI-YUH HWANG

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

ACCEPTED:

[illegible]	12-9-93
THESIS COMMITTEE CHAIR	DATE
[illegible]	12-9-93
THESIS COMMITTEE CHAIR	DATE
[illegible]	1/3/94
DEPARTMENT HEAD	DATE

APPROVED:

[illegible]	1/3/94
DEAN	DATE

Abstract

To model the acoustics of a large vocabulary well while staying within a reasonable memory capacity, most speech recognition systems use phonetic models to share parameters across different words in the vocabulary. This dissertation investigates the merits of modeling at the subphonetic level. We demonstrate that sharing parameters at the subphonetic level provides more accurate acoustic models than sharing at the phonetic level. The concept of subphonetic parameter sharing can be applied to any class of parametric models. Since the first-order hidden Markov model (HMM) has been overwhelmingly successful in speech recognition, this dissertation bases all its studies and experiments on HMMs. The subphonetic unit we investigate is the state of phonetic HMMs. We develop a system in which similar Markov states of phonetic models share the same Markov parameters. The shared parameter (i.e., the output distribution) associated with a cluster of similar states is called a *senone* because of its state dependency. The phonetic models that share senones are shared-distribution models or SDMs. Experiments show that SDMs offer more accurate acoustic models than the generalized-triphone model presented by Lee.

Senones are next applied to offer accurate models for triphones not experienced in the system training data. In this dissertation, two approaches for modeling unseen triphones are studied — purely decision-tree based senones and a hybrid approach using the concept of Markov state quantization. Both approaches indeed offer a significant error reduction over the previously accepted approach of monophone model substitution. However, the purely decision-tree based senone approach is preferred for its simplicity.

The concept of Markov state quantization can also be applied to the automatic determination of a *senonic baseform*. A senonic baseform is a word HMM whose output distributions are replaced by the closest senones, resulting in no increase in the the number of parameters in the existing senonic system. Because it is acoustics driven, the senonic baseform is useful for speaker adaptation and learning new words.

Finally, we explore relaxation of the mixture-tying constraint in semi-continuous HMMs. We move from a system in which the VQ probability densities are tied across all the Markov states in a system to one in which only similar states share the same densities. To reduce computation in order to make experiments possible, phone-class dependent VQ codebooks are studied.

A large suite of experimental results are presented to demonstrate the relative effectiveness of each component of the thesis. After integrating the senonic decision tree with 8 phone-class dependent VQ codebooks into SPHINX-II, we attained a word error rate of 6.7% on the speaker-independent 5,000-word Wall Street Journal continuous-speech recognition task.

Acknowledgments

First and foremost, I would like to thank Raj Reddy, my thesis advisor, for his support, encouragement and guidance. His ability to foresee the future and his unending quest for excellence are the key leading to all the success of my thesis research. Without his generous financial and intellectual support, I would not have been able to survive my graduate life with a daughter and a long-distance husband. Xuedong Huang, my thesis co-advisor, has been advising me closely for the last two years and has given me precious comments on my work. Kai-Fu Lee opened the door to speech recognition for me. Inside Kai-Fu's SPHINX system, I learned the heart of a highly-accurate speech recognizer. I am grateful to Richard Stern's faithful answering of my questions on signal processing and Tom Mitchell's offering me a valuable distinct perspective on AI.

I greatly appreciate the CMU speech group. Fil Alleva, now at Microsoft, was the decoding engine. Without his decoder, I would not have been able to finish my experiments as quickly or with as much success as we had. Ravishankar Mosur, Lin Chase and Roni Rosenfeld all helped with the decoder in the final months. Hsiao-Wuen Hon, now at Apple, generously provided suggestions on research ideas. Bob Weide never hesitated in offering me his linguistic clues; Eric Thayer always helped me with machine setup; and all the other members of the group tolerated my stealing machine cycles and disks everywhere. I owe a great deal to Lin. She proofread my thesis faster than I could write it. Without her help, I would have had to struggle another few months to finish the writing. She is not only a colleague, but also a great and precious *girl* friend, in this tough engineering graduate school.

The School of Computer Science at CMU offers me the greatest learning environment I have ever had — the computing environment, the brilliant faculty, the hard-working students, and the chance to meet famous researchers all over the world. Among the friendly administrators, I would like to especially thank Sharon Burks for her encouragement while I was blue and her suggestions whenever I was puzzled. She is a great mother.

I am indebted to my parents and my parents-in-law who have been supportive and patient throughout the years. Without my mother-in-law's looking after my daughter, I would not have been able to spend twelve hours a day on my thesis. Last but not least, I appreciate my husband's endurance of my bad temper while I was under stress, and his encouraging me to have a successful career. No matter how many quarrels we have, he is always with me when I am sick. The Chinese have a proverb that says

> When there are too many people who you want to thank but you don't know how to enumerate them, then thank God.

Yes, I thank God who brought me to the United States, which offers me an equal opportunity to show my talent, no matter where I am from.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Stochastic modeling techniques have been useful and successful for many applications, including speech recognition. Most of the state-of-the-art speech recognition systems are based on a statistical technique — hidden Markov models or HMMs. In designing an automatic speech recognition system, finding a good speech unit as the basic model is one of the most important issues. Simply-structured units (e.g., phonemes) require minimal training data and result in insufficiently detailed but robust models. To elaborate the structure of the stochastic model of speech in order to obtain more detailed models, the complexity or dimensionality of the speech unit usually needs to be increased (e.g., using word models); this leads to an increase in the degree of freedom in the models. In consequence, more training data are needed to reliably estimate the increased number of free parameters. For instance, training the 50 phone HMMs for English usually requires only 1–2 hours of training data, while to sufficiently train syllable models may require 50 hours of speech. Faced with a limited amount of training data, the advantage of the improved structure of the stochastic model may not be realized. Because of the tradeoff between detailed model structures and available training data, there is always a compromise involved in choosing the speech unit.

This thesis will present the merits of choosing subphonetic events as the basic modeling unit, by both presenting the modeling technique and experimental experiences. In this chapter, we will first discuss the problem of selecting a good speech unit as the basic modeling element. Next, we describe a solution based on subphonetic units called senones. Applications of senones follow next. The last section summarizes the organization of the remainder of this thesis. Since senones were developed on an HMM-based system (though the concept is easily extended to other systems) and because most successful speech recognition systems rely on HMMs, it is crucial to get acquainted with HMMs before reading this thesis. Appendix A contains a quick review of the definition of an HMM, together with references to the fundamental theories and algorithms.

1.1 Selection of a Good Speech Unit

The selection of a particular speech segment as the basic modeling unit for a recognizer is an important design choice, as it determines the detail and robustness of the basic model. To elaborate the structure of a stochastic model in order to obtain more detailed models, the complexity or dimensionality of the model usually needs to be increased; this leads an increase in the degree of freedom in the models. In consequence, more training data are needed to reliably estimate the increased number of free parameters. Conversely, faced with a limited amount of training data, the advantage of the improved structure of the stochastic model may not be realized. Because of the tradeoff between detailed model structures and available training data, there is always a compromise involved in choosing the speech unit.

1.1.1 Word Models

Word models are able to capture within-word phonological variations since the same phone is assimilated by different models when it is realized differently in different words. For example, the expected acoustic realization of phone `/T/` in *ten*, the flapped `/T/` in *thirty*, and the deleted `/T/` in *twenty* are modeled by different parameters within the respective word models. Words are also the most natural units of speech because they are exactly what we want to recognize. For small vocabulary tasks, word models may be suitable since it is not difficult to collect enough utterance samples for each word. For large vocabulary speech recognition, word modeling becomes difficult because many repetitions of each word are needed to obtain reliable estimates. Moreover, acoustic examples of one word are exclusively used for training that particular word; no other word can benefit from them. Furthermore, when a new word is added to a system, there is no way to obtain a model for the new word without collecting utterance samples for this word. Last but not least, it is difficult to model word-boundary co-articulation effects, which are quite serious in spoken English.

1.1.2 Monophone Models

To achieve parameter sharing across different words, subword units like phonetic models are often used because they are both trainable and sharable [6, 125, 89, 84, 104]. Since there are only about 50 phones in English, phones can be sufficiently trained with just a few thousand sentences. Hence, unlike word models, monophone models have no training problem. Word models are formed by concatenating a sequence of phones [6, 125, 89], or networking a set of alternative phonetic pronunciations [30]. Thus, different words that contain the same phone share the same phone model and thus utterances of one word provide training data for parts of the other words. A new word can be easily added to a system without collecting utterance samples.

Despite these many advantages, monophone modeling assumes that a phone produced in one context is equivalent to the same phone in any other context. This is far from the truth [117]. Although we may try to say each word as a concatenated sequence of independent phones, phones are not produced independently because our articulators cannot move instantaneously from one position to another. Thus, the realization of a phone is strongly affected by its surrounding phones. For example, Figure 1.1 illustrates the waveforms and spectrograms of phone /T/ in four different contexts. Actually, the T in *trip* is more similar to phoneme CH and the T in *stop* is more similar to D. Pooling all the acoustic variants of the same phone into one model produces inaccurate modeling for any particular variant.

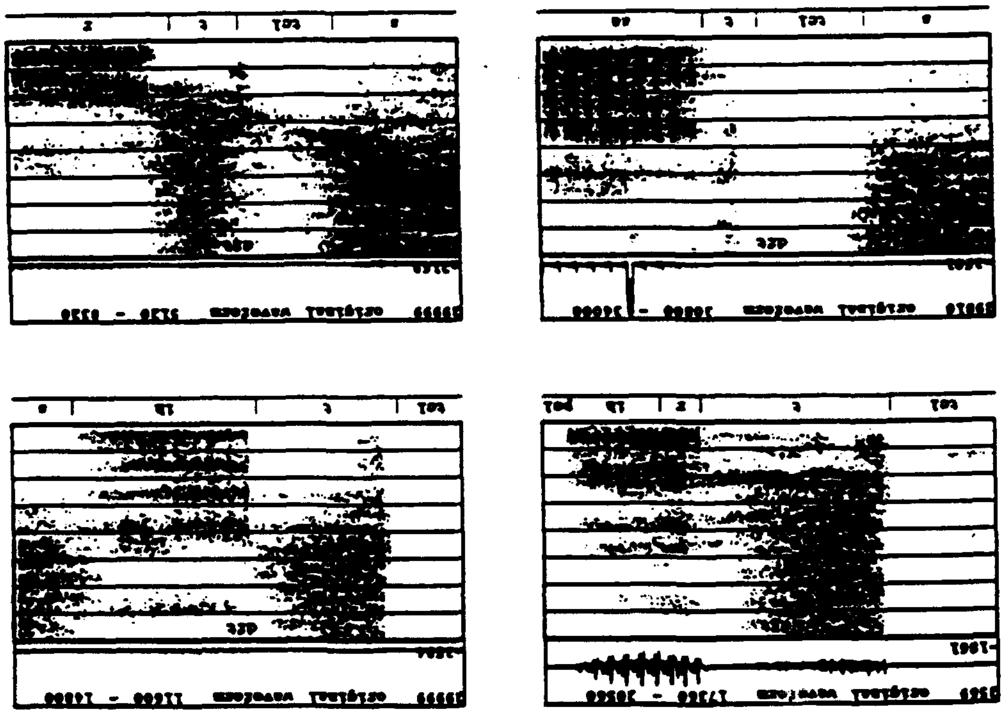


Figure 1.1: The waveforms and spectrograms for the phoneme /T/ in four different contexts: part of /T R/, left of /IH/, part of /S T/, and part of /S T R/. It is clear that the realizations of /T/ are highly dependent on context and that a context-independent /T/ model is inadequate to represent all of them.

1.1.3 Multi-phone Models

One way to share parameters and at the same time keep detailed and consistent models is to use multi-phone units. Examples of these include syllables [59], demisyllables [119], and traditional diphones [124, 79] [1]. Each of these models cover a sequence of phones that contain the most severe contextual effects. However, while the co-articulations in the middle portions of these units are well modeled, the beginning and ending portions are still susceptible to some contextual effects. Even when we define multiple-phone units by breaking word pronunciations at points where contextual effects are as small as possible, we find that the cost of the information we lose at these multiple-phone-unit boundaries is not recovered by the superior modeling of the middle portions [60].

The large number of multi-phone models presents another considerable problem for large-vocabulary tasks, as there are over 20,000 syllables in English. Although there are fewer syllable models than word models in a very large vocabulary, reliable estimation of these parameter values is still not possible with a typical amount of available data (e.g., 20,000 sentences). Moreover, as with word models, there is no parameter sharing between different multi-phone models and thus adequate occurrences in the training data for every single unit must be guaranteed to obtain reliable estimates.

1.1.4 Context-Dependent Phones — Biphone and Triphone Models

As illustrated in Figure 1.1, it is well known that a phone's immediate left and right neighboring phonemes have the most significant impact on the realization of the central phone. Thus, when modeling contexts, we typically try to capture only these relationships. A *left-context biphone* is defined as a *phone* model with a particular phone as its left context. Similarly, a *right-context biphone* is dependent on its right context. Although biphones are sharable across different words and take into account half of the most important contextual effects (either the left or right context, but not both), the remaining contextual effects go unmodeled. For example, in Figure 1.1, the /T/ phones in *t̲rip* and in *st̲op* have different left contexts and are thus well modeled by two different left-context biphones. However, the /T/ biphone with left context /S/ cannot distinguish the two /T/ phones in *st̲op* and *st̲ring*.

To take both contexts into consideration, *triphones* must be defined [125]. A triphone still models only a single phone; yet the same phone surrounded by different left and right phones

[1] Don't confuse the traditional diphones with biphones which will be described in the following subsection. Diphones span more than one phone, while biphones are still phone-sized models with either the left or right contextual phone specified, but not both.

is captured using distinct models. For example, the /T/ in *trip* is modeled by the triphone /T(#,R)/ [2] where # denotes the word boundary, the /T/ in *stop* is modeled by /T(S,AO)/ and the /T/ in *string* by /T(S,R)/. Each triphone model is used to represent a certain phone surrounded by two particular phones. To model word-boundary co-articulation in continuous speech, between-word triphones[62] can be defined using this exact scheme.

Although triphones are very desirable because they are able to capture the most important acoustic variations, they are difficult to train due to the tremendous number of them (on the order of 50^3 in English). With the finite amount of training data typically available, we must either smooth model parameters or reduce the number of parameters, or both.

To smooth parameters, deleted interpolation [72] and heuristic interpolation [125, 123, 29] have been successfully employed to combine less well-trained detailed models with relatively well-trained, but less detailed ones. For example, triphone models can be interpolated with biphone or context-independent phone models [125, 86]. To reduce the number of parameters, one approach is to simply throw away infrequent triphones and train only the most frequent triphones [82]. A missing triphone is then substituted with a biphone or monophone model. This way we obtain sensitive modeling for frequent triphones and sacrifice less frequent contexts to compensate for inadequate data. Another way to reduce the number of free parameters is through sharing. *Generalized-triphone* models [86] share parameters by clustering similar triphones together. That is, all triphones that have similar contexts, such as /T(EY,L)/ in *greatly* and /T(EY,M)/ in *statement*, can be represented by the same generalized triphone model. Generalized triphones were used successfully in the SPHINX system, which was the first system to demonstrate that highly accurate, large vocabulary, speaker-independent continuous speech recognition is possible with real-time performance [85]. One of the important contributions of this thesis is to provide an improvement over generalized triphones by sharing subphonetic units, which will be described shortly.

1.1.5 Subphonetic Models

With regard to choosing a speech unit as the basic modeling unit, we have reviewed word models, multi-phone models (including syllables), monophone and context-dependent phone models. IBM [9] first proposed *fenones* as front-end based subphonetic units in 1987. In 1990, we started working on the *shared-distribution model* [57], which was later developed to be *senonic* models [67]. Meanwhile, Dragon presented subphonetic segments [19] and PELs (phonetic elements) [15] for constructing triphone or PIC (phoneme-in-context) model. Soon after, GEC-Marconi presented *phonicles*, or phonetic particles [136]. In 1992 ATR Interpreting

[2]In this dissertation, a triphone is denoted by specifying the phone first, and then putting the left and right contexts in a pair of parentheses.

Telephony Research Laboratories in Japan proposed the hidden Markov network (HMnet) constructed by the SSS (Successive State Splitting) algorithm [128], which conceptually resembles our work but is very distinct in its implementation. Among the above work, fenones and senones are probably most successful and well known. We will describe the senone and present its achievements first. Details on the related work will be presented in the next chapter.

1.2 Subphonetic Modeling Using Senones

The main goal of this thesis is to demonstrate the superiority of modeling subphonetic units over modeling phones. The subphonetic unit we investigate is the Markov state in phonetic HMMs.

1.2.1 Generalization at a Subphonetic Level

The goal of generalized triphones — grouping acoustically similar triphones into a single model — is to reduce the number of free parameters in the system's acoustic models. However, since it clusters acoustic events at a relatively crude unit, an entire phoneme, it may render over-generalized phonetic models. When two triphone HMMs are merged, all pairs of corresponding Markov-state information (e.g. the output distributions) are averaged. Partial sharing is not supported. This inflexibility can lead to inaccurate Markov states, as it is not necessarily the case that *every* pair of states in two similar triphones is similar.

One solution, which is explored in this thesis, is to analyze each subphonetic event carefully and group only at the subphonetic level in order to avoid over-generalization. We propose to extract the information associated with Markov states of phonetic models (e.g., the output distribution and/or the transition probabilities) and locate similar ones by either a bottom-up or top-down clustering procedure. With this approach, it is possible for two triphones to share only some common states. This flexibility results in accurate modeling at every state and at the same time achieves the goal of parameter reduction.

Conceptually, this searching of Markov state prototypes is similar to VQ clustering, where matches are sought among a set of prototype vectors. Implementationally, output distributions and/or transition probabilities are probabilities rather than simple vectors. Therefore, their distortion measures (which make the notion of "match" precise) need to be defined differently. In Section 3.1, we will first present a bottom-up clustering algorithm which essentially considers all possible configurations [3] and chooses the nearly optimal one, in a minimum entropy sense. After clustering similar Markov states of triphone models, the original Markov states are labeled by the cluster identification, according to the clustering result. The resulting clusters are named senones, after fenones, because they are state-dependent and are analogous to fenones [4]. The set of clusters becomes the contents of a "senone codebook". In SPHINX-II, which will be described in Section 4.2, each state is identified by its associated output distribution (because many researchers share the same experience that transition probabilities play a minor role in the HMM methodology). Thus, a senone is essentially a representative of the output distributions associated with the Markov states in the same cluster. States labeled by the same cluster share the same senone (or interchangeably output distribution). The set of HMMs that share output distributions through senones are called shared-distribution models (SDMs). When senones are used, we have the luxury of using more states for each phonetic model to achieve more detailed acoustic modeling. We rely on the clustering procedure to tie redundant states, if there is any. Given the sharing structure, we then train the HMM parameters so that the likelihood of generating the training data is maximized under the senone sharing constraint.

The shared distribution model has proved to be significantly better than generalized triphones. For instance, on the 1000-word Resource Management speaker-independent continuous speech recognition task (using the standard bigram language model), the senone-based SDM outperformed generalized triphones by 19% in the word error rate. This experiment will be addressed in Section 5.1.

1.2.2 Decision-Tree Based Senones

Faced with limited amount of training data, a recognizer often needs to handle "unseen" triphones — those needed for decoding but never encountered in the training data. When unseen triphones occur, most systems simply substitute monophone or biphone models. As mentioned earlier, both monophones and biphones are not as detailed as triphones, and thus will typically not perform as well. The problem of unseen triphones becomes increasingly serious as we scale up to larger tasks, where a growing number of triphones are needed but commensurate growth in the training data is not possible.

For unseen triphone modeling, a top-down generalization approach based on decision trees [27, 12, 88, 43] has been used to model allophones [5]. Binary decision trees with linguistic

[3]Thus the computational complexity is an exponential function of the input size, the number of states to be clustered.

[4]For a comparison between fenones and senones, see Section 2.4.

[5]Allophones mean phonemes with all possible contextual information. For example, it includes triphones and phonemes with stresses, and so on.

questions are used to classify allophones. Moreover, to improve the quality of the classification, composite questions formed by disjunction, conjunction and/or negation of simple linguistic questions are used with each node in the classification tree. Both seen and unseen triphones traverse the tree, starting from the root, by answering the associated composite question until a leaf is reached. Leaves of the tree represent clusters of similar allophones. This way, both seen and unseen triphones can be modeled by the generalized allophone model, in a unified fashion. Researchers [43, 12] have demonstrated that modeling unseen triphones using the decision-tree based generalized allophones could reduce the word error rate significantly when vocabulary-independent acoustic models were applied to new tasks, in which unseen triphones were more common than in vocabulary-dependent systems.

To incorporate the merits of both the shared distribution model and the decision-tree generalization for unseen triphones, we propose to replace the decision-tree based allophones with decision-tree based senones. That is, we use decision trees to classify Markov states of allphones, rather than classifying the entire phonetic HMM. This yields a decision tree whose leaves represent clusters of similar Markov states. Each leaf corresponds to a senone. To find the senone associated with a given Markov state of a given triphone, we traverse the related tree until a leaf is reached. The Markov states of every triphone, either seen or unseen in the training data, are all labeled by the senonic decision tree in exactly the same way. Figure 1.2 shows an example tree used to pick the senone to be associated with the first state of triphone `K(L,AX)`, which could be seen or unseen in the training data. The highlighted path shows that this state maps to the second senone.

As will be discussed in Chapter 5, modeling unseen triphones with senones reduced the word error rate by 10% on various test sets of the 5,000-word Wall Street Journal task.

1.2.3 Quantization of Markov States

State Quantization for Unseen-Triphone Modeling

The advantage of the senonic decision tree is its capability of modeling unseen triphones with detailed parameters. However, one of its disadvantages is that the senone labeling for seen triphones is likely to be sub-optimal because tree growing is a greedy algorithm in which splits cannot be undone. In contrast, the bottom-up state clustering procedure generates a near-optimal clustering configuration because almost all possible configurations are considered. Therefore, when there are not many unseen triphones in the material to be decoded, decision-tree based senones may render sub-optimal results.

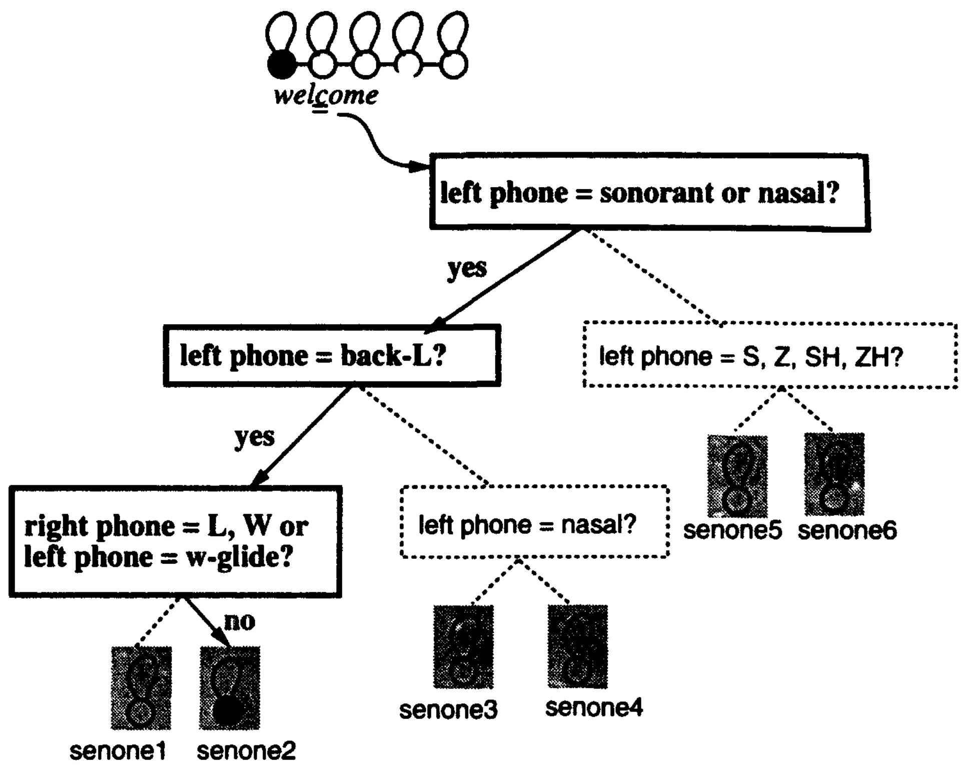

welcome
left phone = sonorant or nasal?
yes
left phone = back-L?
left phone = S, Z, SH, ZH?
yes
right phone = L, W or left phone = w-glide?
left phone = nasal?
senone5
senone6
no
senone3
senone4
senone1
senone2

Figure 1.2: A decision tree for the classification of the beginning states of all K-triphone models.

One way to keep the best clustering configuration for seen triphones and still use senones for unseen triphones is to adopt a hybrid approach. In this hybrid approach, seen triphones are clustered using the bottom-up clustering procedure to produce the best configuration. This is the shared-distribution model, introduced in Section 1.2.1, whose shared distributions constitute the senone codebook. All seen triphones share parameters according to this clustering configuration. To model unseen triphones, decision trees for classifying *triphones* are next constructed [88, 43]. Now we have two sets of parameters, one from senones that comprise the shared-distribution model, and the other from the generalized allophones represented by the tree leaves. To avoid duplication of parameters, the output distributions of the generalized allophone models are replaced with the closest senone each. This process amounts to quantization of the Markov states of the decision-tree based allophones. For each output distribution in the generalized allophone models, we search for the most similar senone and replace the parameters accordingly. To find the senones to be associated with an unseen triphone, the decision tree is first traversed to find the corresponding generalized allophone. Then the sequence of senones resulting from quantizing the Markov states of the generalized allophone is used as the final model for this particular unseen triphone.

With the hybrid approach, seen triphones are guaranteed to have the best senone sharing configuration, while unseen triphones also share the senones, but possibly have a less accurate representation than the representation used in the purely decision-tree based senones, because of possible state-quantization errors. Although this hybrid approach (Markov state clustering and quantization) is not the best for unseen triphones, we expect it is better than simply using monophone or biphone models.

By a series of experiments in Section 5.4, we will show that

- When unseen triphones are modeled by monophones, the SDM based on the bottom-up clustering is indeed slightly better than the purely decision-tree based senones. However, it is only marginally better, as explained by (2).
- Although the clustering for seen triphones using decision trees is less flexible, we will show that the loss for seen triphones can be minimized by (a) using composite (disjunction, conjunction, and/or negation of) linguistic questions to allow the classification taking into account many more configurations than a purely greedy tree-growing algorithm, and (b) incorporating linguistic questions that are able to capture phenomena that the insufficient statistics in HMMs have missed.
- The hybrid approach works as well as the purely decision-tree based senone, when both seen and unseen triphones are modeled by senones. It does not dominate the purely decision-tree senones, as the latter achieves better senone sharing for unseen triphones since there is no state quantization error.

As a result of these experiments, our conclusions are

- The decision tree based senones are more appropriate for very large vocabulary tasks and vocabulary-independent systems, in which unseen triphones are often encountered. They are simpler to implement than than the hybrid approach, because only the senonic decision tree needs to be constructed. In the hybrid approach, three major steps have to be taken: (1) bottom-up clustering for Markov states, (2) top-down tree growing for triphone models, and (3) state quantization.
- For small tasks where essentially all triphones needed for the decoding are covered by the training set, the SDM created by the bottom-up clustering algorithm performs better than the purely decision-tree based senones.

State Quantization for Pronunciation Learning

Foreign (non-English) words often cause a serious problem for acoustic modeling because of distinct phoneme sets used in different languages. They constitute an even more difficult problem when coupled with speaker variations. In this situation purely acoustic driven models are the most useful since the phoneme notation of the home language is discarded. Senones can be applied to construct purely acoustic driven models, using the technique of state quantization.

After the bottom-up clustering or top-down tree classification on the triphone Markov states is finished, we obtain a senone codebook. The Markov states of triphone HMMs are labeled by the senone codebook. Once the senone codebook is generated, it can be used to construct models of all kinds of acoustic phenomena. To visualize this, consider that each senone describes a very short distinct acoustic event (shorter than a phoneme) and that any acoustic event can be described by a permutation of these subphonetic events. For example, when there are enough utterance examples of a certain word, we can estimate a word HMM with a certain number of states by the forward-backward algorithm in order to model all of the acoustic transitions inside a word. After the word model is estimated, we then search for the closest senone for each state of the word HMM. That is, the Markov states in the word HMM are then *quantized* with the senone codebook. This quantized sequence of senones describes the sound for the given word and is called the senonic baseform of the word. A benefit of constructing the senonic baseform is that the number of parameters remains intact even after the word senonic baseform is learned and added to the system. This allow us to obtain new detailed representations for words, without increasing the system complexity.

The senonic baseform was applied to the Resource Management *Spelling* task, where the vocabulary is the 26 letters of the English alphabet. The lexicon of the 1000-word Resource Management task is spelled in this task. We automatically learn two senonic baseforms for each letter, according to the algorithm described above. Senonic baseforms outperformed phonetic baseforms by 15% in word error rate. Follow on attempts to apply senonic baseforms to large vocabulary tasks have not yet proven successful, due to insufficient training utterances and the high acoustic confusibility among the large vocabulary. However, the approach did show potential for applications to speakers with idiosyncratic pronunciations and thus may turn out to be useful for speaker adaptation. We will describe the details of this experiment in Chapter 5.

1.2.4 Semi-continuous HMMs with Senone-Dependent VQ Codebooks

The disadvantage of discrete HMMs (DHMMs) is the inevitable quantization error during vector quantization. On the other hand, although continuous HMMs (CHMMs) get rid of quantization completely, they require the expensive computation of tens of probability densities for every Markov state in the system. The philosophy behind semi-continuous HMMs (SCHMMs) (or tied-mixture HMMs) is to relieve the quantization error in DHMMs by choosing the top few codewords for each data frame instead of just the top one as with DHMMs, and at the same time avoid the expensive computation in CHMMs by tying all the Markov states in a system to the same small set of probability densities. SCHMMs have proved to offer significantly better performance than DHMMs, with only a small increase in computation [44, 51, 22, 21, 77].

As the computer hardware becomes cheaper and faster (150 MHz clock on a DEC alpha 3000/500 in 1993), we may be able to calculate CHMMs in real time in less than one decade. One way to approach the performance of CHMMs without actually going to CHMMs is to reduce the amount of tying in SCHMMs. Traditionally, the VQ probability densities are tied across all the Markov states in a system. When the tying constraints are completely removed, that is, when each state has its own set of VQ probability densities (i.e., mixtures in CHMM terminology), a SCHMM becomes a CHMM. Any small amount of state tying should in principle produce system performance in between the two extreme cases (SCHMMs and CHMMs), as long as there is enough training data.

An example compromise between strict SCHMMs and CHMMs can be derived by training SCHMMs with senone-dependent VQ densities. That is, only similar Markov states will share the same set of VQ densities. Since there are often a large number of senones (4000 – 7000) in a system, the senone-dependent VQ densities are likely to be under-trained if the amount of training data is not increased accordingly. Faced with a limited amount of training data, we may resort to a tighter constraint like phone-dependent VQ densities, [5, 84, 32], or even phone-class-dependent VQ densities. Alternatively, the VQ size (i.e., the number of prototypes) could be reduced in order to be able to train more densities with the fixed training corpus. If we treat DHMMs and CHMMs as the two extreme cases, a spectrum between the SCHMM and its variants can be illustrated as in Figure 1.3.

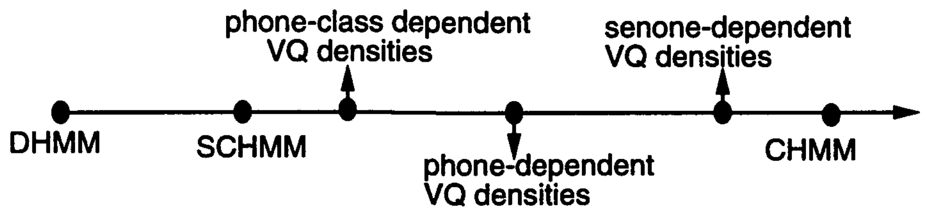

phone-class dependent VQ densities
senone-dependent VQ densities
DHMM
SCHMM
phone-dependent VQ densities
CHMM

Figure 1.3: The relationship of the DHMM, CHMM, SCHMM and its variants. As we release the mixture-tying constraint in SCHMMs, they approach CHMMs.

We will show that with the fixed Wall Street Journal phase-zero training corpus (about 7000 utterances), the best performance is achieved by clustering the 50 phones for English into 8 classes and training phone-class dependent VQ densities. Although the improvement is marginal (8% in total), it is neverless stable across different test sets. Moreover, we believe the improvement will become more appreciable when more training data is available and VQ size is adjusted appropriately.

1.3 Summary and Dissertation Outline

Senones provide a subphonetic modeling based on subunits inside a phonetic HMM. This work will show the superiority and flexibility of using subphonetic units by reference to a series of intensive experiments on ARPA's common data corpora [6]. Although all the subphonetic modeling techniques were implemented for speaker-independent continuous speech recognition in this thesis, they are equally well applicable to speaker-dependent speech recognition and speaker adaptation, either on continuous or isolated speech.

Chapter 2 will review related work on subphonetic modeling for speech recognition. This includes IBM's fenone, GEC-Marconi's phonicle in the U.K., and ATR's HMnet in Japan. The last section will compare these subphonetic models with our work on senones.

Chapter 3 is the central chapter, where the philosophy and methodologies of creating and using senones are elaborated. These include the state clustering for the shared-distribution model, the construction of decision-tree based senones, various state quantization algorithms for unseen triphone modeling and pronunciation learning, and the construction of SCHMMs with phone-dependent VQ codebooks. Experimental results are presented sequentially in Chapter 5. Readers interested in a particular technique can jump to the appropriate section in Chapter 5 to find experimental results.

To put the experiments in context, Chapter 4 describes the experimental environment in which this thesis work was produced. The first section describes the experimental tasks, training corpora and testing conditions. The second section describes the overall structure of the SPHINX-II system, including the signal-processing front end, and the HMM clustering, training and decoding subsystems. The subphonetic modeling techniques presented in this thesis are among the key components of SPHINX-II. All the experiments were carried out on SPHINX-II.

Chapter 5 is a big chapter that contains all the experiments conducted in support of this dissertation. This includes

- Applying the SDM to the Resource Management task.
- Applying the SDM to the Wall Street Journal task.
- Applying decision-tree based senones to the Wall Street Journal task.
- Applying state quantization to the Wall Street Journal task.

[6]ARPA = Advanced Research Projects Agency, Department of Defense.

- Applying state quantization to pronunciation learning in the Resource Management task and the 26-letter Spelling task.
- Applying SCHMM senones with phone-class dependent VQ codebooks to the Wall Street Journal task.

Discussions will be followed after each experiment.

Chapter 6 concludes and situates the experiments by analyzing the conditions under which a particular subphonetic modeling technique is most useful. It also describes the contribution of our work to the literature and some extensions that are being pursued by other people. Finally, some promising future directions of acoustic modeling are discussed.

Chapter 2

Related Work on Subphonetic Acoustic Modeling

In Chapter 1, we reviewed several options for the choice of the basic modeling unit, including word models, multi-phone models (including syllables), monophone, and context-dependent phone models. We also introduced one particular subphonetic modeling technique — senones. Other subphonetic modeling research includes fenones, PELs, phonicles and the HMnet. Here we review each one of these briefly and make a comparison with senones.

2.1 Fenones

IBM [9, 10] first proposed *fenones* as a subphonetic unit in 1987. The purpose of a fenone is to model acoustic phenomena centered around a certain point in the cepstrum domain. These central cepstrum points are the vector prototypes created by the VQ clustering procedure. In other words, each fenone models one VQ codeword. The word *fenone* comes from the *f*ront-*e*nd phone to replace the linguistic phone. The VQ level used at IBM is 200; therefore there are 200 fenonic HMMs. Each fenone is defined by a one-state discrete HMM, as Figure 2.1 shows, where the dotted arc is a null transition.

Since each fenone has only one Markov state, it represents a very short acoustic interval and thus provides a much finer level of detail than a phonetic model. To decide the fenonic pronunciation (or fenonic baseform) for a word, utterance samples for the word have to be collected first. When there is only one utterance example, the fenonic baseform is decided by the sequence of fenones corresponding to the sequence of VQ codewords when the utterance

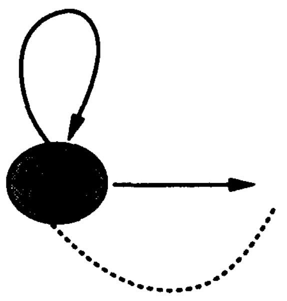


Figure 2.1: The HMM structure of a fenone.

is vector quantized. When there are multiple utterances ($y^{(1)}$, $y^{(2)}$, ..., $y^{(n)}$) for the given word, the fenonic baseform is decoded by searching for the *best* sequence of fenones, assuming that a set of fenonic models already exists. The best sequence of fenones, $\hat{\mathbf{f}}$, has the maximum joint probability of generating the given utterances:

$$\begin{aligned} \hat{\mathbf{f}} &= \underset{\mathbf{f} \in \mathbf{F}^*}{\text{argmax}} \; Pr\{y^{(1)}, y^{(2)}, ..., y^{(n)} | \mathbf{f}\} \\ &= \underset{\mathbf{f} \in \mathbf{F}^*}{\text{argmax}} \prod_{i=1}^{n} Pr\{y^{(i)} | \mathbf{f}\} \end{aligned} \tag{2.1}$$

where **F** is the fenone alphabet, **F*** is the transitive closure, and utterance samples $y^{(i)}$s are assumed to be independent. Solving $\hat{\mathbf{f}}$ is similar to *stack decoding* [7] a sentence over all possible fenone strings, with **F** as the vocabulary, no language model (any fenone can be followed by any fenone), and replacing the acoustic match probability with the product of the acoustic match probabilities over all the utterance samples.

Fenones were tested on a 2000-word office correspondence task in a speaker-dependent isolated-speech mode [9]. Both the training and testing data sets each contained one utterance for each word in the vocabulary, from one speaker only. No language model was used. The word error rate using monophone models is shown in the first row of Table 2.1. In the second experiment, a third utterance (other than the training and testing utterances) was used to obtain the fenonic baseform according to the sequence of VQ codewords. The learned fenonic baseforms were trained by the training corpus which consists of one sample for each word in the vocabulary. The third experiment used another 4 utterances for each word to obtain the fenonic baseform according to (2.1). With the same training corpus, the fenonic baseform learned from 4 separate utterances performed significantly better than monophone models.

Acoustic Model	# utterances for learning fenonic baseforms	Word Error Rate
monophone	0	5.2%
fenone	1	7.4%
fenone	4	2.2%

Table 2.1: Word error rates on the 2000-word office correspondence speaker-dependent isolated speech recognition task, without any language model.

2.2 Phonetic Elements

In 1990, Dragon Systems Inc. manually located about 2,000 subphonetic segments with the help of an interactive program [19]. Each triphone or PIC (phoneme-in-context) was defined as a sequence of these subphonetic segments. When they tried to build large vocabulary systems in 1992, this manual process was replaced by a k-means clustering algorithm which clustered the spectral frames of speech from the same phoneme [15]. These automatically identified distinct frames were called the PELs (phonetic elements). Each PEL was modeled by one Markov state with an output distribution and a Laplacian duration. A PIC model is thus a linear sequence of PEL models.

PICs with PELs were tested on a 5,000-word Wall Street Journal task in the speaker-independent, continuous speech mode. In this test run, 21,000 PELs were created. 12320 PICs which occurred at least 10 times in the training corpus were selected. Most PICs consisted of 2 PELs; a small number of them consisted of only 1 PEL. PELs were not shared across different PICs in this particular run. The reported word error rate was 14.1% [103] As we will see in Chapter 5, SPHINX-II with senones performed a word error rate of about 7% on the same test set for the same task.

2.3 Phonicles

GEC-Marconi presented phonicles in 1991 [136]. *Phon*etic part*icles* or phonicles, are two-state HMMs that represent either the first half or last half of a phone. The philosophy is that the behavior of the left part of a triphone should be similar to that of another triphone with the same left contextual phoneme; analogously the right part with the right contextual phoneme. Each triphone HMM is thus defined to be 4 states, composed of two phonicles. Furthermore, phonicles are both left-context and right-context dependent. A top-down linguistic-knowledge based clustering approach is used to reduce the number of phonicles from 6000 to 2000 and

Model Level	Explanation	Left Phonicle	Right Phonicle
level 1	triphone	I_1(S,K)	I_2(S,K)
level 2	reduced triphone	I_1(S,VELAR)	I_2(ALVEOLAR,K)
level 3	biphone context	I_1(S,*)	I_2(*,K)
level 4	reduce biphone	I_1(ALVEOLAR,*)	I_2(*,VELAR)
level 5	monophone context	I_1(*,*)	I_2(*,*)

Table 2.2: Representation of the triphone I(S,K) using phonicles. Phonicle level i is used only if level $i-1$ does not occur in the training data. * is a wild card, i.e., any phone context can be used.

to capture unseen contexts. For example, the triphone I(S,K) is modeled by the five pairs of phonicle models shown in Table 2.2, with phonicle level i being used only if phonicle level $i-1$ is not seen in the training data.

Phonicles were tested on a 98-word Air Traffic Control (ATC) task in a speaker-dependent mode [136] over four male speakers. For isolated speech recognition, each subject spoke two repetitions of the ATC words in isolation. For continuous speech recognition, each subject spoke the same 100 continuous ATC sentences. Neither test used a grammar. The training data consisted of 200 phonetically rich sentences spoken by the four speakers. The vocabulary of the training data was not based on any application-specific domain.

Average word error rates over the 4 speakers are shown in Table 2.3. The results show that triphones without generalization perform significantly worse than generalized phonicles, because of insufficient training data. Advanced work on phonicles can be found in [41].

Acoustic Model	Modeling of Unseen Contexts	Isolated Speech Word Error Rate	Continuous Speech Word Error Rate
monophones	monophones	29.9%	45.3%
triphones	monophones	24.9%	42.7%
context-dependent phonicles	top-down knowledge-based clustering	13.8%	27.6%

Table 2.3: Word error rates on the 98-word Air Traffic Control speaker-dependent speech recognition task, without any language model.

2.4 The HMnet

In 1992, ATR Interpreting Telephony Research Laboratories in Japan proposed the hidden Markov network (HMnet) constructed by the SSS (Successive State Splitting) algorithm [128], which conceptually resembles our Markov state sharing but is very distinct in implementation.

The rationale behind the *h*idden *M*arkov *net*work or HMnet [128] is the same as with senones, i.e., to share state information in order to reduce parameter redundancy. An HMnet consists of a network of Markov states. Each state specifies the acceptable contextual class at that particular state, in addition to the normal output distribution and transition probabilities. There is only one HMnet for the entire given task. A triphone is defined by the sequence of states that accepts the specified contexts. For example, the path consisting of the shaded states in Figure 2.2(a) shows the component HMM for triphone `G(I,I)`, and Figure 2.2(b) shows the HMM for both `G(I,A)` and `G(I,O)`.

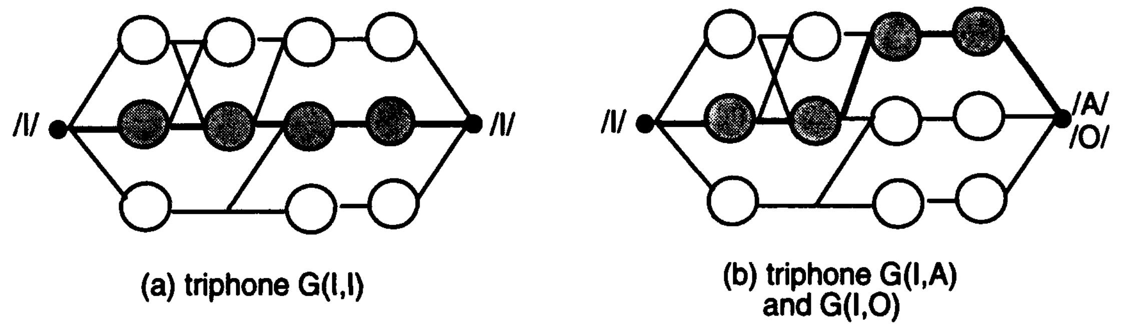


Figure 2.2: An HMnet example. The shaded path shown in (a) corresponds to triphone `G(I,I)`. Figure (b) illustrates the path for both triphone `G(I,A)` and `G(I,O)`.

An HMnet is generated automatically by the SSS algorithm. Starting from a one-state CHMM, the SSS algorithm successively splits the state which has the maximum value of the output probability density, into two states. The split can be either horizontal in the temporal domain or vertical in the contextual domain. To be able to handle triphones, the HMnet is combined with a phoneme-context-dependent LR parser [101, 130] to predict the right phoneme context so that a particular path in the HMnet can be selected and evaluated.

The HMnet was tested on a 1000-word speaker-dependent Japanese phrase recognition. Here only a professional announcer was tested. The training data consisted of 5240 common Japanese isolated words, plus 216 phonetically balanced words, plus some phrase utterances to compensate for different speaking rates. The test data consisted of another 279 phrase utterances. Performances using the HMnet with single Gaussian densities and using continuous HMMs with Gaussian mixture densities are shown in Table 2.4.

Acoustic Model	Phrase Error Rate
HMnet (single Gaussian)	4.7%
CHMM (mixture Gaussians)	6.8%

Table 2.4: Speaker-dependent Japanese phrase recognition using the HMnet with continuous output probability densities.

2.5 Comparison of Subphonetic Modeling Techniques

Fenones are used to model the VQ prototypes. Hence, they are purely acoustic. No linguistic information such as phonemic analysis is used. Therefore, the approach is as suitable in automatic pronunciation learning and cross-language acoustic modeling as the senonic baseform is. However, there are only 200 fenones, which may be too few to provide enough resolution in the speech cepstrum space for large-vocabulary tasks. Increasing the number of fenones will increase the number of parameters significantly since these one-state HMMs use discrete output distributions. One way to increase the fenone resolution is to increase the number of fenones without increasing the VQ level for the discrete output distributions. That is, two separate VQ clusterings are needed, one for creating the central cepstrum points which fenones model, the other for quantizing the training data while training the discrete output distributions.

Unlike fenones, senones have excellent resolution for modeling various acoustic phenomena. The shared-distribution model is both phonetic and acoustic [1] in the sense that words are still spelled by phones, but phones are defined by a sequence of senones, which are acoustically similar events. We can also construct purely acoustic driven baseforms using the senonic baseform. To determine a senonic baseform, the best sequence of senones to describe a word sound is determined. This is similar to the fenonic baseform in which the best sequence of fenones is searched. However, two dissimilarities are noted. First of all, senones are Markov-state based, while fenones are time-frame based. Secondly, the senonic baseform is solved as a word-HMM training and then state quantization problem, while the fenonic baseform is solved as a decoding problem as Section 2.1 describes.

Another drawback of fenones is that there is no way to decide a word's baseform unless utterance examples are given, although fenones are shared across different words. We cannot build the senonic baseform for a word, either, when there is no utterance sample, but the phonetic shared-distribution model for the word can be used together with the senonic baseforms for other words, without an increase in the number of free parameters since both the senonic baseform and the phonetic shared-distribution model share the same set of senones.

[1] The senonic baseform is purely acoustic.

PICs with PELs are not successful because parameter sharing is either not achieved, which results in under-trained models, or the sharing structure is manually defined and thus subjective. Clustering on individual spectral frames instead of a wider range (which offers more information) may also count for the inaccurate model.

A phonicle is similar to a senone except the size of the model is doubled and thus some detailed modeling is disabled. For example, triphones that have the same left phonicle are restricted to use the same two output distributions. That is, the first/last two states of two similar triphones are always indistinguishable. Given their high error rates on the ATC task, we believe that there is still plenty space on improving their clustering quality. Splitting the clustered objects from two states to one state as senones do will help their system also.

The concept of state sharing in the HMnet is the same as senones. The implementation, however, is quite distinct. To build senones, we start from the full phonetic HMM structure for all triphones and then reduce the number of shared states to a trainable size. The HMnet, on the other hand, starts from one single state and grows (splits) successively to form a final network for the entire task. The HMnet is currently used for only phoneme and phrase recognitions. Applying the HMnet to continuous sentence recognition requires the incorporation of language models. More importantly, it is unclear if the SSS algorithm can be efficiently and accurately applied to very large vocabularies. How to decide the size of the HMnet and how to optimize the splitting algorithm so that the HMnet can represent all the enormous number of triphones for large vocabulary tasks are the most difficult problems.

2.6 Summary

Related work on subphonetic acoustic modeling includes fenones, PELs, phonicles, and HMnet. Among them, fenones are probably the best known. A fenone is a VQ-based one-state HMM. It is purely acoustic driven and is most suitable for automatic pronunciation learning given few training tokens. PELs are generated by a k-means clustering algorithm which clusters the spectral frames. A phonicle models either the front or the back half of a phoneme. Two triphones with the same left or right context are forced to share the same left or right phonicle. Shared phonicles were demonstrated to be better than triphones without generalization. An HMnet is a network of hidden Markov states, each of which is associated with acceptable contexts in addition to the normal HMM parameters. The HMnet is constructed by the SSS algorithm and an allophone is defined by a path in the HMnet. The HMnet with a single Gaussian output density was demonstrated to perform better than CHMMs with Gaussian mixtures.

The shared distribution model described in Chapter 1 is both phonetic and acoustic. The algorithm (either bottom-up clustering or the top-down decision-tree classification) used to generate senones is based on the trained parameters of triphone HMMs, which contain acoustic statistics about phonetic structures. After senones are generated, they are used to label the original triphones and then senones are trained through the sharing structure across all triphones. The phonetic structure is still maintained — words are spelled by the phonetic transcription. Thus, the senonic modeling analyzed from the signal-cepstrum distribution is combined elegantly with the phonetic transcription derived from informed linguistic analysis. The trained senones can also be used to build purely acoustic driven senonic baseforms for speakers with idiosyncratic pronunciations or for task-specific jargon.

Chapter 3

Subphonetic Modeling on Markov States

The goal of this chapter is to explain all the subphonetic modeling techniques explored in this thesis. This includes the clustering of Markov states to create state prototypes named senones (the model that shares senones is called the shared-distribution model), the use of decision trees to create senones for unseen-triphone modeling, the quantization of Markov states for alternative unseen-triphone modeling and pronunciation learning, and the introduction of SCHMMs with phone-dependent VQ codebooks for relaxing the mixture-tying constraint in SCHMMs without an expensive computational cost as CHMMs.

Experimental results for these techniques on a standard set of tasks are presented in Chapter 5. These experiments were based on speaker-independent continuous speech recognition. The same techniques, however, can be applied to speaker-dependent speech recognition and speaker adaptation, either on continuous or isolated speech tasks. Interested readers can jump to the appropriate section in Chapter 5 to get acquainted with the experimental performance of a particular modeling technique, while reading the descriptions that follow.

3.1 Clustering of Markov States — the Shared-Distribution Model

Parameter sharing plays an important role in statistical modeling since the amount of training data is usually limited. On one hand, we would like to use models that are as detailed as possible, such as triphones or even context-dependent multi-phone models. On the other hand, with models that are too detailed, we quickly lose the ability to reliably estimate the growing number of parameters. Faced with a limited amount of training data and a huge number of free parameters, we must either smooth the parameters or share the parameters across different models and very often do both. This section will present a novel technique, the shared-distribution model (SDM), for parameter sharing. Parameter smoothing will be summarized in the following paragraph. Interested readers are encouraged to browse the indicated references.

To smooth parameters, both deleted interpolation [72] and heuristic interpolation [125, 123, 29] have been successfully employed to smooth less well-trained detailed models with relatively well-trained, but less detailed ones. For example, triphone models can be interpolated with biphone or context-independent phone models [125, 86]. Semi-continuous HMMs (SCHMM) [49] or tied-mixture HMMs [22, 21] achieve both parameter smoothing and sharing. For smoothing, SCHMMs use the top few best-matched codewords instead of only the best one to represent the input signal during vector quantization. Additionally, SCHMMs share the parameters by tying all Markov states of all phonetic models to the same set of continuous probability density functions (i.e., the VQ density functions).

To share the parameters, generalized-triphone models [86] group similar triphones together when they are poorly trained or when they resemble each other. Generalized triphones were successfully used in SPHINX and demonstrated the feasibility of highly-accurate large-vocabulary speaker-independent continuous speech recognition. In the next subsection, we will describe the inherent problem with generalized-triphone models and present as a substitute the shared distribution model. The SDM reduces the number of free parameters and at the same time maintains models that are as detailed and accurate as possible. A series of experiments will be performed in Section 5.1 to demonstrate the superiority of the SDM over the generalized-triphone model.

3.1.1 Parameter Reduction and Parameter Sharing

Triphones have been one of the most successful acoustic modeling units in the state-of-the-art systems [66, 121, 82]. The simplest way, which is pursued by [82], to reduce the number of free parameters is to eliminate infrequent triphones since it is unlikely that the trained models are robust. The consequences of this are that part of the training data is wasted and the system loses the capability of modeling these discarded triphones, which could be considerable in number.

When triphones are used, some parameters are redundant in the same sense that they are so similar that they could have been represented by the same model and thus make the model more trainable. The model which represents a cluster of similar triphones is called a generalized triphone [86]. For example, the place of articulation has an important effect on the right-neighboring vowel. Figure 3.1 illustrates that both /R/ and /W/ have similar effects on the right-neighboring vowel /IY/; therefore, there is no reason to keep two separate /IY/

triphones for the /IY/ phone in *street* and *sweet*.

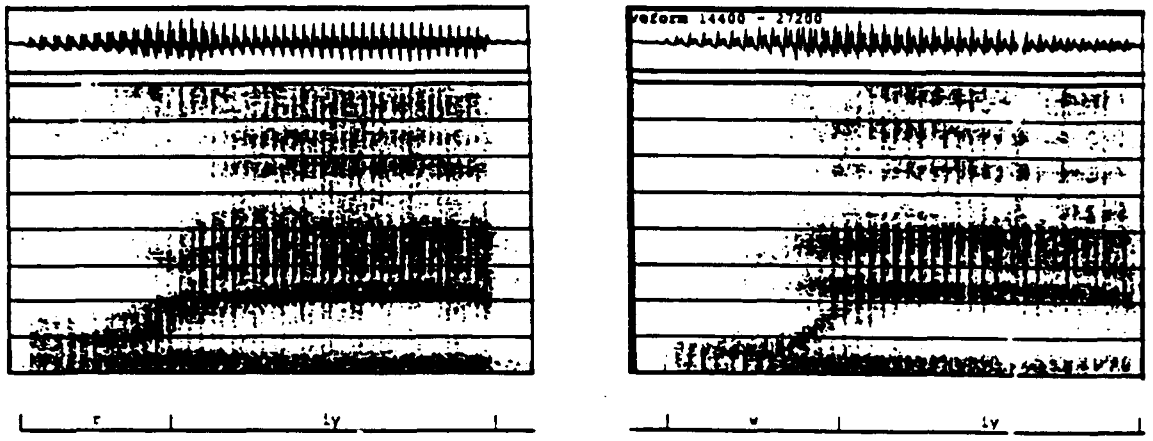


Figure 3.1: The waveforms and spectrograms for the phoneme /IY/ following /R/ and /W/. Note that R and W have similar effects on IY.

The Problem with Generalized Triphones

As a parameter-sharing technique, generalized triphones are intended to represent a cluster of similar triphones with the same model. However, averaging a set of triphones completely can result in blending dissimilar subphonetic units when only parts of the triphones are similar. Figure 3.2 illustrates the problem with generalized triphones. The top model shows the output distributions of triphone EY(L,S) in a word like *place*, and the middle model for triphone EY(L,SH) in *relation*. It is reasonable that we see the strong similarity between the first and second output distributions since they have the same left context. The first two output distributions could be so similar that the overall distance between these two triphones is small and thus the two triphones are merged into one generalized triphone. This results in inaccurate representation for the last state when the last states of these two triphones have different distributions. We call this the "mis-average" problem.

The Shared-Distribution Model

The mis-average problem associated with Generalized triphones can be avoided if sharing/clustering is carried out at a sub-model level — the Markov state. State-information sharing has been successfully employed for speaker-adaptive speech recognition where adaptive distributions are shared across different phonetic models [45]. Unlike model-level clustering, state-level clustering merges two states only if the associated output distributions exhibit

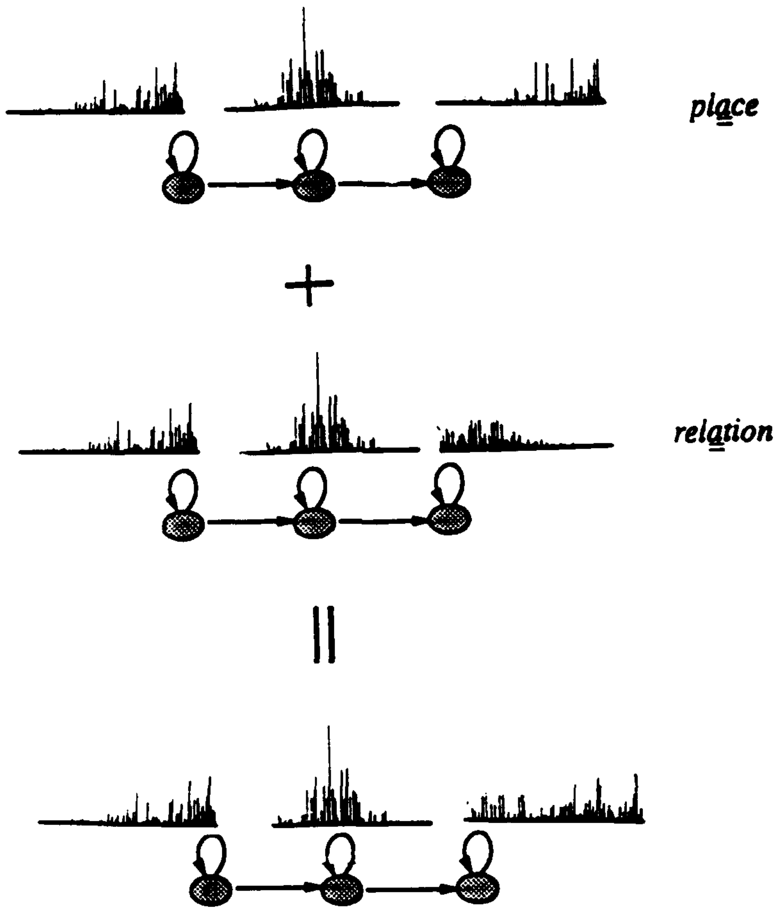


Figure 3.2: Illustration of the problem with generalized triphones. When the last states of these two triphones are represented by their average, the average is no longer an accurate representation for either of them. We call this the "mis-average" problem.

acoustic similarity. Thus, as illustrated in Figure 3.3, the first two states of triphone EY(L,S) share the same output distributions as the first two states of triphone EY(L,SH). However, the last states of both triphones are maintained with different output distributions in order to keep the distinction.

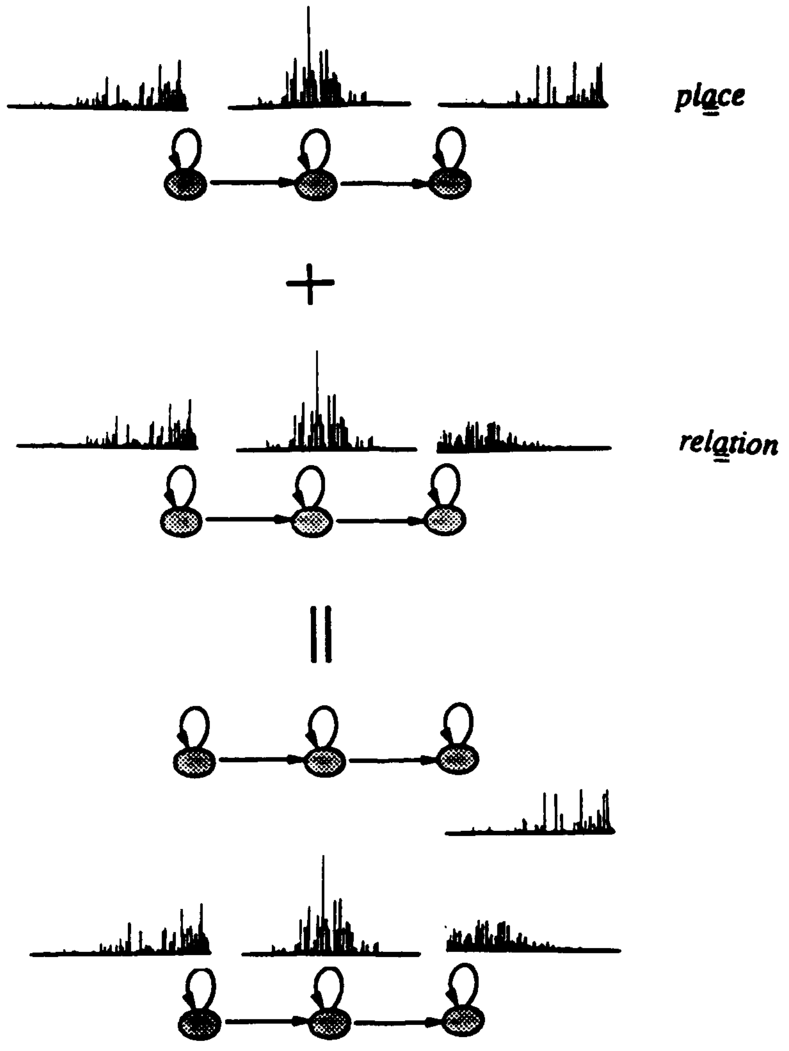


Figure 3.3: State-information sharing circumvents the "mis-average" problem of generalized triphones, and at the same time achieves parameter reduction.

This sharing also gives us the freedom to use a larger number of Markov states for each phonetic model in order to achieve more detailed acoustic modeling. Although an increase in the number of states will increase the total number of free parameters, with state-information sharing we can "collapse" redundant states by tying them to the same output distribution, while maintaining the more useful distinct states. As long as the number of states per HMM and the total number of shared output distributions are properly determined as a function of the training data, we can achieve both detailed and robust modeling of speech signals. The principle of state-information sharing can be applied to either discrete, continuous, or semi-continuous HMMs.

To build the SDM, an information-theoretic distortion measure is used to cluster state information across different triphone models. The information we use to uniquely identify states is the associated output distribution only, as transition probabilities have been shown to be of negligible importance. In Section 5.1 we will demonstrate that the SDM indeed provides a better representation than the generalized-triphone model. On the 997-word DARPA Resource Management (RM), speaker-independent continuous speech recognition task [112], the word error rate was reduced by 20% in comparison with generalized triphones.

3.1.2 The Bottom ɹp State-Clustering Algorithm

Suppose a set of triphone names and their corresponding models are given and we want to locate similar subphonetic units so that they can be shared across different triphones. Representing a large number of triphones in terms of a sequence of a small number of subphonetic units reduces the number of free parameters and thus each parameter can be better trained. The concept of subphonetic sharing can be applied to any modeling technique that provides subphonetic units, not just HMMs. We use HMMs as our platform because the HMM is the start-of-the-art speech modeling technique. The individual Markov states inside phonetic HMMs are the subphonetic units we investigate.

> The goal is to search for similar Markov states so that their output distributions can be averaged to form a *senone*, which is a general and still accurate representation of the original component distributions of triphone models. The set of triphone models that is built using senones is called the shared-distribution model (SDM).

Since similar Markov states of distinct triphone models are tied with the same senone as their output distribution, the number of free parameters associated with the triphones is reduced. Unlike generalized triphones' *all-or-nothing* principle which forces either all or none of the Markov states of two triphones to be shared with each other, the state sharing allows partial sharing between two triphones.

Distance Measure between Two Markov States

We assume that similarity between two states is determined solely by the associated output distributions since the output distribution plays a much more important role than transition probabilities in HMMs. There are many ways to measure the distance between two distributions/HMMs. Examples include cross entropy [81, 75], output string/symbol probability [34], maximum mutual information [34], and the chi-square measure [110]. Both generalized triphones [86] and the SDM use a variant of cross entropy as the distance measure. Because of the key role of entropy and cross entropy (i.e, directed divergence), readers are encouraged to review the definitions in Appendix C.

Entropy Increase as the Distortion Measure Since both SPHINX and SPHINX-II use discrete output distributions, all the details about the distance measure described here are defined for discrete probability distributions. It would not be difficult to modify our definition for continuous probability densities. Suppose we want to compute the distance between distribution $\mathcal{A}$ and distribution $\mathcal{B}$. The occurrence *counts* for each entry in $\mathcal{A}$ and $\mathcal{B}$ are denoted as $a_1, a_2, ..., a_L$, and $b_1, b_2, ..., b_L$ respectively, where L is the VQ level. Moreover, let $\sum_i a_i = A; \sum_i b_i = B$.

When we decide two distributions are similar and merge them into the same cluster, the occurrence count for the resulting distribution/cluster ($\mathcal{A} + \mathcal{B}$) is the summation of the two merged distribution counts. Therefore, the entropy for the resulting distribution is:

$$H_{\mathcal{A}+\mathcal{B}} = -\sum_i \frac{a_i + b_i}{A+B} \log(\frac{a_i + b_i}{A+B})$$

We define the distance measure between $\mathcal{A}$ and $\mathcal{B}$ as the entropy increase, weighted by counts, due to merging $\mathcal{A}$ and $\mathcal{B}$ as:

$$\begin{aligned}
\Omega(\mathcal{A},\mathcal{B}) &= (A+B)H_{\mathcal{A}+\mathcal{B}} - AH_{\mathcal{A}} - BH_{\mathcal{B}} \\
&= -\sum_i a_i \left[\log\left(\frac{a_i+b_i}{A+B}\right) - \log\frac{a_i}{A}\right] \\
&\quad -\sum_i b_i \left[\log\left(\frac{a_i+b_i}{A+B}\right) - \log\frac{b_i}{B}\right] \\
&= -A\sum_i \left(\frac{a_i}{A}\right)\log\left[\left(\frac{a_i+b_i}{A+B}\right) / \left(\frac{a_i}{A}\right)\right] \\
&\quad -B\sum_i \left(\frac{b_i}{B}\right)\log\left[\left(\frac{a_i+b_i}{A+B}\right) / \left(\frac{b_i}{B}\right)\right] \\
&= A\,\chi(\mathcal{A},\mathcal{A}+\mathcal{B}) + B\,\chi(\mathcal{B},\mathcal{A}+\mathcal{B}) \qquad (3.1)
\end{aligned}$$

where $\chi(.,.)$ stands for the cross entropy. It turns out that $\Omega(\mathcal{A},\mathcal{B})$ is the summation of two *weighted* directed divergences — one between $\mathcal{A}$ and $\mathcal{A}+\mathcal{B}$, the other between $\mathcal{B}$ and $\mathcal{A}+\mathcal{B}$. As cross entropy is non-negative, $\Omega(\mathcal{A},\mathcal{B})$ is also non-negative. It reaches its minimum (zero) if and only if $a_i/A = b_i/B, \forall i$ (when the two *probability* distributions are exactly the same). This entropy increase is what we use in SPHINX-II for measuring the distance between two output distributions. The smaller the entropy increase is, the more similar the two distributions are. In SPHINX, the distance between two triphones was defined as the summation of all the entropy increases between pairwise Markov states of the involved triphone models.

Weighting entropy by the occurrence count can also lend information about how well each distribution is trained. Distributions that appear infrequently will be merged before those that are well trained. Thus infrequent distributions become more trainable after clustering. Another advantage of weighting cross entropy by the occurrence count is that large clusters are prevented from getting blindly bigger. To see this, suppose $\mathcal{C}_1$ is a big cluster with many elements, compared with a smaller $\mathcal{C}_2$, as shown in Figure 3.4. Because of the significant gap between the occurrence counts of $\mathcal{C}_1$ and $\mathcal{C}_2$, the count entries in $\mathcal{C}_1+\mathcal{C}_2$ are almost the same as those in $\mathcal{C}_1$, resulting in $\chi(\mathcal{C}_1,\mathcal{C}_1+\mathcal{C}_2)\approx 0$. Had the cross entropy *not* been weighted by the occurrence count, $\mathcal{C}_1$ and $\mathcal{C}_2$ could have been merged before anther small cluster $\mathcal{C}_3$ got a chance to be collapsed with $\mathcal{C}_2$, even when $\mathcal{C}_2$ and $\mathcal{C}_3$ are quite similar, since $\chi(\mathcal{C}_2,\mathcal{C}_2+\mathcal{C}_3)+\chi(\mathcal{C}_3,\mathcal{C}_2+\mathcal{C}_3)$ might be greater than $\chi(\mathcal{C}_1,\mathcal{C}_1+\mathcal{C}_2)+\chi(\mathcal{C}_2,\mathcal{C}_1+\mathcal{C}_2)\approx\chi(\mathcal{C}_2,\mathcal{C}_1)$.

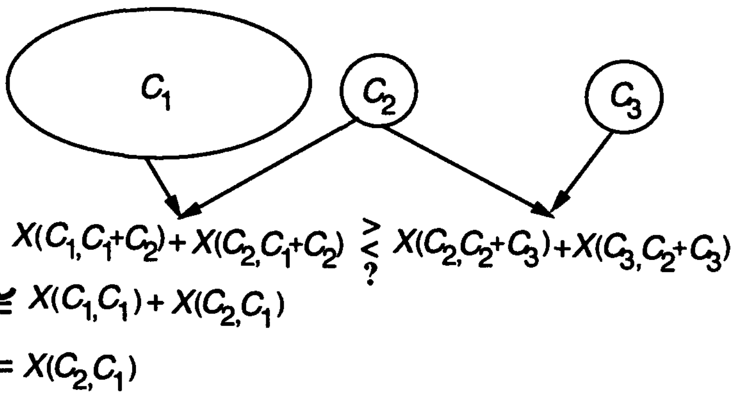


Figure 3.4: Hypothesizing the distance measure without weighting cross entropy by the occurrent count. Both $\mathcal{C}_2$ and $\mathcal{C}_3$ are small clusters, while $\mathcal{C}_1$ is a such a big cluster that $\mathcal{C}_1+\mathcal{C}_2\approx\mathcal{C}_1$.

However, we should remember that weighting cross entropy by the occurrence frequency has a risk of mis-representation when the frequency in the training data does not reflect the test data and can be harmful under this situation.

The Bottom-Up Clustering Algorithm

Using the distance measure defined in (3.1), a bottom-up agglomerative state clustering algorithm is summarized in Figure 3.5 [86]. The output of the clustering is a table listing clusters of similar Markov states. States in the same cluster use the same senone to represent their output distributions. The set of triphone models whose Markov states share output distributions according to the table is called the shared-distribution model.

1. Estimate all triphone HMMs (e.g., using the forward-backward algorithm[20, 7]).
2. Create a single-element cluster for every output distribution of every HMM.
3. **Merge**: Find the most similar pair of clusters, and combine them.
4. **Shift**: For each element in each cluster of the current configuration, move it to another cluster if that results in an improvement. Repeat this shifting until no improvement can be made.
5. **Repeat**: Go to step 3 unless some convergence criterion is met.

Figure 3.5: The algorithm for output-distribution (state) clustering.

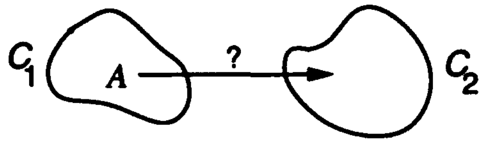
Step 1 is necessary in order to obtain statistics for measuring the distance between Markov states. Step 4 is a heuristic optimization that attempts to improve the cluster configuration by allowing elements to be moved from one cluster to another. In consequence, this clustering is not a simple bottom-up tree growing algorithm. Without Step 4, this would be a simple greedy algorithm, where every merge would be permanent and tree-structured merges would be formed in a bottom-up fashion.

At Step 4, a new configuration is considered to be an improvement if its *grand weighted entropy* is less than the existing configuration's. The grand weighted entropy of a configuration is the summation of the entropies (weighted by counts) of all the clusters in the configuration. To reduce computation at Step 4, shifting can be terminated prematurely when the number of movements exceeds a predefined threshold, or when a minimum improvement is not achieved.

During Step 4, after an element is moved to another cluster, recomputation of the entropies for the two affected clusters is expensive. Therefore, unlikely movements should not be considered and poorly trained distributions are not worthy of re-configuration. To detect poor candidates for the shift, we, as a heuristic, utilize the entropy-increase matrix of all pairs of the distributions in the input HMMs:

$$\begin{aligned}\Omega(\mathcal{A},\mathcal{B}) &= \text{entropy increase when distribution } \mathcal{A} \text{ and } \mathcal{B} \text{ are merged,} \\ &= (3.1)\end{aligned}$$

where $\mathcal{A}$ and $\mathcal{B}$ are any two distributions in the input HMMs. Note that $\Omega(\mathcal{A},\mathcal{B})$ is computed only once when Step 3 is activated the first time. While considering whether to move distribution $\mathcal{A}$ from cluster $\mathcal{C}_1$ to cluster $\mathcal{C}_2$, we first compare the *average* entropy-increases for $\mathcal{A}$ before and after its moving to $\mathcal{C}_2$:



$$\begin{aligned}\Omega(\mathcal{A}) &= \left(\sum_{* \in \mathcal{C}_1-\{\mathcal{A}\}} \Omega(\mathcal{A},*)\right) / \left(size(\mathcal{C}_1)-1\right) \\ \Omega'(\mathcal{A}) &= \left(\sum_{* \in \mathcal{C}_2} \Omega(\mathcal{A},*)\right) / size(\mathcal{C}_2)\end{aligned}$$

Only if the ratio of $\Omega'(\mathcal{A})/\Omega(\mathcal{A})$ is less than a predefined threshold do we compute the grand weighted entropy of the new configuration to see if moving $\mathcal{A}$ will lead to an improvement.

The convergence criterion in Step 5 could be the desired number of clusters remaining or a maximum ratio of the new grand weighted entropy to the previous one.

The clustering algorithm provides an excellent compromise between trainability and sensitivity. Given a fixed amount of training data, it is possible to find the largest number of trainable distributions or the smallest number of sensitive distributions.

3.1.3 Phone-Dependency and State-Dependency

Phone-Dependency

While clustering the Markov states in Figure 3.5, we usually disallow cross clustering between different phones as otherwise it can make minimal pairs, such as t$\underline{\underline{e}}$n and t$\underline{\underline{i}}$n, even harder to distinguish. We call this constraint the *phone-dependency* constraint [1]. Notice that because of the Shift action in Figure 3.5, the computation for the clustering algorithm is an exponential function of the number of objects being clustered. With the phone-dependency constraint, the input size is reduced by a constant factor and thus the clustering is speeded up. Moreover, clusterings for different phones can be run in parallel. After all the clusterings for all phones are finished, a final merge sort is run to take the most significant *merge* actions (with minimum entropy increase) and *shift* actions (with maximum entropy reduction) to produce the desired total number of clusters for the entire system [2].

State Dependency

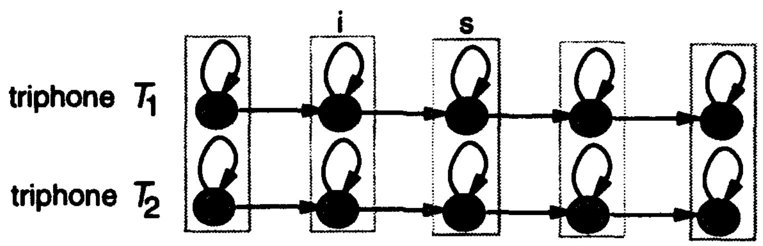
As we will see in our experiments in Section 5.1, a large percentage of the state members in the same cluster are from the same state location. This suggests us to enforce the *state dependency* constraint, prohibiting clustering across different state locations, for speeding the process of clustering. As the vocabulary size grows, the number of triphones needed for each phone increases accordingly, so is the total number of Markov states to be clustered. Suppose that we represent spoken English using p phones ($p \approx 50$ usually), and there are n triphones in total across all phones for a certain task and each triphone has d Markov states. On average, there are n/p triphones per phone. The combined phone dependency and state-dependency constraints reduce the input size to the clustering algorithm from nd to n/p on average; but pd independent runs of clustering have to be issued. That is, the computation cost drops from $\exp(nd)$ to $pd\exp(n/p)$. Again a final merge sort has to be run to produce the desired total number of clusters for the entire system. Experiments in Section 5.1 show that without the state-dependency constraint, more than 90% of the elements in the same cluster are indeed from the same state location in the model topology. Therefore, when we are faced with very large vocabulary tasks, the state-dependency constraint is usually adopted.

Neighboring-State Information

When the state-dependency constraint is used, the clusterings for different state locations are independent. To relate the clusterings across different state locations, we propose to incorporate *neighboring-state* information while measuring the distance between two output distributions at the same state location:

[1] It is also possible to merge distributions across different phonetic models, although experiments on speaker adaptation indicated this did not make any significant difference [45].

[2] The best data structure to use is a heap, with one element for the list of actions of each phone.



$$\Omega^*(T_1(s), T_2(s)) \;=\; \sum_i w_{s,i}\, \Omega(T_1(i), T_2(i)) \tag{3.2}$$

where $T_1(s)$ represents the output distribution associated with the s-th state of triphone T_1 and $\Omega()$ is defined in (3.1). $w_{s,i}$ is a weighting function and is inversely proportional to the topological distance between the state under consideration (s) and all other states (i). That is, the closer a state i is to state s, the more important the information it provides to state s is. We call the distance measure defined in (3.2) ***entropy increase with neighboring-state information.*** To make the entropy increase comparable across different states of different phones without bias, $\sum_i w_{s,i} = 1.0$ for every state s. Note that in order to compute the distance measure with neighboring-state information, all the output distributions inside a phonetic model have to be available, not just the output distribution under consideration. This also imp'ies that when the two s-th states of two triphones are merged, the other pairs of states must also be merged in order to compute future distances for further clustering actions on state s.

Section 5.2 presents the results (7% – 9% word error rate) of applying the SDM with both the phone-dependency and state-dependency constraints to the 5,000-word Wall Street Journal task.

3.1.4 Flexible Initialization for SDM Training and Interpolation

After the state-clustering procedure has terminated, we obtain a mapping table consisting of clusters of similar Markov states. Each cluster is represented by a *generalized* output distribution called a senone. States in the same cluster share the resulted senone as their output distribution. Table 3.1 shows a partial list of a senone mapping table that is generated with the phone-dependency constraint only. For example, the 11-th and 12-th rows show that both the first state (state 0) and the second state of triphone AA(B,Z) are mapped to the ninth AA senone.

For each state s of each triphone model $\mathcal{T}$, the mapping table $\mathcal{M}$ maps it to a senone $\mathcal{M}(\mathcal{T}, s)$. The set of triphone models that share senones as their output distributions is called

triphone	state id	senone id
AA(F,R)	0	1
AA(B,R)	0	1
AA(G,R)	0	2
AA(DD,R)b	0	2
...	...	...
AA(P,R)	2	8
AA(K,R)	2	8
AA(R,R)b	0	9
AA(AXR,R)b	0	9
AA(N,K)	1	9
AA(B,Z)	0	9
AA(B,Z)	1	9
AA(L,N)	1	9
AA(SIL,B)b	3	10
AA(AXR,B)b	1	10
AA(R,B)	3	10
AA(K,M)	3	10

Table 3.1: Partial list of a senone mapping table.

the shared-distribution model or the SDM. With this sharing in place, we can re-train the triphone model so that the HMM parameters are optimized under the fixed sharing structure.

Initialization

Although we usually start triphone training from context-independent models, the training re-estimation will converge faster if we start from existing context-dependent models. This section will describe a flexible initialization that enables fast training from any model.

Suppose we would like to train the shared distribution model for the triphone set Λ_1 with a senone mapping table $\mathcal{M}_1$ and we are given a set of initial models with a mapping table $\mathcal{M}_2$, possibly for another set of triphones Λ_2. When Λ_2 triphones are SDMs, $\mathcal{M}_2$ is given. When they are not SDMs, it is trivial to build the mapping table $\mathcal{M}_2$ corresponding to the distribution sharing configuration in Λ_2. For example, the generalized-triphone model is a special case of the shared-distribution model, where all the i-th states of all the triphones in the same cluster share the same output distribution or senone. On the other hand, if Λ_2 has no distribution sharing at all, the $\mathcal{M}_2$ simply maps each different Markov state to a distinct senone id. Finally, when only context-independent phone models are given, both Λ_2 and $\mathcal{M}_2$ are just empty files.

To initialize the occurrence count of a senone in $\mathcal{M}_1$, we add together the occurrence counts of all the corresponding senones in $\mathcal{M}_2$. When there is no corresponding triphone in Λ_2, the context-independent phone model is used (we assume that monophone models are always available). This initialization algorithm is described in Figure 3.6.

```
Clear the occurrence counts for all senones in $\mathcal{M}_1$.
For each triphone $\mathcal{T} \in \Lambda_1$
    For each state $s$ of $\mathcal{T}$
        set senone $\phi \leftarrow \mathcal{M}_1(\mathcal{T}, s)$
        if ($\mathcal{T} \in \Lambda_2$) set senone $\psi \leftarrow \mathcal{M}_2(\mathcal{T}, s)$
        otherwise set senone $\psi \leftarrow$ context-independent($\mathcal{T}$,$s$)
        count($\phi$) $\leftarrow$ count($\phi$) + count($\psi$)
```

Figure 3.6: The algorithm for initializing the occurrence counts of senones in mapping table $\mathcal{M}_1$ for triphone set Λ_1, given the mapping table $\mathcal{M}_2$ for triphone set Λ_2.

When the input models are context-independent phones, Λ_2 is empty. From Figure 3.6, we see that the initial value for senone s of phone p becomes the summation of all the output distributions in monophone model p, with each output distribution weighted by the number of component members. For example, Table 3.2 shows all the members of a particular senone cluster for phone D. Since there are 13 members from state s_1 and 3 members from state s_2, the initial count of this senone is computed as

$$13\ \text{count}(\text{D}, s_1) + 3\ \text{count}(\text{D}, s_2)$$

according to the initialization algorithm in Figure 3.6. Finally, to get the probability entries for a senone, we simply normalize the counts for the senone such that the individual entries sum to 1.

triphone	distribution	triphone	distribution	triphone	distribution
D(S,AA)	state s_1	D(CH,AA)b	state s_1	D(S,AE)	state s_1, s_2
D(S,AO)	state s_1, s_2	D(S,EH)	state s_1	D(S,ER)b	state s_1
D(S,EY)	state s_1	D(S,IY)	state s_1	D(S,IY)b	state s_1
D(S,R)b	state s_1, s_2	D(TS,ER)b	state s_1	D(Z,AX)b	state s_1
D(Z,EY)	state s_1				

Table 3.2: Members of a D senone. There are 13 members from state location s_1 and 3 members from s_2 in this cluster.

Interpolation

To improve the generalization capability, deleted interpolation [72] is used to smooth context-dependent models. Here, we interpolate each senone with the uniform distribution and the context-independent distribution(s) corresponding to the component state(s). For example, the D senone shown in Table 3.2 is interpolated twice in parallel. One copy is interpolated with the output distribution associated with state s_1 of monophone model D and the uniform distribution, and the other copy is interpolated with state s_2 of monophone model D and the uniform distribution. To get the final smoothed senone, these smoothed copies are averaged, weighted by the state frequencies in the senone. Figure 3.7 explains the smoothing of the D senone shown in Table 3.2.

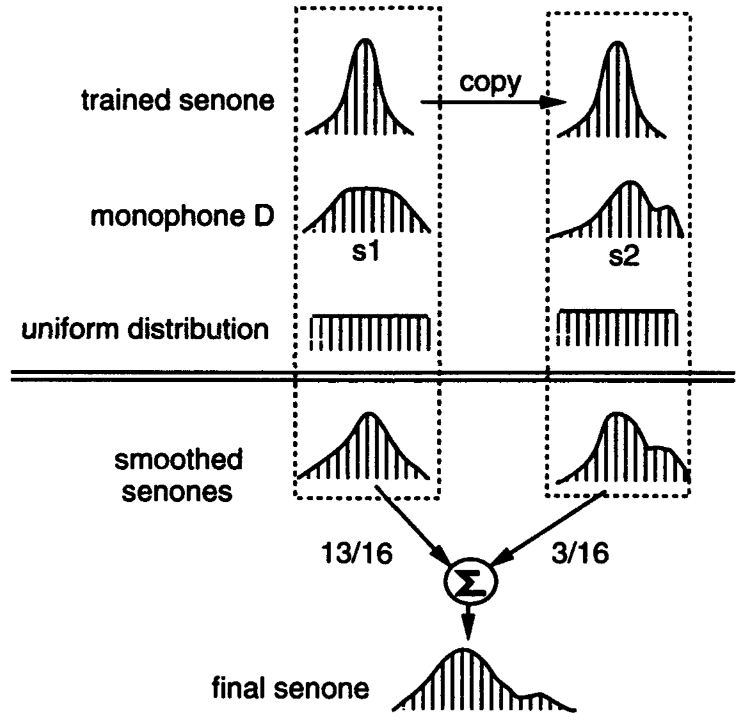


Figure 3.7: The smoothing of the D senone shown in Table 3.1. It contains only state s_1 and s_2. Therefore it is interpolated with the s_1 and s_2 output distributions of the monophone model.

3.2 Decision-Tree Based Senones

The number of triphones needed for a large vocabulary task is usually very large, especially when position-dependent between-word triphones [62, 83] are considered. It's essentially impossible to cover all the triphones needed even with a fairly large amount of training data.

Therefore, there are often many unseen triphones for large vocabulary tasks. Finding a good representation for unseen triphones becomes increasingly important as we scale up to larger tasks.

When unseen triphones occur, most systems simply use context-independent phone models to represent unseen triphones. Due to the inaccuracy of blindly mixing all contexts in monophone models, performance degrades as the number of unseen triphones grows. Biphones may be used to replace monophone models in order to distinguish some contextual effects. However, biphones still do not contain as much detailed information as triphone models. Moreover, the number of parameters is increased slightly by introducing biphone models into the system. In the bottom-up clustering algorithm (Figure 3.5) (either state clustering or triphone clustering), it is not clear which clusters the states of a new triphone should belong to. Therefore, both the generalized-triphone system and the SDM system use context-independent phone models to represent unseen triphones.

Representing a triphone with either monophone or biphone model introduces the problem of over-generalization. To alleviate the over-generalization problem, *c*lassification *a*nd *r*egression *t*rees (CARTs) have been used in many research areas [24, 120, 11]. Some researchers [88, 43, 12] used decision-tree based generalized triphones to provide better models for unseen triphones. In these systems, a decision tree was built to classify all the allophones derived from the same phone. The root of a decision tree contained all of the allophones for a single base phone. Each node in the tree had a binary question for the component triphones about their contexts (e.g., "*Is the right phoneme a back vowel?*"). The binary question split the component allophones into two classes, each corresponding to a child node. At the end, each leaf of the tree represented an equivalence class (i.e., a generalized allophone) consisting of allophones in similar contexts. To find the generalized allophone for any triphone, either seen or unseen in the training data, the tree corresponding to the phone was traversed from the root, by answering the question attached to each node until a leaf node was reached. The generalized allophone represented by the leaf was then used to model the given allophone. For example, Figure 3.8 illustrates the classification tree for /K/ allophones [3] and highlights the path which triphone `K(L,AX)` traverses. Here the phone-dependency constraint is applied. The question "Right = back vowel?" asks if the right phone of the considered triphone a back vowel. It shows that triphone `K(L,AX)` is mapped to the fourth `K` generalized-allophone, according to the linguistic questions associated with the tree.

To incorporate the merits of both the SDM and the classification tree, we construct decision-tree based senones. That is, we use decision trees to classify the Markov states of allophone models instead of the entire allophone. The decision-tree based senone will inherit the merit of the shared-distribution model where parameter sharing is carried out at a subphonetic level.

[3]Revised from [42].

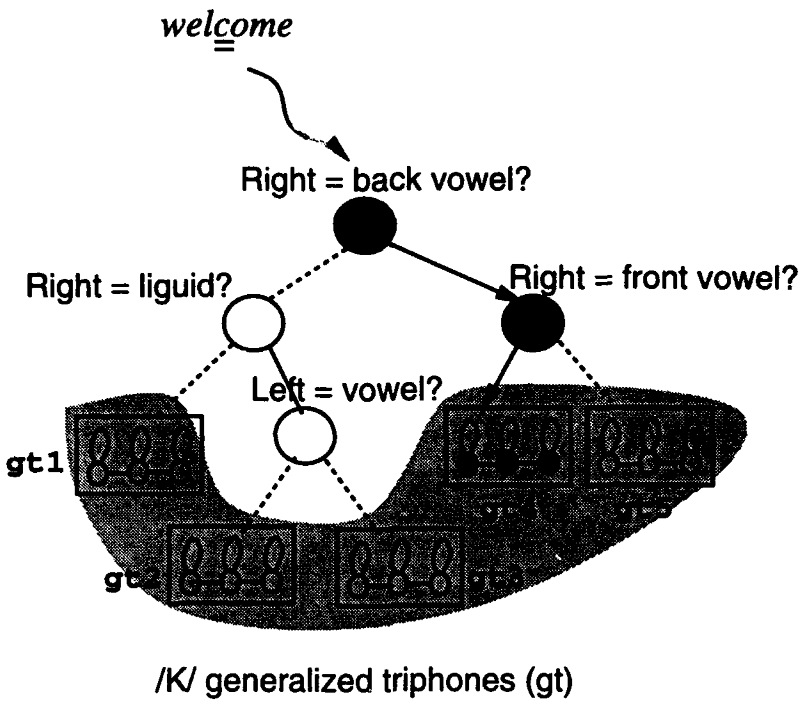


Figure 3.8: The decision tree for classifying K-allophones. Triphone `K(L,AX)` is mapped to the fourth K generalized-allophone.

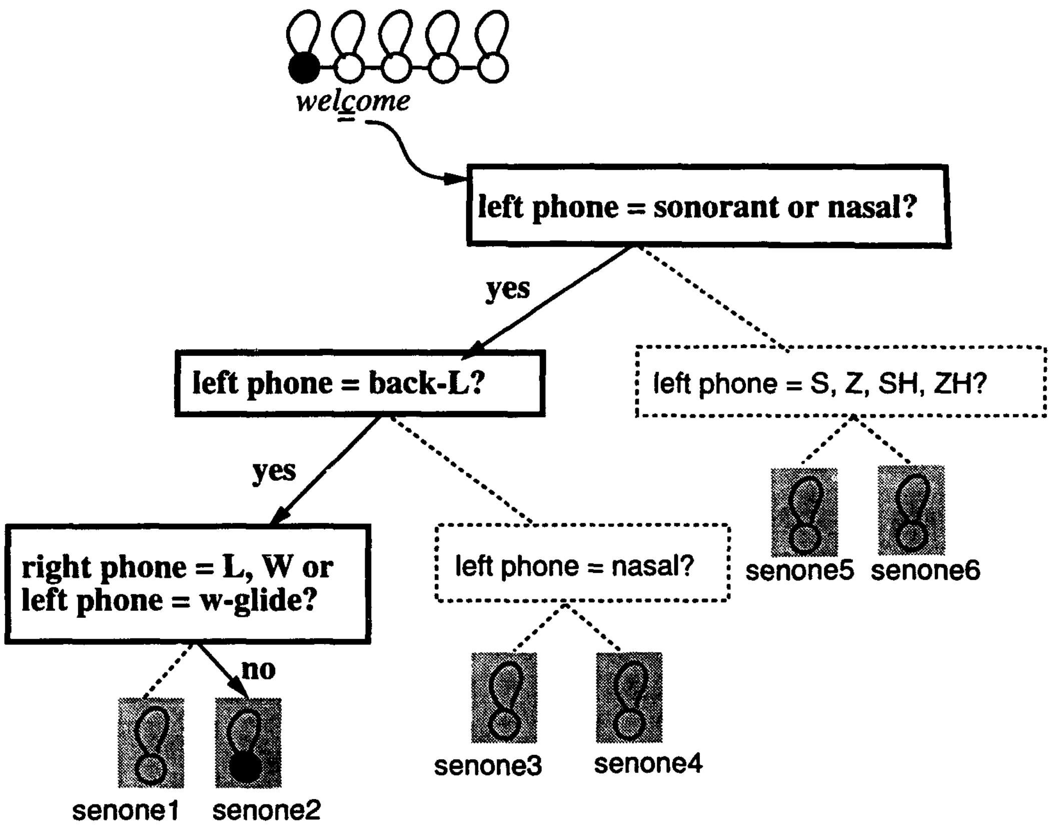
In addition, it offers detailed parameters for unseen triphones. Unseen triphones, like seen triphones, also traverse the trees to find appropriate senones as their output-distribution representation. In the experiments in Section 5.3, we will show that by modeling unseen triphones with decision-tree based senones, we can reduce the word error rates by 10 – 20% (depending on the number of unseen triphones contained in each test utterance), compared with the situation in which unseen triphones are represented by monophones.

Even for utterances that contain no unseen triphones, we believe the improved modeling for unseen triphones helps the decoder prune those wrong paths in which unseen triphones are needed.

3.2.1 Construction of the Senonic Decision Tree

To predict unseen triphones with senones, we extend the principle of distribution sharing to decision trees [43]. The algorithm we present here is constrained by both the phone dependency and state dependency described in Section 3.1.3. It is easy to modify the algorithm to relax these two constraints.

We *automatically* build a decision tree for each Markov state of each phone. We use 50 phones in SPHINX-II for English. With each phonetic model relying on the 5-state Bakis topology [131, 70, 18], the result is a set of 250 decision trees. A decision tree is a binary tree, with a (possibly composite) linguistic question associated with each node, asking about the category of the left- or right-phone context of a triphone. Triphones with similar contexts thus follow the same traversing path and fall into the same leaf. Each leaf of the senonic decision tree is a cluster of similar Markov states, whose average forms the senone at the leaf. To find the senone to be associated with a certain state of a given triphone, either seen or unseen in the training data, we traverse the corresponding tree until a leaf node is reached. The senone associated with the located leaf is then used to represent the output distribution associated with the given Markov state. Figure 3.9 illustrates an artificial example tree to classify the first state of $/K/$-triphones. It also shows that the first state of triphone model K(L,AX) is associated with the second senone among the senones for the first state of all K-triphones.



welcome
left phone = sonorant or nasal?
yes
left phone = back-L?
left phone = S, Z, SH, ZH?
yes
right phone = L, W or left phone = w-glide?
left phone = nasal?
no
senone1
senone2
senone3
senone4
senone5
senone6

Figure 3.9: An artificial example for the decision tree that classifies the first state of $/K/$-triphones. There are 6 senones in total for the first state of /K/-triphones. The first state of the /K/-triphone in "wel$\underline{\underline{c}}$ome" is mapped to the second senone.

Figure 3.10 summarizes the construction of the decision trees [42].

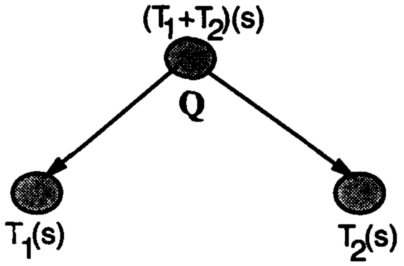
At Step 1, a linguist creates manually a set of categorical questions about a triphone's left

1. Locate a set of appropriate linguistic questions manually.
2. For each phoneme p,
   - Estimate all p-triphone HMMs (e.g. using the forward-backward algorithm).
   - For each Markov state s in the model topology, classify all the s-th Markov states in all p-triphones.
     (a) Create a root consisting of all the s-th states of all p-triphones.
     (b) Find (automatically) the best *composite* question for the root. (to be explained in Figure 3.11.)
     (c) **Best-First Expansion:** Select the node with the overall *best question* among the current leaves, and split the node according to the question associated with it.
     (d) For each of the two newly created leaves at step (c), find (automatically) the best *composite* question. (to be explained in Figure 3.11.)
     (e) Go to step (c) unless some convergence criterion is met.

Figure 3.10: The algorithm for the top-down Markov state classification, using binary decision trees. Note both phone-dependency and state-dependency constraints are used here.

or right context [134]. These questions are designed to capture linguistic classes of contextual effects. For example the deletable stops { BD DD GD PD TD KD } are captured in one single categorical question like "is the left/right context a deletable stop?". A complete list of the linguistic questions we used in this dissertation is shown in Appendix D. It is the same as the one used in [42].

The overall classification structure in Figure 3.10 is the same as for the tree-based generalized allophone in [42]. However, two basic differences should be noted. First of all, the classification is now on subphonetic units (i.e., the Markov state) rather than on the entire HMM. One decision tree is built for each state of each phone. Secondly, the goodness of a question at a node is defined as the entropy reduction with neighboring-state information as (3.2) shows, with $\mathcal{T}_1$ and $\mathcal{T}_2$ representing the child nodes. To re-phrase, suppose a tree for the s-th state of a certain phone is under construction and suppose question Q splits the elements of node $(\mathcal{T}_1 + \mathcal{T}_2)$ into node $\mathcal{T}_1$ and $\mathcal{T}_2$:



The goodness of question Q with respect to state s at node $\mathcal{T}_1 + \mathcal{T}_2$ is defined to be a weighted summation of the weighted cross entropies of all pair-wise states:

$$\begin{aligned} \text{goodness}(Q, s) &= \Omega^*(\mathcal{T}_1(s), \mathcal{T}_2(s)) \\ &= \sum_i w_{s,i}\, \Omega(\mathcal{T}_1(i), \mathcal{T}_2(i)) \end{aligned}$$

In other words, the more entropy reduction a question renders, the better the question is. Recall that to utilize neighboring-state information, all the output distributions inside a triphone model have to be available at the time of the entropy measuring computation. Because we need the HMM statistics for measuring the goodness of a question, we need to obtain an estimate of the triphone model at the beginning of Step 2 in Figure 3.10.

3.2.2 Composite Questions

One disadvantage of the tree-based top-down classification is that once a split is made, individual elements cannot be moved to new positions in the tree. This may result in a sub-optimal clustering configuration. In particular, data fragmentation often occurs in which similar elements fall into different leaves. To alleviate the problem of data fragmentation, we have to build either complex composite questions [43] or decision *networks* where two siblings and/or siblings of ancestors can be merged [14]. Therefore, at Step 2(b) and 2(d) of Figure 3.10, we build composite questions, formed by conjunctions, disjunctions and/or negations of the simple linguistic questions, to alleviate the data fragmentation problem. Even with composite questions, the decision-tree clustering still has less freedom in re-configuration than the bottom-up clustering for the SDM does. This will be evidenced in the experiments in Chapter 5.

Figure 3.11 shows the algorithm to find the best composite question for a given node. The tree-growing step (Step 1) is similar to the one in Figure 3.10, with the exception that only simple questions are considered.

1. Create a tree of "simple" questions under the given node.
    - (a) Create a root containing all the elements in the given node.
    - (b) Find the best question for the root, among the hand-crafted linguistic questions created at Step 1 of Figure 3.10.
    - (c) **Best-first expansion:** Split the node with the overall best question among the current leaves.
    - (d) For the two new leaves created at step (c), find the best question among the hand-crafted linguistic questions created at Step 1 of Figure 3.10.
    - (e) Goto Step (c) until some convergence criterion is met.
2. For the final tree created by Step 1, cluster the leaves into two groups. Try all possible combinations and choose the one with minimum total entropy.
3. Form the composite question for the given node as the disjunction of all the paths to one of the groups in the configuration decided at Step 2. The question for any path leading to a leaf is the conjunction of all the questions along the path.

Figure 3.11: The construction of the best composite question for a given node. A composite question is represented as a tree of simple questions.

At step 1(b) and 1(d), the number of simple questions that have to be considered is finite and usually small. For example, using the question set in [42], only 89 questions are considered (43 categorical questions about the left or right context, plus 3 questions about a triphone's position in a word). Figure 3.12 illustrates an example of forming the composite question, given a tree of simple questions with clustered leaves. The composite question leading to the black leaves

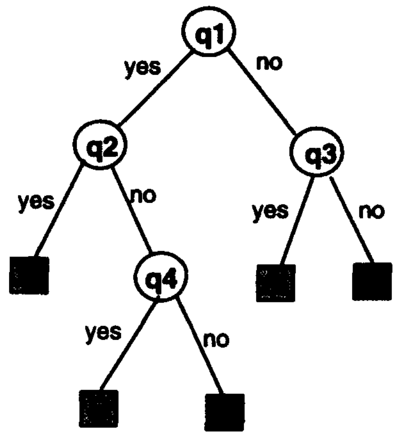


Figure 3.12: A composite question, $q_1\bar{q}_2\bar{q}_4 + \bar{q}_1\bar{q}_3$, for the root. q_i is the best simple question at each node. Black leaves satisfy the composite question. Shaded leaves do not.

is

$$q_1\bar{q}_2\bar{q}_4 + \bar{q}_1\bar{q}_3$$

where the plus sign stands for disjunction, concatenation for conjunction, and the bar sign for negation.

The computation for grouping leaves into two clusters is an exponential function of the number of leaves in the "simple" tree [4]. The number of leaves under a "simple" tree is much smaller than the entire number of elements at the "composite" root, which the bottom-up distribution clustering always considers at the shift step of Figure 3.5. Our empirical results reflect this predicted cost savings, as the decision-tree clustering runs much faster than the bottom-up agglomerative clustering.

Given a configuration created by the bottom-up agglomerative clustering, it is possible to find a decision tree with each leaf equivalent to each cluster, as long as we allow the complexity of the composite questions to be unlimited and as long as there are a sufficient number of detailed simple questions (e.g., each categorical question contains only one phone). At that extreme, the decision-tree clustering converges to the bottom-up agglomerative clustering and there will not be any cost savings.

[4]We usually set the depth of the "simple" tree to be no greater than 3. Therefore, the computation of the grouping is only a constant factor of at most 2^7.

3.2.3 Cross Validation

Cross validation, like deleted interpolation [72], is a technique to make an established system (e.g., the decision trees) more robust against new testing conditions (e.g., a new task). The basic idea is to divide the entire available data into two sets. For each data set, we train the set of triphone HMMs represented in the data. We then use one set of trained HMMs to build the decision trees, and the other to validate the trees. This validation keeps the tree structure and the composite questions fixed, but replaces the entropy reduction (i.e., the goodness measure) by applying the same question to the the second set of triphone HMMs. It may also eliminate the splits that have insignificant entropy reduction computed by the second set of models. That is, the second set of HMMs is used to validate the goodness of the questions computed by the first set.

Cross validation is important to build a robust vocabulary-independent (VI) system [42] because a vocabulary-independent training corpus usually does not cover a good percentage (e.g., 95%) of the triphones needed for any specific task. The work in [42] used two thirds of the VI training corpus to build the VI decision trees. The remaining third of the VI training corpus served as the source of a new triphone set for a new task. The set of triphone models estimated by the rest of the VI training data was used to validate the goodness measures computed by the first two thirds of the data.

Cross validation becomes less important when a vocabulary-independent system is not pursued or when almost all the tree leaves are kept for the final system (i.e., almost all splits are preserved). Particularly in our experiments on the Wall Street Journal task using a vocabulary-dependent training corpus, cross validation offered little advantage.

3.2.4 The Senone Mapping Table Using Decision Trees

After the decision trees for all the state locations of all phones are constructed [5], we can prune, from leaves to the root, the worst splits (i.e., the nodes with least entropy reduction) to get the desired total number of leaves (senones) [6]. The desired total number of senones depends on the available training data (and possibly memory capacity of the computer hardware). Given a fixed amount of training data, it is possible to find the largest number of trainable senones or the smallest number of sensitive senones.

Whenever a triphone (either seen or unseen in the training data) is needed, we traverse the

[5]Or wait until the cross validation is finished.

[6]Again, a heap is the best data structure for this pruning.

corresponding pruned d trees to find the associated senones, assuming there are d states per HMM. This way, both seen and unseen triphones are modeled by senones in a unified fashion. As shown in Section 5.3, modeling unseen triphones with senones reduces the word error rate by more than 10%, compared with the situation in which unseen triphones are represented by monophones.

3.3 Quantization of Markov States

3.3.1 Predicting Unseen Triphones

The senonic decision-tree described in the previous section is ideal for unseen-triphone modeling. However, the drawback of decision-tree based classification is the limited capability for re-configuration. Once a node is split, the elements in a child stay in that subtree forever. Even with the help of composite questions, the re-configuration capability is still not as powerful as the shift step in the bottom-up clustering algorithm. Therefore, the decision-tree classification often yields a sub-optimal configuration. The pros and cons of the bottom-up clustering versus top-down classification can be summarized as follows:

state clustering	quality for seen triphones	quality for unseen triphones
bottom-up	nearly optimal	monophones
top-down classification	sub-optimal	decision-tree based senones

For seen triphones, we may want to use the best configuration the bottom-up clustering algorithm generates. This suggests the use of a hybrid approach in which seen triphones obtain the best sharing structure, while unseen triphones carry a sub-optimal sharing configuration (but still better than context-independent phones).

The hybrid approach we propose here can be illustrated in Figure 3.13.

To elaborate,

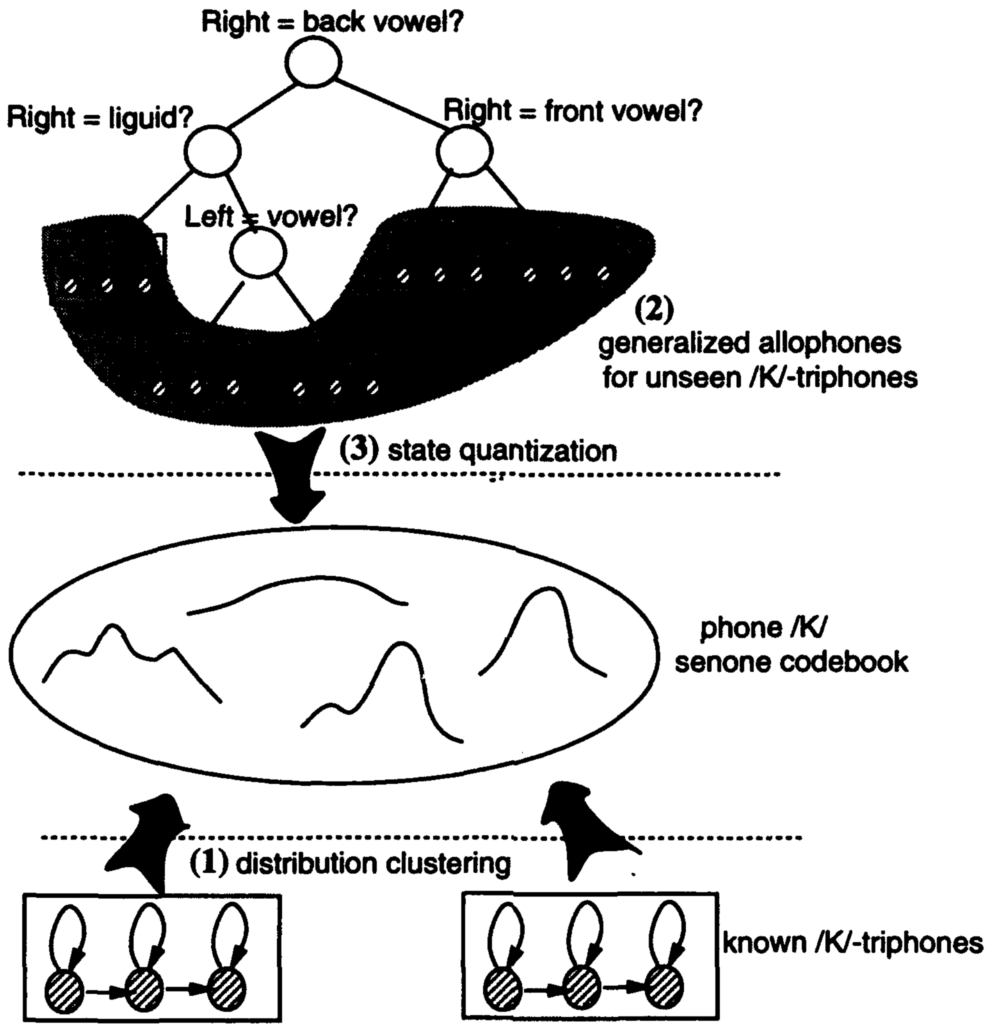


Figure 3.13: A hybrid system of senone sharing for modeling unseen triphones. The bottom-up distribution clustering and the top-down triphone classification are linked through state quantization. Seen triphones share senones according to the distribution clustering result. An unseen triphone traverses the decision-tree to locate the generalized allophone it belongs to, then shares senones according to the state quantization result.

1. Using the seen triphone models, run the bottom-up *state* clustering and then train the shared-distribution model.
2. Using the seen triphone models, run the top-down *triphone* classification using decision trees and then train the generalized allophone models.
3. To avoid doubling the number of system parameters, *quantize* the Markov states of the generalized allophone model trained at step 2 with the senones generated at step 1.

At Step 3, state quantization (SQ) means to search for the best-matched senone among the senones shared by the SDM. Conceptually, this searching for the best-matched senone is similar to vector quantization, in which we search for the best-matched vector prototype. This implies that a distance measure has to be defined to measure the distance between a Markov state and a senone. Section 3.3.3 describes three algorithms to compute this distance.

With the hybrid approach, seen triphones share senones according to the bottom-up clustering mapping table. To model an unseen triphone, we first traverse the corresponding tree until a leaf is reached. Then the generalized allophone with quantized states (that is, every state refers to a senone) is used as the model for the unseen triphone. In this fashion, seen triphones use the best sharing structure achieved by the bottom-up clustering algorithm. Unseen triphones share senones through the state quantization process. The sub-optimal configuration caused by the generalized-allophone decision tree, together with possible state-quantization errors, may degrade the modeling for unseen triphones. However, with the help of the linguistic questions in the tree and a good state-quantization algorithm, we believe that the hybrid approach is able to model unseen triphones better than context-independent phone models. This belief is borne out in the experiments in Section 5.4.

To sum up, the advantages of the hybrid approach are:

- Optimal distribution clustering is achieved for seen triphones, which are usually the most frequently used parameters.
- More detailed parameters than monophone models are obtained for unseen triphones.
- The number of system parameters i[illegible]d. Only the senones constituting the SDM need to be maintained.

As the experiments in Section 5.4 demonstrate, the hybrid approach performs as well as the purely decision-tree based senones. Although the SDM indeed provides better models for seen triphones, the mis-average problem associated with generalized allophones (Figure 3.2) and the state-quantization error for unseen triphones are unavoidable. On the other hand, the senonic decision tree is able to improve its structure through composite questions and the linguistic questions are able to capture phenomena that are missed by the HMM statistics due to insufficient training data. The hybrid approach, which is a great deal more complex, performs better than the senonic decision tree only when there are few or no unseen triphones in the test set.

3.3.2 The Senonic Baseform for Pronunciation Learning

Phonetic pronunciation optimization has been considered by several researchers [96, 23, 13]. Most speech recognition systems use a fixed phonetic transcription for each word in the vocabulary. If a word is transcribed improperly, it will be difficult for the system to recognize it. More importantly, some words may be pronounced in several ways because of dialect differences. For example, the word *tomato* could be pronounced either as /T AX M *EY* DX OW/ or /T AX M *AA* DX OW/. In this case, we may use multiple phonetic transcriptions as alternatives. However, alternative phonetic pronunciations are not always possible. For example, it is hard to define alternative phonetic pronunciations for the 26 English letters. More importantly, an intelligent system should be able to learn alternative pronunciations automatically.

The necessity of learning alternative pronunciations also arises for foreign words due to the distinct phoneme sets used in different languages. Coupled with speaker variations this constitutes an even more difficult problem. In this situation purely acoustic driven models are most useful since the phoneme notation of the local language is discarded. Senones can be applied to construct purely acoustic driven models for learning alternative pronunciations, using the technique of state quantization.

Since the senones created by the distribution clustering procedure in Section 3.1.2 are representative output distributions, we can treat senones as models for the state prototypes in the Markov-state space, just as fenones are models for the VQ prototypes in the signal vector space. In fenonic baseforms, words are spelled as a sequence of fenones. A word is treated as a walk through the VQ prototypes. Similarly we can construct senonic word baseforms, treating a word as a walk through the Markov state prototypes. Each senone describes a very short distinct acoustic event (shorter than a phoneme). Each different permutation of senones constitute the model for a different sound. Figure 3.14 shows the concept of senone sharing or distribution sharing across all possible models (phones, multi-phones, words, etc.).

In this section, we will present the algorithm for finding the senonic baseform for a word,

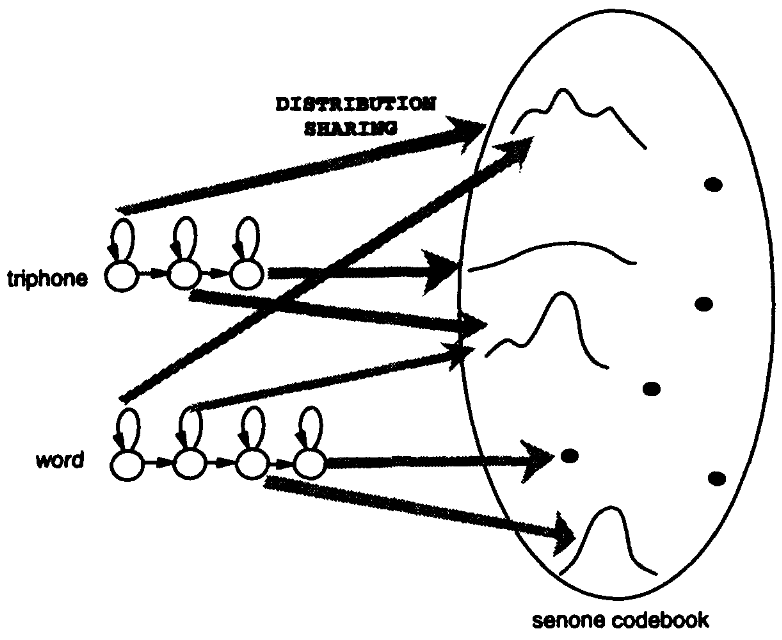


Figure 3.14: The concept of senone sharing across all hidden Markov models, such as triphones, multi-phones, words, or even phrase models.

given utterance examples. The senonic baseform we show here is purely acoustic driven, and therefore is very suitable for speaker adaptation, specialized jargon, and acronyms (e.g., *pc-store*, pronounced as 'pee-see-store') [13].

Construction of the Senonic Baseform

Given multiple utterances of a particular word, we can estimate a word HMM by the forward-backward algorithm. For each state in the estimated word HMM, a most similar senone is identified to replace the estimated output distribution. This concept of state quantization is the same as the one in the hybrid approach, except that the SQ distance measure may be different. Figure 3.15 shows the algorithm for constructing the senonic baseform for a word, given multiple utterances. Here we assume that a senone codebook has already been created. The word model is allowed to be labeled by an arbitrary sequence of senones to provide the maximum freedom for automatically learned pronunciations. Section 3.3.3 describes three distance measures for the state quantization process at Step 4 of Figure 3.15.

The senonic baseform is applied to two tasks in Section 5.5. The vocabulary for the first task contains only the 26 letters of the English alphabet. With a rich training corpus, each letter has ample training data and thus two senonic baseforms per letter are automatically learned in our experiments. With these purely acoustic-driven senonic baseforms, the word error rate is

1. Compute the average duration (number of time-frames) of the word, based on the given multiple utterance samples.
2. Build a Bakis word HMM with the number of states equal to a portion of the average duration.
3. Run several iterations of the forward-backward algorithm on the word model, using the given utterance tokens.
4. **[SQ step]** Quantize each state of the estimated word model with the senone codebook.

Figure 3.15: The determination of the senonic baseform for a word, given multiple training utterances.

reduced by 15% compared with the phonetic baseform [67].

The second task attempts to attack the problem of learning a good senonic baseform in large-vocabulary tasks when there are few training tokens. Unfortunately, with few training tokens, the proposed SQ algorithm in Section 3.3.3 does not yield a highly-accurate senonic baseform, due to the high acoustic confusability implied in the vocabulary inventory. Interestingly, we find that the senonic baseform is indeed able to adapt to speakers, even with very few training samples. For example, the word *arctic* was transcribed as /`AA R KD T IX KD`/ in the lexicon while a speaker pronounced it as /`AA R T IX KD`/. The missed /KD/ phoneme caused many errors made by the phonetically based system, but the senonic baseform was able to recognize the word easily.

Comparison with the Fenonic Baseform

In comparison with the 200 VQ-dependent fenones, the senone codebook has better acoustic resolution, as there are usually a few thousand senones in a system. In addition, when training examples are unavailable, phonetic baseforms can be used together with the senonic baseforms for other words without any increase in the system complexity, since both phonetic and senonic baseforms share the same set of senones. Each senonic word model, built by the algorithm described in Figure 3.15, generally has more states than the traditional phone-concatenated word model and hence is capable of modeling more acoustic details. Although it usually has more states, the senonic word model never increases the number of free parameters in the system.

To determine a senonic baseform, we search for the sequence of senones in the Markov state space that best describes a word. This is similar to the fenonic baseform in which the best sequence of fenones in the vector space is searched. However, two dissimilarities are noted. First of all, senones are state based, while fenones are time-frame based. Secondly, the senonic baseform is solved as a word-HMM training and then state quantization problem, while the fenonic baseform is solved as a decoding problem (see Section 2.1).

Further work, such as incorporating spelling information, should be done to improve the senonic baseform in order to take advantage of the adaptation capability hinted at in the *arctic* example. Another improvement would be to incorporate senone bigram models or other higher-level models in the state quantization procedure, or even to use a decoding scheme like the fenonic baseform with multiple training samples.

3.3.3 Distance Measure for Markov State Quantization

Both the hybrid approach for modeling unseen triphones and the senonic baseform for learning word pronunciations rely on the concept of Markov state quantization. The goal is to find the best-matched senone, given the information associated with a Markov state of an estimated model. The estimated model can be the generalized allophone model as in the hybrid approach, or it can be any other model such as the estimated word HMM in the senonic baseform work.

What is a good distance measure between a Markov state and a senone? The problem of state quantization is different from the problem of state clustering. State clustering not only ties similar Markov states, but also merges infrequent states before merging frequent ones to make each shared distribution more trainable. A re-training on the shared distribution model is followed after the clustering procedure. For state quantization, no merge of two distributions is involved. State quantization is simply a state labeling procedure, like the vector quantization procedure. Once the given model is labeled by senones, the original senone values can be used directly, or be re-adjusted by running another round of training.

Euclidean Distance

When there are ample training tokens, the output distribution of the trained model is robust and thus can represent the state sufficiently. Therefore the problem of measuring the distance between a Markov state and a senone becomes the problem of measuring the distance between two probability distributions. The simplest distance measure between two given discrete probability distributions, $\mathcal{P}$ and $\mathcal{Q}$, is the Euclidean distance.

$$\text{distance}(\mathcal{P}, \mathcal{Q}) = \sqrt{\sum_i (p_i - q_i)^2}$$

where p_i is the *probability* entry in distribution $\mathcal{P}$; similarly for $\mathcal{Q}$. Other approaches like Bhattacharyya distance [76] could also be used.

As mentioned earlier, there are ample training samples for each English letter in the spelling task. Therefore, the Euclidean distance will be used in the state quantization process to learn the senonic baseforms for the 26 English letters in Section 5.5.

Cross Entropy

A probability distribution is not a meaningless multi-dimensional vector. It measures the uncertainty of the occurrence of an event. Therefore, cross entropy for measuring distribution divergence is a good choice for the distance measure between two probability distributions.

$$\begin{aligned} \text{distance}(\mathcal{P}, \mathcal{Q}) &= 1 \cdot \chi(\mathcal{P}, \mathcal{P} + \mathcal{Q}) + 1 \cdot \chi(\mathcal{Q}, \mathcal{P} + \mathcal{Q}) \\ &= \sum_i p_i \log \frac{p_i}{(p_i + q_i)/2} + \sum_i q_i \log \frac{q_i}{(p_i + q_i)/2} \end{aligned}$$

Unlike (3.1), cross entropy need not be weighted by the occurrence count here, since the problem of over-sized clusters illustrated in Figure 3.4 does not occur because distributions are not merged. The goal is simply to quantize the target output distribution (in the estimated model) with one of the source distributions (i.e., senones). No further merge is imposed. Note that in this case the entry of $\mathcal{P} + \mathcal{Q}$ is the average probability of the corresponding entries in $\mathcal{P}$ and $\mathcal{Q}$. That is, $\sum_i p_i = 1$, $\sum_i q_i = 1$ and each entry in $\mathcal{P} + \mathcal{Q}$ is $(p_i + q_i)/2$.

As we will see in Section 5.4, this is the SQ measure we use for the hybrid approach for modeling unseen triphones.

The Semi-Continuous Output Probability

The two distance measures mentioned above utilize the discrete output distribution exclusively. The VQ probability density functions for modeling the cepstral codewords are ignored and thus the advantages of the semi-continuous HMMs are not fully incorporated.

To take advantage of semi-continuous HMMs, we align the training utterances with the estimated model by the Viterbi algorithm [132, 123]. This results in a set of data frames associated with each state in the model. These frames describe what data that particular state is likely to emit. Each state in the word model is quantized to the senone that has the maximum probability of generating the aligned data frames. The probability of generating a set of data frames $\mathbf{X} = \{ \mathbf{x} \}$, given a state associated with senone $\mathcal{S}$ is computed as:

$$Pr(\mathbf{X}|\mathcal{S}) = \prod_{\mathbf{x} \in \mathbf{X}} Pr(\mathbf{x}|\mathcal{S}) = \prod_{\mathbf{x} \in \mathbf{X}} \sum_{k} f_k(\mathbf{x})\mathcal{S}(k)$$

where f_k is the VQ density function for codeword k in SCHMMs, $\mathcal{S}(k)$ is the kth entry in the given senone, and each data frame is assumed to be independent of each other.

This SQ approach is most suitable when the output distributions in the estimated model are sparse due to limited training examples. We will see an example use of this distance measure in Section 5.5. When there are sufficient training samples, using output distributions to represent states is more compact and efficient.

3.4 Relationship Between CHMMs and SCHMMs with Senone-Dependent VQ Codebooks

As indicated in Appendix A, there are three ways to compute the probability, $Pr(\mathbf{x}|ij)$, of outputting a data frame $\mathbf{x}$, given a transition from state i to state j in an HMM, according to different representations of the output distribution:

$$\begin{aligned}
Pr(\mathbf{x}|ij) &= b_{ij}(k) \quad \text{for DHMMs if VQ}(\mathbf{x}) = \text{k} \\
Pr(\mathbf{x}|ij) &= \sum_{m=1}^{M} w_m\ f_m(\mathbf{x}) \quad \text{for CHMMs} && (3.3) \\
Pr(\mathbf{x}|ij) &= \sum_{k \in \eta} b_{ij}(k) f_k(\mathbf{x}) \quad \text{for SCHMMs} && (3.4)
\end{aligned}$$

where η is the set of top few best-matched codewords for $\mathbf{x}$

where the density function f_k is usually assumed to be a Gaussian density with a mean μ_k and a diagonal covariance matrix Σ_k. The dimensionality of μ_k and Σ_k is the dimensionality of the data frame $\mathbf{x}$. Table 3.4 compares the pros and cons of these three types of HMMs, where n is the number of triphones in a system and each triphone has d (typically 3 or 5) distinct output distributions. In a 20,000-word task, n is on the order of 20,000. If infrequent triphones (e.g. frequency < 20) are discarded, the number of triphones is still in the range of 1000 — 2000. When the shared distribution model is used, typical numbers of senones for the same task range from 6,000 — 10,000. For M, the number of mixtures per state, common values are

10, 20, or 32. The VQ level, L, is often 256. Typical sizes for η range from 4 to 8. Since the transition probability is not crucial in terms of recognition performance, we can simply set it to be context-independent and thus maintain the number of parameters for transition probabilities to be a constant. In view of this, Table 3.4 does not count the transition probability when the number of free parameters is considered.

	DHMM	CHMM	SCHMM
number of parameters (Typical values)	ndL $(2000 * 3 * 256 = 1.5\text{M})$	$nd(M + M + M)$ $(2000 * 3 * (32 + 32 + 32) = 0.6\text{M})$	$ndL + L + L$ (1.5M)
$Pr(\mathbf{x}\|ij)$ computation during decoding	0 (memory access only)	ndM density-computation + ndM multiplication + $nd(M-1)$ addition	L density-computation + $\|\eta\|$ multiplication + $(\|\eta\|-1)$ addition
VQ error	inevitable	no	minor
distribution assumption	no	pre-assumed	no

Table 3.3: Comparison of DHMMs, CHMMs, and SCHMMs.

Despite of the many advantages of DHMMs, the most serious problem with DHMMs is the VQ error. SCHMMs relieve quantization errors by choosing the top few codewords for each data frame instead of just the top one. On the other hand, CHMMs assume the parametric form (e.g. the Gaussian form) of the output distribution in order to be able to estimate the parameters. The problem about the parametric-form assumption becomes less serious as the model becomes more detailed or the number of mixtures for each state becomes larger. However, another serious problem associated with CHMMs is the heavy floating-point computation. SCHMMs avoid the expensive computation in CHMMs by constraining all the Markov states in a system to refer to the same set of densities. From the viewpoint of CHMMs, the discrete output probability $b_{ij}(k)$ in (3.4) corresponds to the weighting coefficient w_m in (3.3) [44, 51, 22].

One way to approach the performance of CHMMs without actually going to CHMMs is to relax the tying condition in SCHMMs. Currently, the VQ probability densities are tied across all the Markov states in a system. When the tying is completely relaxed, that is, when each state has its own set of VQ probability densities (i.e., mixtures in CHMM terminology), the system becomes a CHMM. Any other intermediate state-tying condition should in principle generate a performance in between the two extreme cases (SCHMMs and CHMMs), as long as there is enough training data.

As an example, consider SCHMMs with senone-dependent VQ densities. Within this scheme, similar Markov states share the same set of densities to reduce the computation required by CHMMs. However, since there are often a large number of senones in a system, the senone-dependent VQ densities are likely to be under-trained, if the amount of training data is not increased accordingly. Faced with limited training data, we may resort to a tighter constraint such as phone-dependent VQ densities, or even phone-class-dependent VQ densities. Phone-dependent VQ densities were considered by [5] and [84], where the CHMM density was assumed to be a Laplacian-type or a Gaussian density. Alternatively, the VQ level L (i.e., the number of mixtures M) should be reduced in order to be able to train the parameters with the fixed speech corpus. If we treat DHMMs and CHMMs as the two extreme types of HMMs, the relationship among the SCHMM and its variants can be illustrated in Figure 3.16.

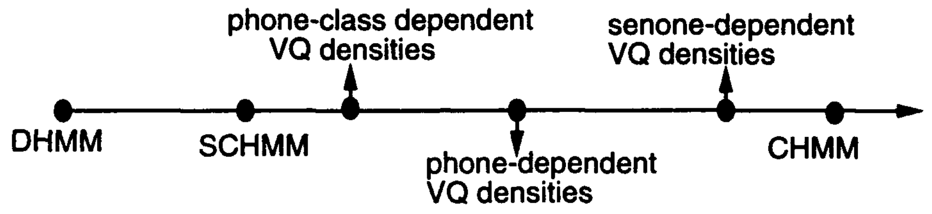

phone-class dependent VQ densities
senone-dependent VQ densities
DHMM
SCHMM
phone-dependent VQ densities
CHMM

Figure 3.16: The relationship of the DHMM, CHMM, SCHMM and its variants. As we relax the mixture-tying constraint in SCHMMs, they approach CHMMs, which have no mixture tying.

3.4.1 SCHMMs with Phone-Class Dependent VQ Densities

If we keep the VQ level L fixed at 256 and use phone-dependent VQ densities with SCHMMs, we are essentially increasing the resolution in the cepstrum space, as Figure 3.17 shows. In consequence, many codewords are never observed (or only observed a few times) when the amount of training data is not increased accordingly. As a result, there will be entries with a zero value in all senones corresponding to the phone which has at least one missing codeword. In Figure 3.17, the X's indicate under-trained codewords, and accordingly the corresponding entries in any senone for that phone become zero or some value close to zero.

Faced with a fixed training corpus, one way to avoid under-training is to cluster phones so that similar phones can share the same VQ subspace. Since there are only about 50 phones for English, clustering phones into classes can be easily achieved manually. Alternatively, the phonetic HMMs can be clustered automatically using cross entropy as the distance measure as in the bottom-up clustering algorithm described in Section 3.1.2. Specifically, we use the automatic clustering algorithm without the shift action to generate a tree-structured clustering for the 50 phones. Figure 3.18 shows the cluster structure produced by the automatic procedure. The number associated with each node indicates the reverse merging sequence. For example, it shows that phone /IX/ and /AX/ are the most similar pair and thus are merged first. This is quite consistent with linguistic knowledge.

With a fixed training corpus and a fixed number of senones, the work in Section 5.6 will

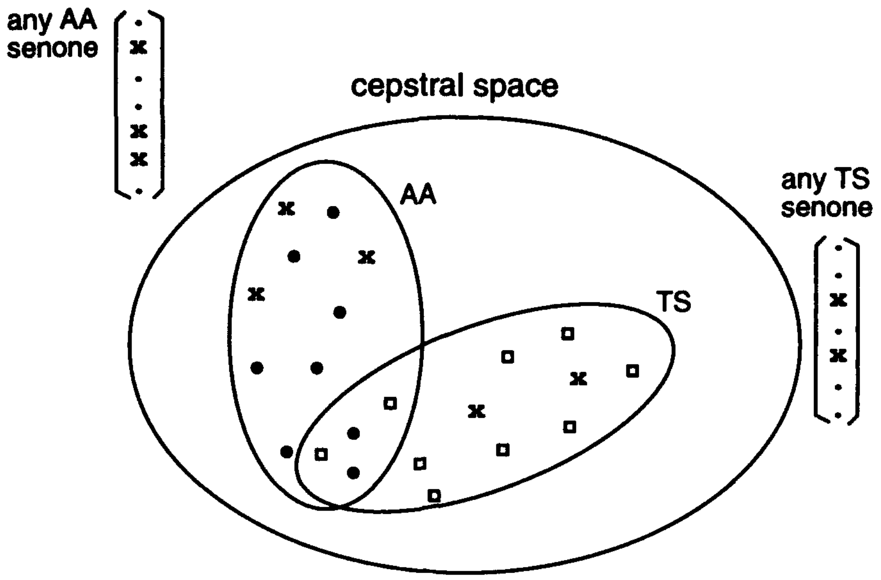


Figure 3.17: When phone-dependent VQ codebooks are used, the resolution in the cepstral domain is increased and thus some codewords are under-trained (never observed or observed a few times only), given a limited amount of training data. The corresponding senone entry is either zero or almost zero.

illustrate the effects of using different numbers of phone classes. We believe that with more training data, we should be able to train more phone classes and achieve more appreciable error reduction.

3.4.2 SCHMMs with Phone-Dependent, Reduced-VQ Densities

Another way to avoid under-training with a fixed training corpus is to reduce the VQ level (i.e., the number of mixtures in CHMM terminology). In Section 5.6, we will show our experience with 128 VQ densities for each phone. The optimal VQ level and the number of phone classes depend on the available training data (the amount and the content). Unfortunately, we don't have an analytic way to define the tradeoff function. Similarly, we don't have an analytic understanding of the optimal number of senones as a function of the training corpus. Neither do other systems have a deterministic function for the optimal number of triphones that should be kept, given a fixed training corpus.

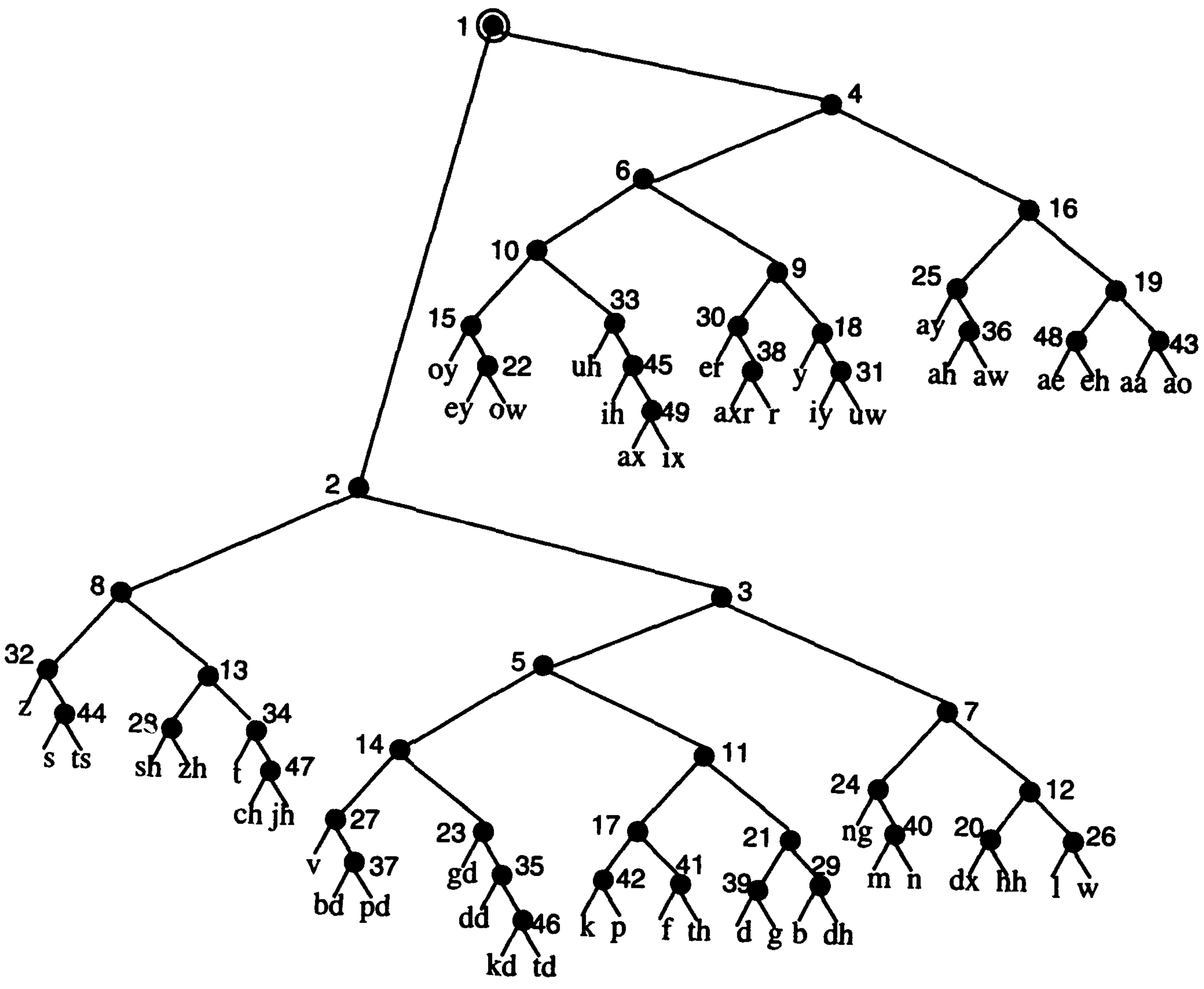


Figure 3.18: Automatic clustering of phones, using cross entropy of the output distribution associated with phonetic HMMs as the distance measure. The number associated with each node indicates the reverse merging sequence. This figure shows that the phone /IX/ and /AX/ are the most similar pair and thus are merged first.

3.4.3 Reducing VQ-Computation During Baum-Welch Training

When phone-dependent VQ codebooks are used, different sets of VQ densities are computed according to the different phones a data frame is aligned to. During decoding, we have to do all VQs with all phone codebooks for every time frame. That is, each time frame requires the computation of 51,200 Gaussian densities (50 phones * 256 densities/phone * 4 codebooks) in SPHINX-II. To save some of the expensive computation, we can use context-independent models as a fast match to prune unnecessary phones and thus avoid the VQ computation for the pruned phone classes. But this usually leads to some accuracy loss. For the purpose of our experiments, we do not use this fast-match pruning.

Fortunately, the costs are not as high during training since the correct sentence transcription is known. Recall the forward α computation of the Baum-Welch estimation of HMM parameters from Appendix B. At each time frame t, we compute the VQ densities $\{f_k^{(p)}(\mathbf{x}_t) \mid k\}$ for phone p only if

$$\exists \text{ state } s \in \text{ phone } p, \quad \text{s.t.} \quad \alpha_{t-1}(s) \neq 0$$

That is, to compute VQ for time frame t, we examine the α cells at time $t-1$ first. If $\alpha_{t-1}(s)$ is not zero, we check which phone (p) state s belongs to. At this point we must perform VQ for frame t with phone p's VQ densities since $\alpha_{t-1}(s)$ has non-zero contribution to some $\alpha_t()$ cells. Note that transitions at phone boundaries are null; therefore no VQ is necessary for null transitions. Figure 3.19 illustrates non-zero $\alpha_{t-1}(.)$ cells with black states. It shows that that data $\mathbf{x}_t$ should be quantized with both the UH and KD codebooks. If we start the training from a set of well-trained models, each time frame is usually aligned to only 3 or 4 phones. That is, only 3 to 4 sets of VQs have to be computed for each time frame, instead of 50. This implies a speed up by a factor of more than 10 for training.

As discussed in Appendix B, normalizing the density values can avoid floating-point exceptions and thus result in stable training runs. Appendix B makes clear that in order to maintain the same re-estimation value as the Baum-Welch formula defines, the same normalization factor has to be applied to all the VQ densities computed at different states for the same time frame. In other words, a data frame $\mathbf{x}_t$ may be aligned to a few phones $\Phi = \{p\}$. To guarantee the correctness of Baum-Welch re-estimation, we have to apply the same normalization factor to all of the VQ density values computed for the same data frame, regardless of which phone this frame is aligned to. Another important property of the normalization factor is that it has to be sensitive to different data frames so that a stable range of normalized densities is guaranteed to prevent floating-point exceptions. As Appendix B.2 discusses, when there was only one universal codebook, the normalization factor we usually adopted was the summation of the selected top few densities. That is, the normalized density was

$$\frac{f_k(\mathbf{x}_t)}{\sum_{k \in \eta} f_k(\mathbf{x}_t)}$$

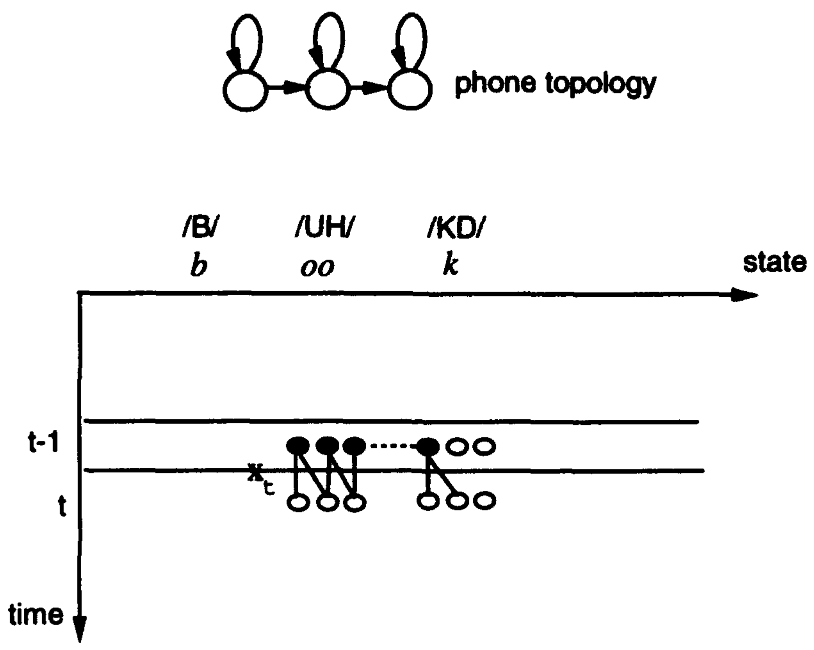


Figure 3.19: Computation of the forward pass in Baum-Welch reestimation, with the phone HMM topology shown in the upper part. Black states have non-zero $\alpha_{t-1}(.)$ values. It illustrates that frame $\mathbf{x}_t$ can be consumed inside phone UH or phone KD. Therefore $\mathbf{x}_t$ has to be quantized with both the UH VQ codebook and KD codebook, but not others.

One good candidate for the normalization factor for phone-dependent VQ codebooks is to take the average of the summations across the selected phone-dependent densities. In other words, the normalized density becomes

$$\frac{f_k^{(p)}(\mathbf{x}_t)}{\{\sum_{p\in\Phi}\sum_{k\in\eta} f_k^{(p)}(\mathbf{x}_t)\}\ /\ |\Phi|}$$

This indeed gives us stable training in our experiments.

3.5 Summary

This chapter explains all the subphonetic modeling techniques explored in this thesis. The most important concept examined in this dissertation is the analysis of acoustic behaviors at the HMM state level so that fine distinctions can be made and accurate models can be built.

The first section introduces the clustering of Markov states to create state prototypes called senones. Each senone is represented by a distribution, modeling the data observation at a state. Clustering at the state level instead of the entire phonetic model avoids the "mis-average" problem at dissimilar states. The set of models that shares senones is called the shared-distribution model (SDM). The SDM approach reduced the word error rate by 20%, in comparison with the generalized-triphone model, as is detailed in Section 5.1.

To be able to model unseen triphones with detailed parameters and at the same time take advantage of state sharing, we present the senonic decision tree. Binary decision trees with linguistic questions are built automatically to classify Markov states. Each leaf is a cluster of similar states and represents a senone. To find the senone associated with a Markov state of any triphone, either seen or unseen, we traverse the corresponding tree until a leaf is reached. This way, both seen and unseen triphones are modeled by senones in a unified fashion. Modeling unseen triphones with senones reduces the word error rate by 10% – 20%, in comparison with the situation in which context-independent phones are used for unseen triphones.

The disadvantage of decision trees is their limited re-configuration capability, even with composite questions, compared with the essentially unrestricted re-shuffling supported in the bottom-up clustering algorithm. Therefore, the decision-tree based sharing structure for seen triphones may be sub-optimal. One way to keep the optimal sharing structure for seen triphones and at the same time to model unseen triphones with senones is through a hybrid approach. The SDM is built for modeling seen triphones with the optimal sharing structure. Next decision-tree based generalized allophones are constructed for unseen triphones. To avoid doubling the number of free parameters, the output distribution associated with the Markov state of the generalized-allophone model is replaced by the best-matching senone. In other words, the Markov states of the generalized-allophone models are *quantized* with the senone codebook. Experimental results in Chapter 5 shows that the SDM indeed performs slightly better than the purely decision-tree based senones when unseen triphones are modeled by context-independent phones in both systems. When unseen triphones are modeled by senones in both systems, the hybrid approach performs as well as the purely decision-tree based senone. The hybrid system does not outperform the purely decision-tree based senone because the unseen-triphone modeling is poorer (possibly due to both the mis-average problem associated with the generalized allophone and state quantization errors).

The concept of state quantization can also be applied to pronunciation learning. Here the senonic baseform of a word is defined to be a word-HMM training procedure followed by state quantization. Like the fenonic baseform, the senonic baseform is purely acoustic driven. Unlike fenones, senones have better resolution in the acoustic space and senonic baseforms are not solved as a decoding problem, as are fenonic baseforms. The senonic baseform offers 15% word error reduction over the phonetic baseform on a small vocabulary task. Applying the senonic baseform to large vocabulary tasks has not yet proven to be successful. However, it does show promise for speaker adaptation and speaker-dependent applications. Possible improvements on the senonic baseform (not yet implemented due to time constraints and other interests) include the incorporation of the spelling information and a senone search strategy similar to the one used in the fenonic baseform, with a senone bigram model or a even more complex senone language model.

CHMMs remove VQ errors completely and have a fewer number of free parameters, compared with DHMMs. Therefore, some systems are able to provide high recognition accuracy using CHMMs [7]. However, one of the most severe problems associated with CHMMs is the heavy floating-point computation. To approach the performance of CHMMs without causing heavy computation, Section 4 introduces the SCHMMs with phone-class dependent VQ codebooks in which all the Markov states for the same *phone class* share the same set of VQ densities. This releases the tied-mixture constraint in SCHMMs where all the Markov states for *all* phones are tied with the same set of mixtures (i.e., the universal VQ codebook). With this higher resolution in the cepstral space and with the same training corpus, Chapter 5 shows that using 8 phone-class VQ codebooks reduces the word error rate by 8%. We believe that with more training data, this small increase in the number of free parameters will produce an appreciable error reduction. Alternatively, when the VQ level is reduced, we may even be able to train senone-dependent VQ codebooks. SCHMMs with senone-dependent VQ codebooks are equivalent to CHMMs with a senone sharing-structure, where similar states share the same set of mixtures.

This chapter concentrates on explaining the problems and suggested algorithms for solutions. To find the experimental performance of a particular algorithm, readers can jump directly to the appropriate section of Chapter 5.

[7]LIMSI in France, Philips in Germany, the HTK system of Cambridge University in England and Stanford Research Institute in the United States all used continuous-density HMMs and were the top accurate systems in the recent ARPA annual evaluation in November 1993, among other systems which used semi-continuous HMMs. See reports from ARPA Human Language Technology Workshop, March, 1994.

Chapter 4

Experimental Environments

To evaluate the techniques proposed in Chapter 3, we run through a series of experiments on ARPA's common speech corpora. This chapter will first describe the tasks, the training and testing corpora used in evaluating the subphonetic techniques depicted in the previous chapter. Next, since all experiments are based on the SPHINX-II speech recognition system [46, 66], we will review the overall structure of SPHINX-II. All experiments are explained and discussed in detail in Chapter 5.

SPHINX-II is an advance over the SPHINX system [90]. Both systems are software only [1], large-vocabulary, speaker-independent, continuous speech recognizers, based on HMMs. SPHINX-II works on a larger scale of vocabulary (5,000 $\sim$ 20,000 and up) than SPHINX (1,000-word vocabulary) and still provides real-time performance and high recognition-accuracy on workstations, such as Alpha3000/500 and Hp735. On the 5,000-word ARPA Wall Street Journal *read* speech, the word accuracy is above 95%. Section 4.2 will describe the new features that make SPHINX-II successful.

4.1 Tasks and Speech Corpora

To evaluate the techniques proposed in this thesis, two official tasks are chosen for experiments — the Resource Management (RM) task and the Wall Street Journal (WSJ) task. The 997-word RM task was chosen because when we began work on this thesis, RM had been the official DARPA (Defense Advanced Research Projects Agency of the Department of Defense)

[1]No special hardware except an A/D converter is required.

[2] evaluation task since 1988. Development of SPHINX and SPHINX-II has been in large part supported by the DARPA project. RM evaluations ended officially in September 1992.

Given the high-accuracy performance achieved at CMU and elsewhere on RM, DARPA encouraged the speech recognition community to work on a more complex dictation task, namely the WSJ task. A dry run was started in February 1992 and the first official evaluation on a 5,000-word dictation took place in November 1992. Soon after, in December 1992, there was another dry run on a 20,000-word open vocabulary task. The evaluation in 1992 demonstrated that highly-accurate, real-time continuous speech recognition systems for speaker-independent, very-large vocabulary tasks in the order of 20K are possible using HMM speech recognition techniques and state-of-the-art workstations for computation.

Researchers are currently concentrating their efforts on improving the accuracy, memory capacity, and speed of their recognizers, particularly for very-large vocabulary, speaker-independent continuous speech dictation. In this thesis, both RM and WSJ are tested in the speaker-independent continuous-speech mode.

4.1.1 Resource Management

The Resource Management task was designed as an inquiry for naval resources [112], but was general enough to cover database query tasks. It was created to evaluate the speech recognition systems of recent DARPA projects. Officially, the RM program started in May 1988 and ended in September 1992. The purpose of this task was to look for techniques that enable *large-vocabulary, speaker-independent continuous* speech recognition. When the program ended, the word accuracy on the task had been brought up to 95% – 97% for different test sets. The recognizer ran at real time on workstations such as the SUN-4 and DEC-5000 workstations with 16MB to 64MB of main memory. These achievements were made possible through the technique of HMMs. Unlike pattern-matching techniques such as Dynamic Time Warping, HMMs represent large vocabularies compactly through the concatenation of phonetic models. Speaker-independence was achieved by pooling training data from many speakers [3]. Continuous speech is elegantly represented by the concatenation of word HMMs and decoding is accomplished through the Viterbi beam search algorithm [132]. These same techniques have been extended to very-large vocabulary tasks such as the 20,000-word vocabulary WSJ task.

[2]DARPA was renamed ARPA in late 1992.

[3]Some work normalized the speaker cepstra before pooling to provide more accurate acoustic modeling [100, 54].

What is Brooke's fuel level and fuel capacity?
When is Plunger changing fleets?
Are Wednesday's casreps worse than Monday's for Kiska?
Display a chart of Canada with DMDS switches set to their defaults.
Is Poughkeepsie's Mob area m-rating below Prairie's?
Was Monday's last HFDF sensor location for the Hawkbill in Mozambique Channel?
Find me USN ships deployed in eighty four.
Display the definition of south Bering sea alert.

Figure 4.1: Sample utterances in the RM official training database.

Lexicon and Grammar

Figure 4.1 lists some sample utterances in the RM official training corpus. At the lexical level, the 997-word RM task is very difficult. There are many confusable pairs, such as *what* and *what's*, *the* and *a*, *four* and *fourth*, *any* and *many*, and many others. Most of the proper nouns can appear in singular, plural, and possessive forms, which creates still more confusable pairs and ambiguous word boundaries. There are many function words (such as *a*, *and*, *of*, *in*, *on*, *to*, *the*) that are articulated very poorly and are hard to recognize or even locate. Moreover, many of these function words are optional according to the grammar. The entire vocabulary of this task, along with the most likely pronunciation of each word can be found in [87].

At the grammatic level, RM is not very difficult because the sentences in this task are generated from a set of 900 sentence templates which resemble realistic questions in a database system. This task design reflects the philosophy that language modeling issues were to be assumed away by providing a complete model of the parses that would be encountered in the test set. Some examples of these 900 templates can be found in [87]. Each template is a network of tags (i.e. categories of words). Given these templates, the list of tags that can follow any given tag can be easily determined. Based on this information and the list of words in each tag, the list of words that can follow any given word can be generated. Among the 994,009 (997 $*$ 997) possible word pairs, only 57,878 are legal, according to the tagged templates. These legal word pairs constitute the official language model — the word-pair grammar. The word-pair grammar specifies only the list of words that can legally follow any given word. Uniform probabilities are used for all legal word pairs. That is, if there are k words that can follow word w, then the probability that any of these k words occurs next given the current word w, is $1/k$. This grammar has a test-set perplexity [71, 78] of about 60. The perplexity of a grammar (or language model) roughly corresponds to the average number of words that can follow a given word according to the grammar. Given that there are 57,878 legal word pairs spawned from 997 words, we can estimate the perplexity of the word-pair grammar to be $57,878/997 = 59$, which is consistent with the test-set perplexity.

RM was also evaluated under the no-grammar condition, where any word can be followed by any word with the uniform probability ($1/n$ where n is the vocabulary size and equal to 997 for RM), resulting in a perplexity of 997.

Training and Testing Corpora

RM has an official common training set, the TIRM corpus, which consists of 3,990 read sentences from 105 speakers (30 or 40 sentences for each speaker). The ratio of male to female speakers is about two to one. These sentences were recorded at Texas Instruments (TI) using a Sennheiser HMD-414-6 close-talking, noise-canceling, headset-boom microphone in a sound-treated room [36, 37]. The speech was sampled at 20 kHz at TI, then down-sampled to 16 kHz at the National Bureau of Standards.

The test set used in this thesis consists of 300 sentences from the February-1989 evaluation set and another 300 sentences from the Oct ber-1989 evaluation set. There are in total 20 speakers; none of them is covered by the training corpus. These testing sentences were also recorded at TI under the same recording condition as used for the TIRM corpus.

4.1.2 Wall Street Journal

Given the promising results from RM, researchers have moved to a more ambitious goal since 1992 — the Wall Street Journal task. The WSJ corpus consists of approximately 45 million words of text published by the Wall Street Journal between the years of 1987 and 1989. This corpus was made available through the Association for Computation Linguistics/Data Collection Initiative (ACL/DCI) [93].

Lexicon and Grammar

For the purposes of the February-1992 dry run, eight standard bigram language models were provided by D. Paul at Lincoln Labs [109]. A bigram language model, like a word-pair grammar, specifies legal word pairs. In addition, the word transition probabilities are trained from a large text corpus, rather than set to be uniform. The WSJ language models were trained only on the WSJ data that was not held out for acoustic training and testing. The language models are characterized along three dimensions, lexicon size (N=5k or N=20k), closed or open vocabulary (c or o), and verbalized or non-verbalized punctuations (vp or nvp). The distinction between open and closed vocabulary is in the method used to choose the lexicon.

For the open vocabulary, the lexicon approximately consists of the N most common words in the corpus. For the closed vocabulary, a set of N words were selected in a manner that would allow the creation of a sub-corpus that would have 100% lexical coverage by this closed vocabulary. The open-vocabulary condition is suitable for the research dealing with out-of-vocabulary (OOV) words, which for large vocabulary is considered a very important problem (and still an unsolved mystery) in speech recognition.

In the verbalized-punctuation condition, punctuations are pronounced verbally. For example, if a sentence written on a Wall Street Journal looks like

```
A white house game of "chicken" with Germany and Japan,
played by talking down the dollar, hasn't helped either.
```

The speaker would have said,

```
A white house game of double-quote chicken double-quote
with Germany and Japan comma played by talking down the
dollar comma hasn't helped either period
```

As punctuations occur in every sentence and punctuations are acoustically easy words, we can predict that the perplexity for the vp condition is lower than that for nvp and the recognition accuracy for vp sentences is higher than that for nvp. Table 4.1 lists the development-set perplexities for the eight language models. As we predicted, the vp perplexities are indeed lower than nvp perplexities. The word lists in close and open vocabularies are quite different, since they are derived using different methods. Therefore, comparing the perplexity of close and open vocabulary conditions is meaningless. For further details about the definition of the WSJ task, see [109].

	5k		20k	
	closed	open	closed	open
vp	80	72	158	135
nvp	118	105	236	198

Table 4.1: Perplexities for the eight WSJ language models.

To be able to afford many experiments in limited time, only the 5k closed vocabulary, non-verbalized punctuation condition (denoted as 5c-nvp thereafter) is tested in this dissertation. As shown in Table 4.1, the development-set perplexity for 5c-nvp is 118.

Despite the decline in stock prices trading volume wasn't overwhelming
And the total allowance for credit losses remained unchanged at one point three billion dollars
Clearly ,comma something is changing in Europe .period
A white house game of "double-quote chicken "double-quote with Germany and Japan ,comma played by talking down the dollar ,comma hasn't helped either .period
But what exactly is its threat to life or health ?question-mark

Figure 4.2: Sample utterances in the WSJ training corpus. The first two utterances are nvp sentences; the last three are vp.

Training and Testing Corpora

The official speaker-independent training corpus (si_trn) provided by the National Institute of Standards and Technology (NIST) consists of 7240 utterances of read WSJ text, equally divided among vp and nvp texts [106]. The texts chosen to train the system were quality filtered to remove very long and very short sentences as well as to remove sentences containing words not among the 64k most frequently occurring words in the WSJ corpus [109]. The data was collected from 84 speakers [4], equally divided among male and female. Data recording was performed at three different locations, Massachusett Institute of Technology (MIT), Stanford Research Institute (SRI) and Texas Instrument (TI). At all three locations the same close speaking, noise canceling microphone was used. However environmental conditions vary from a sound booth to a laboratory environment. Figure 4.2 lists examples of the training utterances, where *"double-quote* represents the pronunciation of *double-quote* (similarly for other punctuation words).

To test the 5c-nvp speaker-independent continuous speech recognition, two data sets are used for performance comparison: si_dev5 and nov92. The development set, si_dev5, originally consisted of 410 utterances from 10 speakers. However, it contains an outlier speaker (422) who has a fast speaking rate and a high-pitched voice which cause significant performance degradation for the entire set. To calibrate the performance across different test sets and to better isolate the problems of general acoustic modeling (which are addressed in this dissertation) and special techniques for handling outlier speakers (under investigation by BBN Systems and Technologies and others [121]), the utterances from speaker 422 were removed from our test set. In addition, the mis-transcribed utterance 053c100n were also removed. This results in 367 utterances from 9 speakers (4 female and 5 male) in the si_dev5 set. The November-1992 evaluation set (nov92) consists of 330 utterances from 8 speakers (4 male and 4 female). None

[4]One of the speakers in the training corpus was recorded twice but at different sites and so this person is counted as two different speakers.

of these 18 speakers is covered by the training corp 's.

4.2 The SPHINX-II System

Since the subphonetic modeling experiments in this thesis were tested in the context of the SPHINX-II speech recognition system, this section will review the overall structure of SPHINX-II. SPHINX-II, like its predecessor SPHINX, is a large-vocabulary, speaker-independent, continuous speech recognizers, based on HMMs.

SPHINX was first announced at CMU in 1988 [87, 89]. Since then, it has undergone significant improvements. Because of the dramatic error reduction in the intervening years, it has been renamed SPHINX-II [46, 66] to distinguish it from the first announced version. The additions include:

- Between-word triphones are incorporated [62] to model word-boundary co-articulation in continuous speech.
- Mel-scale frequency coefficients (MFCs) with cepstral mean normalization (CMN) replaced the linear-predictive coefficients (LPCs) [31], as they offer consistently better performance on a few well-known systems [105].
- Four independent sets of signal features are extracted instead of three [48, 57]. Specifically, second-order differenced cepstra and first-order long-term differenced cepstra are added.
- Sex-dependent semi-continuous HMMs [56] replace sex-independent discrete HMMs.
- A three-pass search algorithm [3] replaces the one-pass Viterbi beam search [132, 95, 123] to process very large vocabularies and long-distance language models efficiently.
- Subphonetic modeling techniques presented in this thesis [57, 67, 64, 68] are incorporated to add accuracy and robustness to the acoustic model.

This section will summarize the above components. Details about the the subphonetic modeling techniques were already presented in the previous chapter. Experimental performance of the subphonetic modeling techniques can be found in the next chapter.

4.2.1 Signal Processing

Speech signals are highly redundant because of their strong correlation between adjacent segments. Therefore, speech recognition systems always use a parametric representation rather than the speech waveform itself. Not only is useful information compactly extracted from the waveform, but computation is saved for both training and decoding (as the speech is compressed). The front-end of SPHINX-II is illustrated in Figure 4.3. The following steps are taken:

1. The input speech waveform is sampled at 16 kHz.
2. The sampled waveform is blocked into frames. Each frame spans 25.6 msec (about 410 samples) and consecutive frames overlap by 10 msec. In other words, each 25.6-msec speech segment is multiplied by a Hamming window. Then the window is shifted by 10 msec to compute the next frame.
3. A preemphasis filter $H(z) = 1 - 0.97z^{-1}$ is applied to get rid of the lip effect [98].
4. 12 mel-scale frequency coefficients are computed for each frame:

   $$\mathbf{x}_t(k) \quad 1 \le k \le 12$$

   where t is in the unit of frames, i.e., 10 msec.
5. For each time frame t, four sets of features are computed from the MFCs:
   - Cepstra normalized by the sentence-based mean:

     $$\begin{aligned} \mu(k) &= \frac{1}{T}\sum_t \mathbf{x}_t(k) \quad 1 \le k \le 12 \\ & \quad \text{where } T = \text{ number of frames of the input utterance} \\ \mathbf{x}_t(k) &\longleftarrow \mathbf{x}_t(k) - \mu(k) \quad 1 \le k \le 12 \end{aligned}$$

   - 40-ms and 80-ms differenced MFCs. This constitutes a vector of 24 dimensions.

     $$\begin{aligned} \Delta\mathbf{x}_t(k) &= \mathbf{x}_{t+2}(k) - \mathbf{x}_{t-2}(k) \quad 1 \le k \le 12 \\ \Delta\mathbf{x}_t'(k) &= \mathbf{x}_{t+4}(k) - \mathbf{x}_{t-4}(k) \quad 1 \le k \le 12 \end{aligned}$$

   - second-order differenced MFCs.

     $$\Delta\Delta\mathbf{x}_t(k) = \Delta\mathbf{x}_{t+1}(k) - \Delta\mathbf{x}_{t-1}(k) \quad 1 \le k \le 12$$

- normalized power (log energy), differenced power, second-order differenced power.

$$
\begin{aligned}
\mathbf{x}_t(0) &\longleftarrow \mathbf{x}_t(0) - \max_i\{\mathbf{x}_i(0)\} \\
\Delta\mathbf{x}_t(0) &= \mathbf{x}_{t+2}(0) - \mathbf{x}_{t-2}(0) \\
\Delta\Delta\mathbf{x}_t(0) &= \Delta\mathbf{x}_{t+1}(0) - \Delta\mathbf{x}_{t-1}(0)
\end{aligned}
$$

These four sets of features

$$
\{\mathbf{x}_t\}_{12}, \quad \left\{ \begin{array}{c} \Delta\mathbf{x}_t \\ \Delta\mathbf{x}_t' \end{array} \right\}_{24}, \quad \{\Delta\Delta\mathbf{x}_t\}_{12}, \quad \left\{ \begin{array}{c} \mathbf{x}_t(0) \\ \Delta\mathbf{x}_t(0) \\ \Delta\Delta\mathbf{x}_t(0) \end{array} \right\}_3
$$

are assumed to be independent for mathematic and implementation simplicity.

Vector Quantization

As we will see later, discrete HMMs are often used as the seed model to bootstrap the training of semi-continuous HMMs. For discrete HMMs, each frame of the speech is represented by a discrete number instead of the continuous vector $\mathbf{x}_t$. As mentioned earlier, vector quantization (VQ) [39, 97] is the data reduction technique that maps a real vector onto a discrete symbol. A vector quantizer is defined by a codebook, which is comprised of the prototype vectors, and a distortion measure that estimates the proximity of two vectors.

In SPHINX and SPHINX-II, one codebook with 256 prototypes is created for each of the signal features mentioned in the previous paragraph, using approximately 10^5 to 10^6 randomly selected vectors. The result is four sets of codebooks, which are assumed to be independent. The prototypes are estimated via a hierarchical clustering algorithm similar to the K-means algorithm developed by Linde *et al* [94], which is an approximate maximum likelihood method. A prototype is the mean of a cluster of *similar* vectors. In addition to the mean, the covariance of each cluster is also computed to bootstrap the training of the SCHMM codebook ($\{f_k\}$ in Formula (A.4)). Figure 4.4 illustrates the mean vector and the covariance matrix for one codeword (k) in the power codebook. In SPHINX-II, each codeword is modeled by a Gaussian with a diagonal covariance. The covariance is assumed to be diagonal (i.e., each dimension is independent of one another) to reduce computation and enable the covariance to be well trained given a limited amount of training data. These means and covariances will be used as the initial values for training SCHMM codebooks.

For discrete HMMs, every input vector of every speech feature is symbolized by the closest prototype using the Euclidean distance. That is, for DHMMs, each frame of speech data

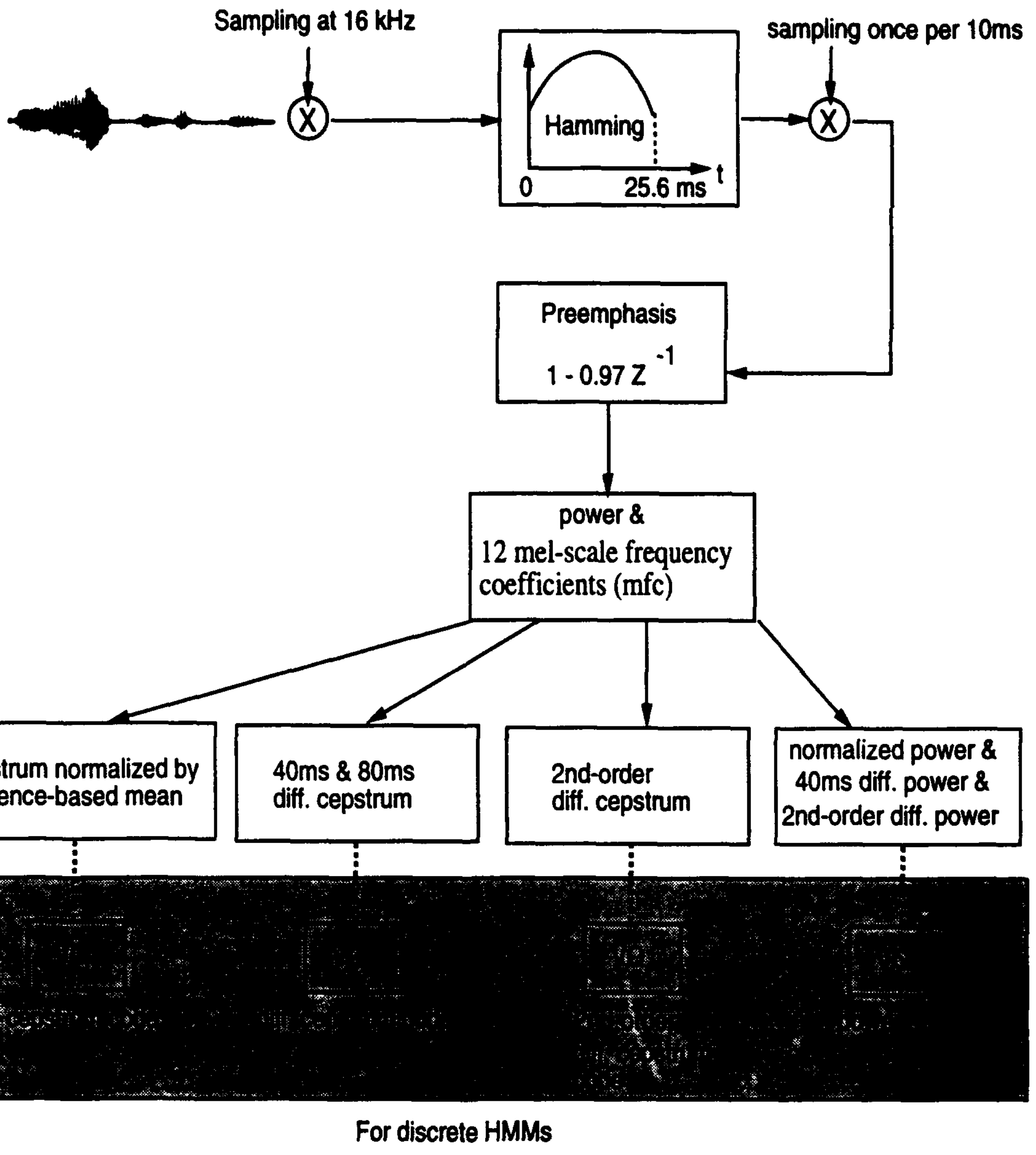
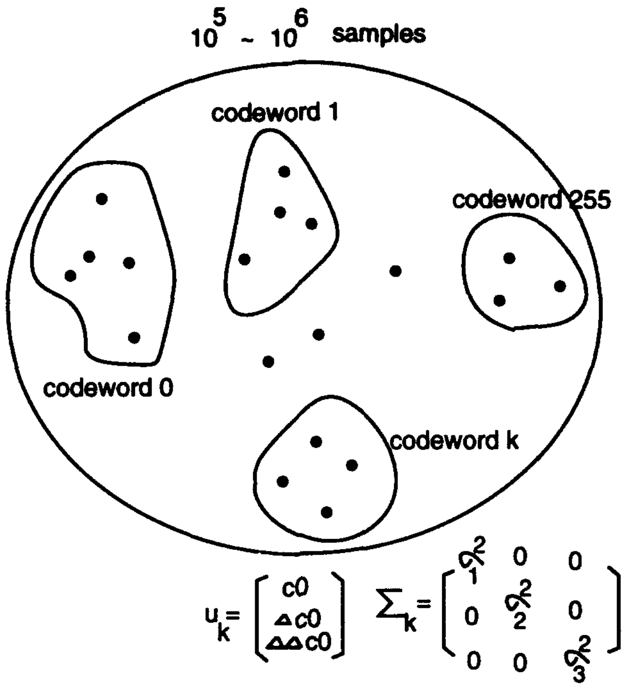

Sampling at 16 kHz
Hamming
0
25.6 ms
t
sampling once per 10ms
Preemphasis
1 - 0.97 Z^-1
power &
12 mel-scale frequency coefficients (mfc)
cepstrum normalized by sentence-based mean
40ms & 80ms diff. cepstrum
2nd-order diff. cepstrum
normalized power & 40ms diff. power & 2nd-order diff. power
For discrete HMMs

Figure 4.3: The signal processing front-end in SPHINX-II.



$10^5 - 10^6$ samples
codeword 1
codeword 255
codeword 0
codeword k
$u_k = \begin{bmatrix} c0 \\ \Delta c0 \\ \Delta\Delta c0 \end{bmatrix}$
$\Sigma_k = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}$

Figure 4.4: Computing the mean and covariance matrix for each codeword in the power codebook.

requires only four bytes of memory (one for each codebook) instead of 52 bytes (twelve 32-bit floating-point MFCs and one power) for CHMMs or SCHMMs. Thus the space requirement for storing the speech corpus for DHMMs is 1/13 of that for CHMMs or SCHMMs, at the price of vector quantization errors. Context-independent phonetic DHMMs will be used as the initial values for training context-dependent SCHMM parameters (transition and output probabilities).

4.2.2 The Phone Set and the Pronunciation Dictionary

The Basic Phone Set

As discussed in Chapter 1, word models are not practical for large-vocabulary tasks because the amount of training data and the storage required are intractable. Therefore, both SPHINX and SPHINX-II are based on context-dependent phonetic models to allow parameter sharing and detailed modeling. In SPHINX-II, in addition to the fifty English phones plus three silence models for silences at the beginning, middle and ending parts of an utterance, we also add ten noise models for non-speech voices like door bumps, tongue clicks, inhalations and so on. The three silence models are motivated by the observation that for read speech inhalation and breath noises often occur at the beginning and end of an utterance, while very seldom in the middle of an utterance. However, making this fine silence separation does not offer any improvement (nor does it hurt the system) empirically. The noise models are helpful for spontaneous speech, where speakers sometimes hesitate or pause with non-speech voices. The experiments conducted in this dissertation are only for read speech; the noise models are therefore not used. Nevertheless, SPHINX-II and all the techniques described in this thesis are applicable to spontaneous speech with little modification. The whole 63-phone set, together with examples, is listed in Table 4.2. [5]

Phone	Example	Phone	Example	Phone	Example
/BUMP/	*door bump*	/CLICK/	*tongue click*	/EXHALE/	*exhalation*
/INHALE/	*inhalation*	/NOISE/	*office noises*	/POP/	*pop noises*
/RUSTLE/	*paper rustle*	/SMACK/	*tongue smack*	/SWALLOW/	*saliva swallow*
/UH/	*uh, ah, em, er, etc..*				
/SIL/	(middle silence)	/SILb/	(begin sil)	/SILe/	(end sil)
/IY/	*b**ea**t*	/R/	**r***ed*	/K/	**k***ic***k**
/IH/	*b**i**t*	/Y/	**y***et*	/BD/	*ro***b**
/EH/	*b**e**t*	/W/	**w***et*	/DD/	*ba***d**
/AE/	*b**a**t*	/ER/	*b**ir**d*	/GD/	*do***g**
/IX/	*ros**e**s*	/AXR/	*din**er***	/PD/	*wra***p**
/AX/	*th**e***	/M/	**m***o***m**	/TD/	*si***t**
/AH/	*b**u**t*	/N/	**n***o***n**	/KD/	*si***ck**
/UW/	*b**oo**t*	/NG/	*si***ng**	/TS/	*ca***ts**
/UH/	*b**oo**k*	/CH/	**ch***ur***ch**	/Z/	**z***oo*
/AO/	*b**ough**t*	/JH/	*ju***dge**	/ZH/	*mea**s**ure*
/AA/	*c**o**t*	/DH/	**th***ey*	/V/	**v***ery*
/EY/	*b**ai**t*	/B/	**b***o***b**	/F/	*brie***f**
/AY/	*b**i**te*	/D/	**d***a***d**	/TH/	**th***ief*
/OY/	*b**oy***	/DX/	*bu**tt**er*	/S/	**s***i***x**
/AW/	*ab**ou**t*	/G/	**g***a***g**	/SH/	**sh***oe*
/OU/	*b**oa**t*	/P/	**p***o***p**	/HH/	**h***ay*
/L/	**l***ed*	/T/	**t***o***t**		

Table 4.2: The phone set used in the SPHINX-II system.

[5]For Resource Management Task described in Section 4.1.1, we did not have phones SILb, SILe, AXR, BD, GD, and ZH , but instead we had EN.

Context-Dependent Phones

Since monophones are inadequate for modeling acoustic phenomena accurately, both SPHINX and SPHINX-II are based on triphone models. Moreover, while modeling continuous speech, we notice strong co-articulation at word boundaries. For example, the spectrograms in Figure 4.5 [61] show that the realization of the `/IY/` phone in *we* is strongly affected by the next phone in the utterance.

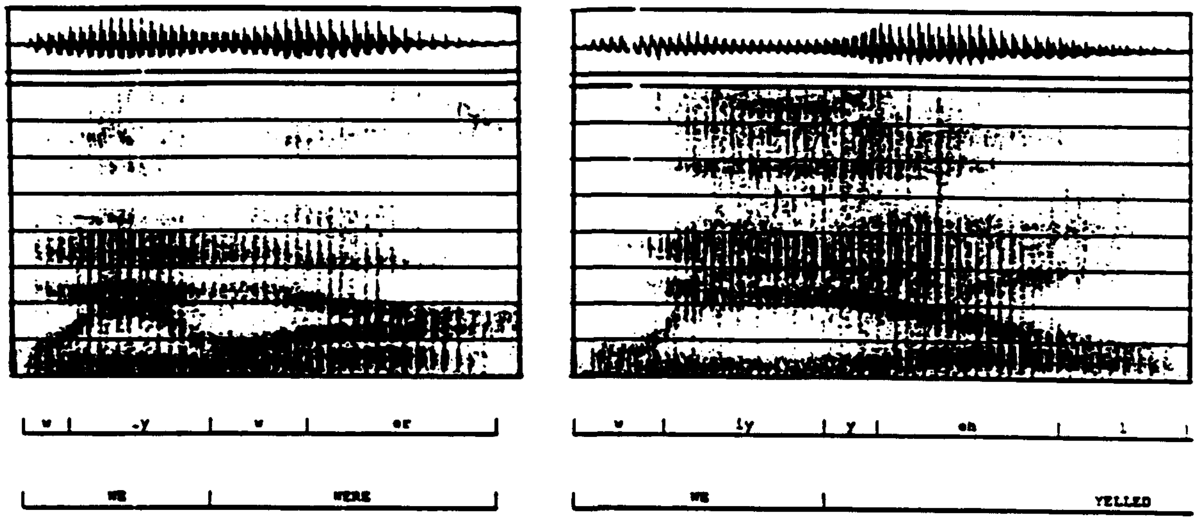


Figure 4.5: The waveforms and spectrograms for *we were* and *we yelled*. It is clear that the realization of phoneme `/IY/` in *we* depends on the contexts across the word boundary.

To improve the stochastic modeling structure for continuous speech, between-word triphones [63, 108, 135], which account for word-boundary co-articulation effects, are also modeled. While considering between-word triphones, we also notice that the same triphone has different acoustic realizations at different word positions. As illustrated in Figure 4.6, it is obvious that the spectrum of triphone `/T(AE,R)/` at the end of a word (*that*) is very different from that of the same triphone in the middle of a word (*theatrical*).

To take into account word-position effects, triphones are classified into four types; different types constitute different triphone models [62, 83]:

- Within-word triphones. This is a triphone occurring within a word, like `T(AE,R)` in *theatrical*.
- Beginning-word triphones. This is a triphone occurring at the beginning position of a word, like `R(T,AO)b` and `R(SIL,AO)b` in *that rock*[6]. Note that this type of triphone

[6]We always allow an optional silence between two words. Therefore, *that rock* can be spoken as *that-rock* or *that-SIL-rock*.

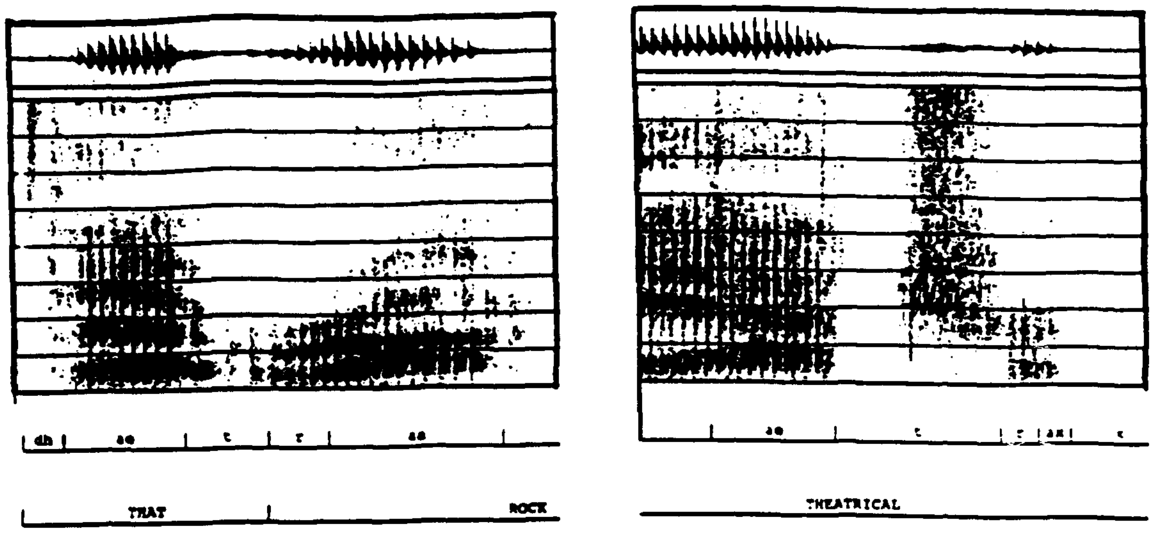


Figure 4.6: The waveforms and spectrograms for *that rock* and *theatrical*. The realization of triphone /T(AE,R)/ is dependent on the word positions.

is appended by the letter b in order to distinguish it from the same triphone at other word positions. Note also that the left-context of the first triphone for a word depends on the last phone of the preceding word. That is, when a word is referred to in isolation, we *can only specify the biphone for its first phone, e.g.* R(?,AO)b *for rock, if the word* consists of at least two phones.

- Ending-word triphones. This is a triphone occurring at the ending position of a word, like T(AE,R)e and T(AE,SIL)e in *that rock*. Note that this type of triphone is suffixed by the letter e to make it distinct with respect to word positions. In analogy with beginning-word triphones, the right context of the last triphone for a word is dependent on the first phone of the succeeding word. Thus, when a word is seen in isolation, we can only specify the biphone for its final phone, e.g. T(AE,?)e for *that*, if the word consists of at least two phones.

- Single-phone triphones. These are triphones particularly for single-phone words, like AX(Z,B)s, AX(Z,SIL)s, AX(SIL,B)s, and AX(SIL,SIL)s in *This is a book*. Note that this type of triphone has a suffix of s. Without the contextual words, we can only specify the monophone for single-phone words, e.g. AX(?.?)s for *a*. Because the acoustic evidence of a single-phone word is so short and highly co-articulated with the neighboring words, it is hard to decide which of the above three types of triphones the single phone should belong to. This is the motivation for the fourth type. Both left and right contexts of a single-phone triphone are unspecified until the contextual words are given.

Because of the huge number of triphones (maximum $51^3 * 4$), triphones are actually not used directly. Instead, a parameter-sharing technique called *triphone generalization* was used in SPHINX and another one called *distribution sharing* is used in SPHINX-II (Section 3.1) to reduce the number of parameters to be tractable and trainable. The performance comparison between generalized triphones and the shared-distribution model is one of the key experiments in the next chapter.

The Pronunciation Dictionary

The triphone transcription for each word is specified according to the above four types of triphones. Table 4.2.2 shows the triphone pronunciations for some words. The question mark represents an unspecified phone which will be instantiated by the last phone of the preceding word or the first phone of the succeeding words once the contextual words are known.

A	/AX(?,?)s/
THE	/DH(?,AX)b AX(DH,?)e/
SPHINX	/S(?,F)b F(S,IH) IH(F,NG) NG(IH,KD) KD(NG,S) S(K,?)e/

Table 4.3: Examples of the lexicon, spelled by position-dependent triphones. The question mark will be instantiated by the last phone of the preceding word or the first phone of the succeeding word in the utterance.

4.2.3 Training Architecture

Sentence Representation Using Hidden Markov Models

Both SPHINX and SPHINX-II are triphone-HMM based speech recognition system. Word models can be easily formed by concatenating component triphones or networking alternative phonetic pronunciations [103]. Sentence models are in turn formed by concatenating a sequence of word models, with an optional silence between two consecutive words[7]. Transitions at phone boundaries and word boundaries are all null arcs; i.e., they do not consume any data. This word concatenation leads conceptually to a large network of legal HMM states. This structure is what makes HMMs particularly suitable for continuous speech recognition. For example, Figure

[7]The other way is to locate silences first and then construct the sentence model with between-word silences dictated by the location found [121].

4.7 illustrates the sentence HMM for the utterance *What's Mars' status?*. For an extensive discussion on constructing sentence HMMs considering between-word triphones, see [61, 62].

$$\begin{tikzcd}[row sep=small, column sep=small]
 & & & \text{TS(AH,SIL)e} \arrow[r] & \text{SIL} \arrow[r] & \text{M(SIL,AA)b} \arrow[dr] & & \\
\text{SIL} \arrow[r] & \text{W(SIL,AH)b} \arrow[r] & \text{AH(W,TS)} \arrow[ur] \arrow[dr] & & & & \bullet \arrow[r, dash] & {} \\
 & & & \text{TS(AH,M)e} \arrow[rr] & & \text{M(TS,AA)b} \arrow[ur] & &
\end{tikzcd}$$

$$\begin{tikzcd}[row sep=small, column sep=small]
 & & & \text{S(R,SIL)e} \arrow[r] & \text{SIL} \arrow[r] & \text{S(SIL,D)b} \arrow[dr] & & & \\
{} \arrow[r] & \text{AA(M,R)} \arrow[r] & \text{R(AA,S)} \arrow[ur] \arrow[dr] & & & & \bullet \arrow[r] & \text{D(S,AE)} \arrow[r, dash] & {} \\
 & & & \text{S(R,S)e} \arrow[rr] & & \text{S(S,D)b} \arrow[ur] & & &
\end{tikzcd}$$

$$\begin{tikzcd}[column sep=small]
{} \arrow[r] & \text{AE(D,DX)} \arrow[r] & \text{DX(AE,AX)} \arrow[r] & \text{AX(DX,S)} \arrow[r] & \text{S(AX,SIL)e} \arrow[r] & \text{SIL}
\end{tikzcd}$$

Figure 4.7: The sentence connection for *What's Mars' status?* when between-word triphones are modeled.

Techniques such as DTW, knowledge-engineering, traditional pattern recognition, and neural networks face serious problems in training their models under continuous speech, because word boundaries are not automatically detectable. Often, hand-marking is needed, which is tedious, typically suboptimal and not feasible for automated systems. HMMs, on the other hand, do not have this problem. To use continuous speech, sentence HMMs are constructed as described above. Then the utterance is automatically aligned with the sentence HMM according to the progressive topology in the HMM, by a maximum likelihood criterion. Training HMMs on continuous speech is not much more difficult than training on isolated words.

Two-Stage Training in SPHINX-I and II

The training procedure involves optimizing HMM parameters given an ensemble of training data. Because there is no known analytic method to obtain the optimal parameters in a maximum likelihood (ML) model, an iterative procedure, the forward-backward algorithm, also known as the *Baum-Welch algorithm* [20, 7], is generally used to estimate both the output and transition probabilities. The goal of the forward-backward algorithm is to maximize the joint probability of generating the training data given the model. Other parameter estimation approaches include maximum mutual information estimation (MMIE) [25] and corrective training [8, 91]. MMIE minimizes the average uncertainty of the training data given the model, instead of finding true model parameters. Corrective training attempts to maximize the difference between the probabilities of the correct word string and the incorrect one. Among these, SPHINX-II uses the forward-backward ML estimation.

In either SPHINX-I or II, there are two stages for acoustic training, as shown in Figure 4.8. The first stage is to generate the generalized-triphone mapping table for SPHINX-I or the senone mapping table for SPHINX-II. After the mapping table is obtained, the second stage is to train the final models which share their parameters according to the mapping table. For SPHINX-I, the final models are sex-independent DHMMs; for SPHINX-II, they are sex-dependent SCHMMs.

First-Stage Training To construct the mapping table, a set of triphone models has to be estimated first for computing the distance measure, as described in Figure 3.5 and Figure 3.10. For the sake of simplicity, one-codebook discrete HMMs are usually used at this stage, where the acoustic feature is packed from the cepstral coefficients, 40-msec differenced cepstrum, power and 40-msec differenced power (i.e., a vector of dimension 26). The VQ level is 256.

As shown in Figure 4.8(a), to estimate the triphone DHMMs we bootstrap the training from context-independent phone models, which are usually in turn trained from the uniform distribution (i.e., every probability entry for the output distribution is 1/256; every transition probability is $1/a_s$, where a_s is the number of transitions out of state s). Deleted interpolation [72] is used to smooth the estimated *monophone* parameters with the uniform distribution, but is *not* used for triphone models. Triphone models are not smoothed because before clustering, they are very much under-trained due to *insufficient training data. Interpolation* would smooth out triphone characteristics and make many triphones indistinguishable.

After the triphone models are estimated, the triphone clustering [86] procedure is applied for SPHINX-I to generate the generalized-triphone mapping table. For SPHINX-II, the distribution clustering procedure [64] is applied to generate the senone mapping table. For generalized triphones, the goal is to cluster similar *triphones* so that these shared parameters can be well trained. For example in Figure 4.8(a), both triphone `EY(L,SH)` (*relation*) and `EY(L,S)` (*place*) are represented by the same generalized-triphone model in SPHINX-I. That is, they are indistinguishable in the generalized-triphone system. Each entry of the mapping table in SPHINX-I maps a triphone to a generalized one. For the shared-distribution model, the goal is to cluster similar *Markov states* so that they can share the same output distribution. Each entry of the mapping table in SPHINX-II maps a triphone to a sequence of senone labels. For example, the mapping table illustrated in Figure 4.8(a) shows that `EY(L,S)` and `EY(L,SH)` can be distinguished by the last Markov states since they are labeled with different senones.

Second-Stage Training After the number of parameters is reduced by the clustering procedure, we are now able to train the generalized triphones or senones reliably. The second training stage is therefore to train the 4-codebook triphone HMMs (Figure 4.8(b)), using the mapping table constructed at the first stage. Note that the mapping table is sex-independent. For

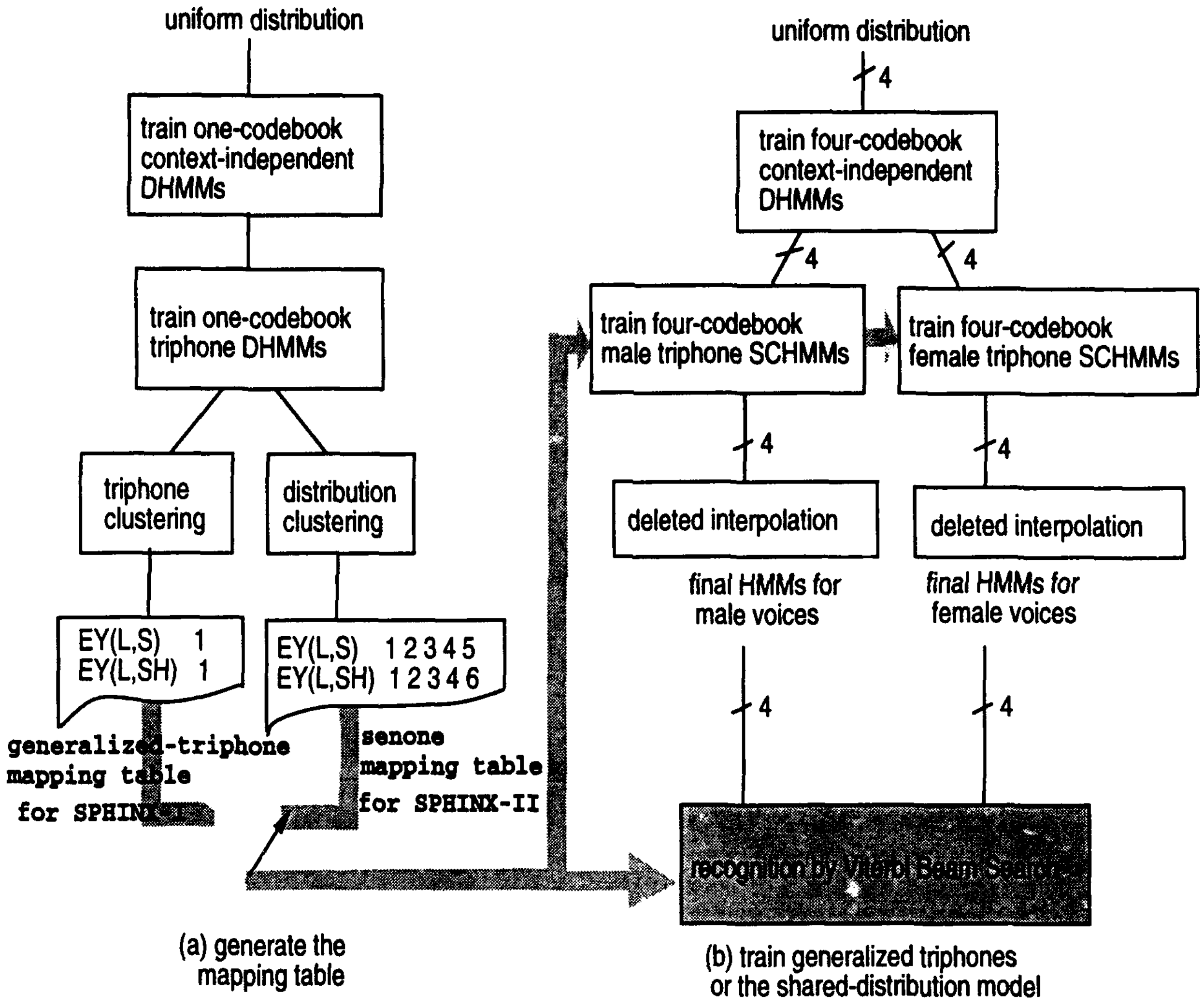

uniform distribution
train one-codebook context-independent DHMMs
train one-codebook triphone DHMMs
triphone clustering
distribution clustering
EY(L,S) 1
EY(L,SH) 1
EY(L,S) 1 2 3 4 5
EY(L,SH) 1 2 3 4 6
generalized-triphone mapping table for SPHINX-I
senone mapping table for SPHINX-II
(a) generate the mapping table
uniform distribution
4
train four-codebook context-independent DHMMs
4
4
train four-codebook male triphone SCHMMs
train four-codebook female triphone SCHMMs
4
4
deleted interpolation
deleted interpolation
final HMMs for male voices
final HMMs for female voices
4
4
recognition by Viterbi Beam Search
(b) train generalized triphones or the shared-distribution model

Figure 4.8: Two stages of system training. When SPHINX-I is used, the connection at the bottom of Figure (a) is switched to triphone clustering and the sex-dependent SCHMMs in Figure (b) are replaced by sex-independent DHMMs.

SPHINX-I, the mapping table is a generalized-triphone mapping table and the triphone models are sex-independent DHMMs. For SPHINX-II, the mapping table is a senone mapping table and the triphone models are sex-dependent SCHMMs. With SCHMMs' smoothing capability (by choosing top few codewords with the density values as weights), we are able to train the two sex-dependent models in order to separate the apparently distinct frequency responses of male and female vocal tracts.

In SPHINX-II, SCHMMs replace DHMMs to reduce vector quantization errors. The continuous densities for modeling the VQ codewords are assumed to be Gaussian densities with diagonal covariances, and are initialized by the sex-independent means and covariances of the VQ prototypes, as mentioned in Section 4.2.1. The output distributions (or senones) and the transition probabilities in the triphone SCHMMs are usually initialized with those of context-independent phonetic DHMMs, which are boosted from uniform probabilities. However, as discussed in Section 3.1.4 we can easily start senonic triphone training from a set of existing triphone models to speed up the convergence of the ML parameter estimation. Since transition probabilities are known to be much less important than the output distributions, they are assumed to be context-independent in SPHINX-II. This also saves us some space. All the parameters (the transition probabilities, output probabilities, and codebook densities) are jointly estimated by the Baum-Welch algorithm [49].

4.2.4 Recognition Architecture

Static Language Networks for Small-Vocabulary Tasks

For a small task like RM, a language network is pre-compiled to represent the search space during recognition to speed the decoding procedure. Word models are formed by concatenating phonetic models. Connections between words are determined by the language model, if any. The language model also decides the probability of the between-word connections. Figure 4.9 shows one part of the pre-compiled language network for Resource Management. It illustrates the connections of *what's* $\rightarrow$ *a* and *what's* $\rightarrow$ *the*. Connections through `SIL` (silence) represent pauses between two words, while those such as `TS(AH,DH)e` $\rightarrow$ `DH(TS,DX)b`, `TS(AH,AX)e` $\rightarrow$ `AX(TS,B)s`, and `TS(AH,AX)e` $\rightarrow$ `AX(TS,F)s` attempt to capture the strong between-word co-articulation in fluent speech.

For each input utterance, the sex is usually determined first, automatically based on average VQ distortions [57, 126]. Suppose $\{f_k^{(sex)}\}_k$ are the codeword density functions for gender sex. Given a segment of speech $\mathbf{X}$, the likelihood of the speech produced by the given sex can

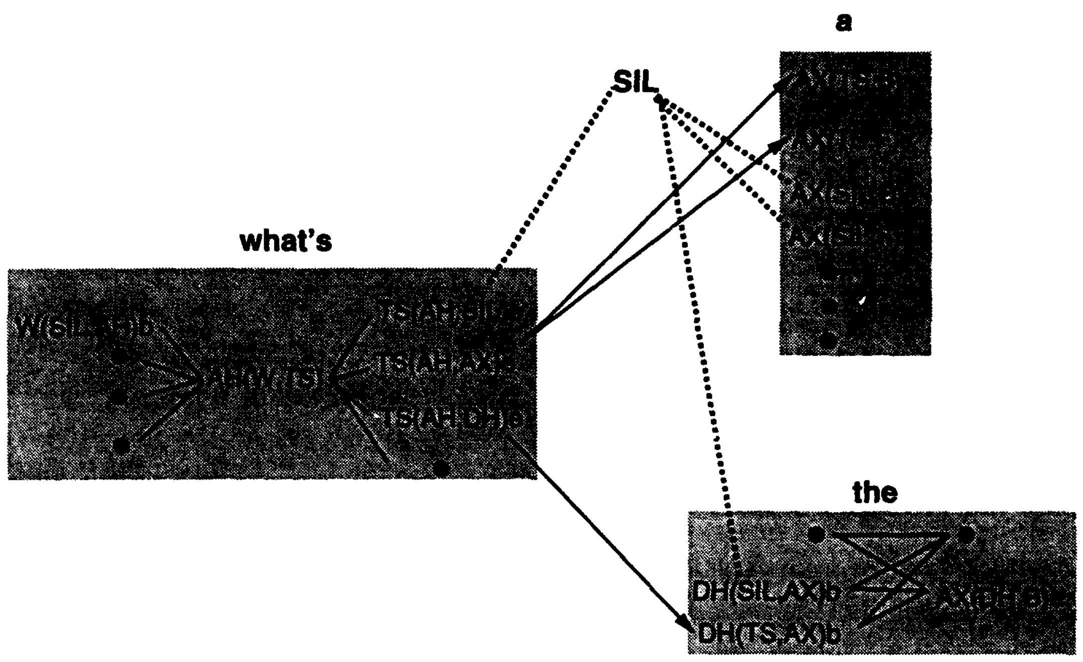


Figure 4.9: Part of the language network in Resource Management, with between-word triphone connections.

be approximated by

$$L(\mathbf{X}|sex) = \prod_{\mathbf{x}\in\mathbf{X}} \sum_{k\in\eta(\mathbf{X})} f_k^{(sex)}(\mathbf{x}) \tag{4.1}$$

where $\mathbf{x}$ is one component frame of segment $\mathbf{X}$ and $\eta(\mathbf{x})$ contains the few codewords that best match the acoustic frame $\mathbf{x}$. If $L(\mathbf{X}|male) > L(\mathbf{X}|female)$, the input speech is assumed to be produced by a male; otherwise, it is from a female speaker. Our experiences on RM show that the accuracy of this sex determination algorithm is above 99% [8].

After the sex is determined, between the two gender-specific acoustic models only the model of the determined sex is activated during recognition. Viterbi beam search is used to find the highest scoring state sequence in the language network. The score of a particular word sequence θ evaluated by a given utterance $\mathbf{X}$ is a weighted summation of the acoustic score and language score:

$$score(\theta|\mathbf{X}) = \log Pr(\mathbf{X}|\text{HMM}(\theta)) + \ w \ \log Pr(\theta) \tag{4.2}$$

where the language weight w is an ad-hoc system parameter that balances the importance of the two knowledge sources (the acoustic model and the language model) [9].

[8]Note that for sex determination, the density value shouldn't be normalized and background silences should be excluded from X.

[9]The language weight is usually empirically tuned. Typical values for a bigram language model range from 3 to 10.

The Three-Pass Search Algorithm for Large-Vocabulary Tasks

Recent work on search algorithms for continuous speech recognition has focused on three dimensions: large vocabularies, long distance language models and detailed acoustic modeling. As large-vocabulary tasks like WSJ are concerned, pre-compiled language networks are no longer practical because of the explosion in the number of between-word connections. Instead, the decoder has to dynamically expand the connections during search. To use complex models such as long-distance language models and between-word triphone acoustic models, several systems have been proposed to use the N-best paradigm [127, 137, 122]. In this N-best paradigm, Viterbi beam search is used as a fast match to generate the N-best hypotheses for each test utterance, with a simple acoustic model (e.g., within-word triphones only) and a simple language model (e.g., bigram). A multi-pass rescoring using more complex models (e.g., the between-word triphone acoustic model and trigram language model) is subsequently applied to these hypotheses to produce the final recognition output. One problem in this paradigm is that decisions made by the initial phase are based on simplified models. This results in errors that the rescoring cannot recover from if the correct hypothesis is not in the N-best list. For very-large vocabularies and very long sentences, it will require an exponential number of hypotheses to be generated in order to guarantee that the correct word sequence is in the list. Another problem is that the rescoring procedure could be very expensive as many hypotheses may have to be rescored. The challenge is to design a search strategy that makes the appropriate compromises among memory bandwidth, memory size, and computational power.

To meet this challenge we incrementally apply all available acoustic and linguistic information in three search passes [3] in SPHINX-II. Pass one, with a bigram language model, is a standard time-synchronous left-to-right (forward) beam search which, as usual, produces the best-matched word sequence. The most detailed acoustic model, including between-word triphones, is used during the initial pass. This is made feasible by carefully handling the fan-in and fan-out acoustic scores at word boundaries. Although this first pass is designed as part of a three-pass decoder, it can be configured to run by itself when more complex language models are not available. By applying the most detailed acoustic model early, we observed significant error reduction over the N-best approach in which the detailed acoustic model is applied later in the rescoring stage [3], with a negligible increase in computational complexity.

When a long-distance language model is available or when N-best hypotheses are desired (e.g., as input for a natural-language parser), pass one, using a bigram (or word-pair) language model, can also produce sets of word ending times and scores. In this configuration, two additional passes are involved. Pass two, guided by the results from pass one, is a right-to-left (backward) beam search which produces possible word beginning times and scores. Pass three is an A^* search that combines the word lattices from pass one and two to produce N-best hypotheses. During the A^* search, scores from the forward pass are used as the scores up to a certain time frame; scores from the backward pass are used to estimate the scores for the future paths. Then the partial hypotheses with the best estimated total score for the whole utterance is chosen to be expanded next. SPHINX-II currently supports trigrams and long-distance language models in the A^* search. Applying these more detailed language models in the decoder rather than as a rescoring scheme has been very profitable [26]. Since this thesis works on acoustic modeling exclusively, only the first pass is activated for all the experiments conducted in the next chapter.

Our goal of maximizing the recognition accuracy with a minimal increase in computational complexity is satisfactorily met with this three-pass incremental search. Applying the most detailed acoustic model in the forward/backward beam search gives us more accurate word lattices. Delaying the long-distance language model until the A^* search greatly simplifies the decoding process when compared with applying it directly in the beam search. Moreover, since long-distance language models are used during the A^* search, we produce a significantly better set of top N hypotheses than with simple language models. Compared with the N-best rescoring paradigm, we don't restrict ourselves to the linear N-best hypotheses produced by a simple language model, which may have pruned away the correct hypothesis. Our principle is to incorporate as many knowledge sources as possible and apply them as early as possible so that we don't produce errors that are never recoverable by later stages.

4.3 Summary

The 1000-word Resource Management task and the 5000-word closed-vocabulary Wall Street Journal task with non-verbalized punctuations are the main tasks experimented on in Chapter 5. With the standard word-pair grammar, RM's test-set perplexity is about 60. The development-set perplexity for the 5c-nvp WSJ task is 118.

The SPHINX-II system is designed for very large vocabulary, speaker-independent continuous speech recognition. The front-end consists of four sets of acoustic features, including first-order and second-order differenced mel-scale frequency cepstra. It is based on sex-dependent semi-continuous hidden Markov models. Between-word co-articulation is modeled by position-dependent between-word triphones. The senone techniques discussed in Chapter 3 are the key component of SPHINX-II explored in this thesis. Figure 4.10 is a block diagram of SPHINX-II [47]. Among the three passes in the decoder, pass one is the standard forward beam search with the most complex acoustic models applied. Pass two is a backward beam search and is activated only if the third pass, the A^* search, is activated. The A^* search is activated only if a long-distance language model is available or the N-best hypotheses are desired. Since this thesis concentrates on acoustic modeling exclusively, only the first pass is activated in all the experiments conducted in the next chapter. The unified stochastic engine (USE) is not used in this dissertation. For a discussion on USE and the language-weight optimization, see [50].

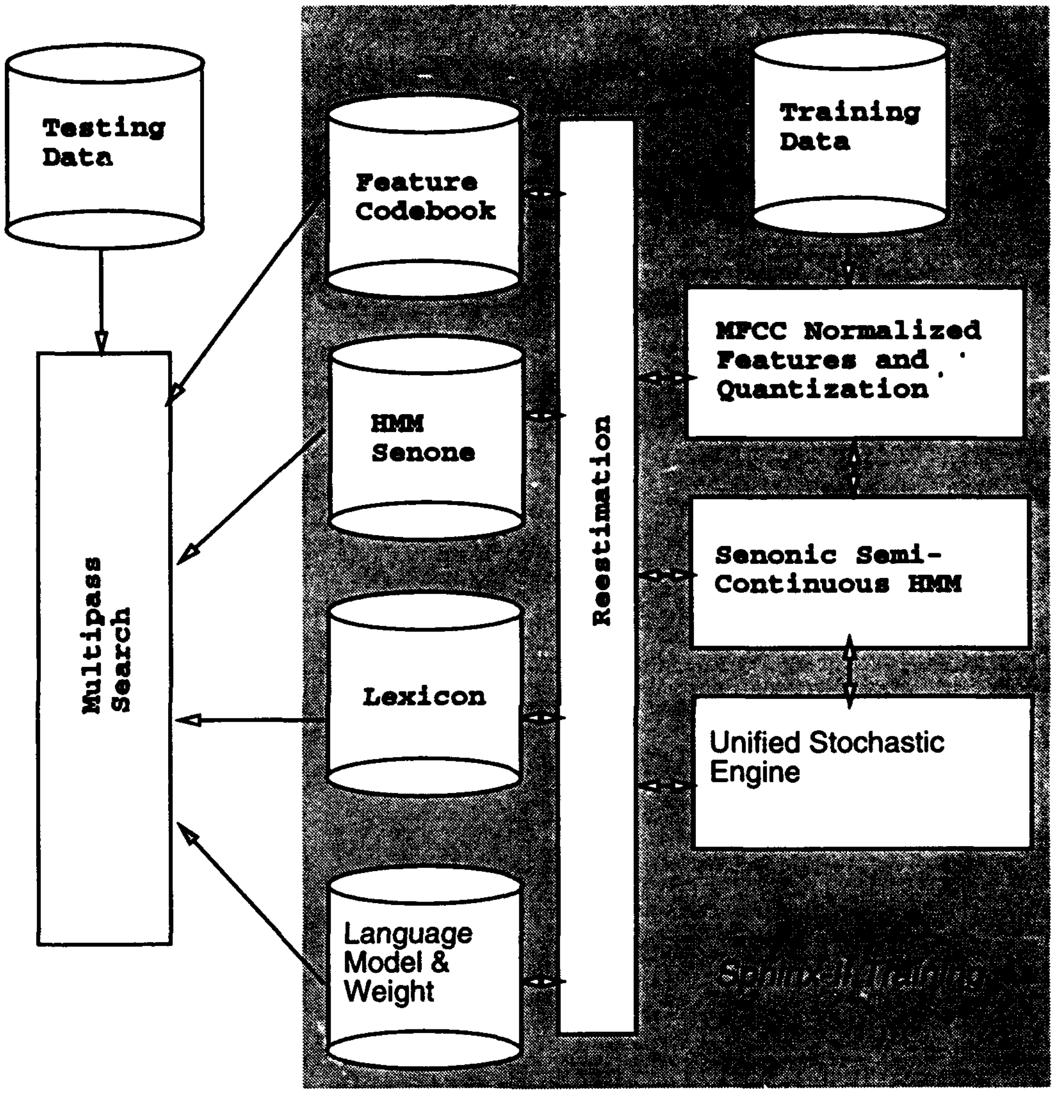


Figure 4.10: The block diagram of SPHINX-II.

Chapter 5

Experiments and Discussions

This chapter will study the performance of the subphonetic modeling techniques presented in Chapter 3 by conducting a series of experiments on the ARPA speech corpora described in Section 4.1. Specifically, these experiments include:

- Applying the SDM with the phone-dependency constraint to Resource Management.
- Applying the SDM with both the phone-dependency and state-dependency constraints to Wall Street Journal.
- Applying the decision-tree based senones to Wall Street Journal.
- Applying the hybrid approach to unseen-triphone modeling to Wall Street Journal.
- Applying the state quantization algorithm to pronunciation learning.
- Applying senonic SCHMMs with phone-dependent VQ codebooks to Wall Street Journal.

When describing the experiments in each section, we will indicate the section in which the concept or technique is elaborated in Chapter 3. Readers are encouraged to review the appropriate section on a certain subphonetic modeling technique before reading the experiment. All of the experiments reported here are conducted using SPHINX-II. For an overview of the SPHINX-II speech recognition system, see Section 4.2. Each time an experimental result is presented, a discussion follows. Some sections will compare the performance on the same data set using different techniques. The last section of this chapter summarizes the discussions and overall experimental performance.

5.1 The Shared-Distribution Model with Phone-Dependency

The motivation of both generalized triphones and SDMs is to share the system parameters across different triphone models so that the number of free parameters is reduced, enabling the parameters to be well trained with a limited amount of training data (Section 3.1). The first experiment presented here demonstrates the "mis-average" problem associated with generalized triphones, by providing a significant error reduction on the Resource Management task using the shared-distribution model. RM (instead of WSJ) was chosen as the first experiment because the WSJ task was not yet well defined and the WSJ speech corpora were not available when we started the SDM work in 1991. RM consists of 997 words in the lexicon with 3990 training utterances from 105 speakers and 600 testing utterances from another 20 speakers. There are 7549 triphones, including position-dependent between-word triphones, in the training corpus.

The senone mapping table for the SDM is generated in this case by the bottom-up state clustering algorithm described in Figure 3.5. To provide a fair comparison, the generalized-triphone clusters used in this experiment are also generated using the bottom-up clustering algorithm, with the entire triphone model as the clustered object. Unseen triphones are modeled by monophones in both systems [1]. The pre-compiled language network described in Section 4.2.4 is used during decoding. The gender decision algorithm (also described in Section 4.2.4) is used to activate the appropriate set of models (choosing between male and female models). The error measure used is the word error rate, which includes word insertions, deletions, and substitutions. In addition to comparing the performance of generalized triphones versus shared-distribution models, we will also examine the contents of several clusters generated by both the triphone-clustering and state-clustering algorithms to show the differences between the two clustering behaviors.

5.1.1 The Generalized-Triphone Model

The phonetic HMM topology used in the generalized-triphone system is shown in Figure 5.1. Although there are 7 states per model, there are only three distinct output distributions associated with the transitions. The output distributions are labeled as beginning, middle, and end as illustrated in the figure. The labeling of the lower five arcs are phone-dependent.

To obtain the generalized-triphone mapping table, the triphone clustering shown in Figure 4.8(a) is activated. For simplicity, the triphone models used for clustering are one-codebook

[1]Actually the testing utterances in RM contain no unseen triphones at all because the word pairs in RM test sets are all covered by the training data. However, some incorrect word sequences during search may require triphones which are not covered by the training data.

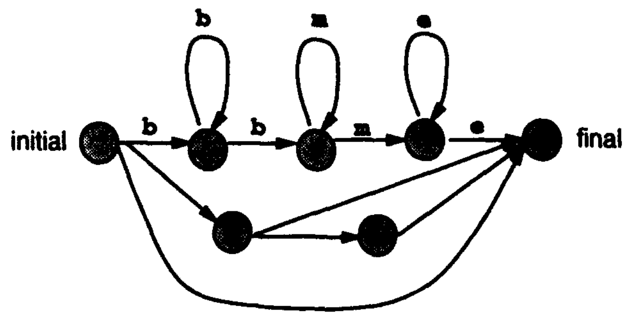


Figure 5.1: The phonetic HMM topology used in the generalized-triphone system.

DHMMs. To speed the clustering procedure, the phone-dependency constraint described in Section 3.1.3 is enforced; that is, clustering across different phones is prohibited.

After clustering, sex-dependent SCHMMs are trained separately from the male and female training data, as depicted in Figure 4.8(b). Varying the number of generalized triphones and tuning the language weight w in (4.2), the optimal performance is obtained by using 1100 generalized triphones for each sex. Table 5.1 shows the word error rates on the test set, using the word-pair grammar (perplexity 60) and no grammar respectively.

language model	word error rate
word-pair	4.7%
no grammar	19.5%

Table 5.1: Word error rates of the 1100 generalized triphones on the 600 utterances consisting of Feb89 and Oct89 evaluation sets.

5.1.2 The Shared-Distribution Model

For the SDM, the triphone clustering in Figure 4.8(a) is replaced with the distribution clustering which generates a senone mapping table. The generalized-triphone model has three output distributions for each HMM. With distribution sharing, we have the luxury of increasing the number of distributions for each model to five in order to have more detailed models. We rely on the distribution clustering procedure to tie redundant states to the same senone. Figure 5.2 shows the 5-state Bakis topology we used for the shared-distribution model. The labeling for the output distribution of each transition is dependent on the source state. The final state depicted here, which has no outgoing arcs, is added to ease implementation.

The phone-dependency constraint and one-codebook triphone DHMMs are again used for the distribution clustering. Varying the total number of shared distributions (or senones) from

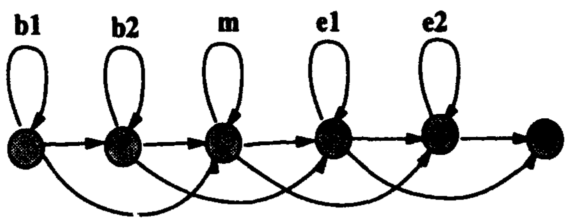


Figure 5.2: The new phonetic HMM topology used in the SDM. The labeling for the output distribution of each transition is dependent on the source state.

3500 to 5500, a series of experiments were conducted to compare with the generalized-triphone models. Table 5.2 shows the word error rates on the test set, using the word-pair grammar. The relative error reduction rates measured against the baseline system (generalized triphones) are also computed.

system	# of output distributions/senones	word error rate	error reduction %
Generalized Triphone	3300	4.7%	—
SDM	3500	4.2%	11%
SDM	4500	3.8%	19%
SDM	5500	4.1%	13%

Table 5.2: Word error rates of different systems on the 600 utterances consisting of Feb89 and Oct89 evaluation sets, using the word-pair grammar.

From these experiments, we can see that SDMs outperform generalized-triphone models. When 1100 generalized-triphone models are used in the baseline system, there are 3300 output distributions in total. With about the same number of parameters, the use of 3500 senones reduces the error rate by 11%. This demonstrates that SDMs have more accurate representations because the "mis-average" problem is avoided. When we increase the total number of senones from 3500 to 4500, the error rate is reduced by about 20% in comparison with the baseline system. As shown in Table 5.2, further increase in the total number of senones to 5500 does not give us more improvement, perhaps because of insufficient training data.

Since the SDM has more states per HMM than the generalized triphone model, our improvements might not come from the distribution clustering, but from the increased number of states within each triphone. To clarify, we also test the 1100 generalized triphone models, each with the same 5-state Bakis topology. The results are shown in Table 5.3. The word error rate is 4.6% for the same test set, which is essentially the same as the the baseline system. As there are 5500 distributions in the 5-state generalized-triphone system, we include the SDM with the same number of parameters in the table. Once again, the SDM outperforms the generalized-triphone model. This demonstrates that distribution sharing *does* contribute to our improvements.

system	total distributions	error rate
3-distribution generalized triphone	3300	4.7%
5-state generalized triphone	5500	4.6%
5-state SDM	5500	4.1%

Table 5.3: Word error rates using generalized triphones with different topologies and using shared distributions with the 5-state topology.

Table 5.4 shows the performances of the 3-distribution generalized triphones and 5-state SDMs without any grammar, where any word can be followed by any word with an equal probability. In both systems, we model *word* durations by the same set of Gaussian distributions (independent of HMM parameters) in order to enhance the performance. To obtain a word duration, we compute phone durations first based on a set of well-trained models and Viterbi alignment on the training corpus. Word durations are then defined as the summation of the component phone durations.

We see that the error reduction is not as high as in the word-pair grammar case. This is perhaps because the language perplexity is so high that without a very strong influence from the acoustic model, it is hard to see a dramatic improvement. Another possible explanation is that the phone durations we used in the experiment were estimated by using within-word generalized triphone models, with the 3-distribution topology. Modeling senone-dependent duration in the 5-state topology may enhance the SDM performance.

system	word error rate	error reduction %
1100 generalized triphones	19.5%	—
SDMs with 4500 senones	17.9%	8%

Table 5.4: Word error rates without any grammar.

In terms of parameter reduction, the generalized-triphone approach reduces the number of models from 7549 to 1100. Since each model has 3 distributions, the reduction ratio in the number of distributions is $(7549 * 3)/(1100 * 3) = 6.86$. In the SDM, the number of distributions drops from $7549 * 5$ to 4500. Thus the reduction ratio is $7549 * 5/4500 = 8.39$ in the SDM approach.

5.1.3 Behaviors of Triphone Clustering versus State Clustering

To understand the behaviors of triphone clustering versus distribution clustering, we examine several clusters generated by both techniques. This section compares the optimal configurations of both systems: the 1100 generalized-triphone models, each with 3 distinct output distributions, and the 5-state shared distribution models with 4500 senones. Both systems combine similar HMM output distributions together, but the shared-distribution model allows more flexibility for sharing. Based on the RM training data, Table 5.5 shows the numbers of triphones, generalized triphones, and shared distributions (senones) for each phone. Note that under the phone-dependency constraint two distributions/triphones are merged only if they represent the same phone. Note also that the silence model (`SIL`) is context-independent; therefore, it is not clustered [2]. Among the 50 phones listed in Table 4.2, AXR, BD, GD, and ZH are not used in RM because there are few cases, if any, in which they are absolutely needed. In addition, the phone EN is used in RM to represent AX N.

Table 5.6 lists several clusters generated by both clustering algorithms. Triphones in the same sub-table are in the same generalized triphone cluster. The notation $P.i$ in the table indicates the i-th senone for phone P. For example, `EY.111` represents the 111th `EY` senone. The last column shows the number of distinct senones that particular triphone used.

In Table 5.6(a), it is reasonable to see that triphone `EY(L,S)` and `EY(L,SH)` are in the same generalized-triphone cluster as they have the same left-context. Unfortunately, triphone clustering must overlook the differences of right-contexts due to the fact that the front parts of these two models are very similar. On the other hand, the distribution clustering approach can keep these right-context differences despite the fact that similar parts are merged. This clearly demonstrates that sharing at the subphonetic level provides more freedom and accuracy in parameter sharing. Besides demonstrating that dissimilar states no longer share output distributions, this example also shows that 5 distinct senones are needed for each of the two triphone models. Examples like this are very common. Figure 5.6(b) shows another one.

There are cases when output distributions of two triphones are not shared at all, even when they have similar contexts. For example, both the left and right contexts of the two triphones in Table 5.6(c) are the same. The only difference is their locations — one of them appears within a word like *thr<u>ea</u>t* and the other appears at the beginning of a word like *<u>a</u>t* in the context *things are <u>a</u>t someplace* [3]. The fact that the distributions are not shared may be partly due to dramatic acoustic transitions at the word boundary in continuous speech, and partly

[2]When non-speech models, like *door bump, inhalation*, etc., are used for spontaneous speech, they are also context-independent and are not clustered. A phone surrounded by non-speech sounds is considered to have silence contexts.

[3]We had by mistake transcribed *at* as /EH TD/ instead of /AE TD/ in our RM lexicon.

because of the fact that the word-boundary triphones do not occur as frequently as within-word triphones. Actually when we analyze this case more carefully, we find a more encouraging explanation. The word *at* is the only word in RM which begins with EH and then is followed by TD. That is, triphone EH(?,TD)b (? means any phone) is trained exclusively by the word *at*. Since we by mistake transcribed *at* as /EH TD/ instead of /AE TD/, triphone EH(R,TD)b and EH(R,TD) should have been quite different models. The distribution clustering discovers the difference and decides to keep them completely apart, while the triphone clustering fails to find the difference.

From Table 5.6(c), we also observe that the first three states of EH(R,TD) all share the same senone. This indicates that three distinct distributions are sufficient to model its acoustic variations. Thus, the distribution clustering procedure is able to elegantly "squeeze" redundant distributions inside an HMM by tying them to the same parameter.

phone	EH	AH	AO	EY	OW	L	R	W	D	Z
# of triphones	211	52	67	178	161	327	251	124	179	385
# of generalized triphones	57	17	16	35	29	48	50	16	22	32
# of senones	211	58	52	129	100	178	171	67	109	142
phone	AE	IH	IY	UH	AX	IX	AA	UW	AW	AY
# of triphones	138	148	309	12	431	119	126	119	27	80
# of generalized triphones	35	34	44	3	63	32	28	28	7	18
# of senones	127	133	162	13	247	144	108	102	31	58
phone	OY	Y	EN	ER	M	N	NG	CH	JH	B
# of triphones	2	59	10	250	252	468	43	89	95	141
# of generalized triphones	1	9	3	40	28	53	9	9	10	15
# of senones	6	35	12	147	125	250	43	44	35	74
phone	DH	G	K	P	T	F	S	SH	TH	V
# of triphones	71	97	249	171	200	237	462	84	128	179
# of generalized triphones	12	11	33	21	23	22	40	11	13	23
# of senones	60	54	126	82	108	91	188	45	54	110
phone	HH	DD	PD	TD	KD	DX	TS	SIL		
# of triphones	109	194	34	269	75	40	97	–		
# of generalized triphones	9	19	3	37	11	11	10	–		
# of senones	50	94	16	167	43	51	48	–		

Table 5.5: The numbers of triphones, generalized triphones, and senones for each phone, when 1100 generalized triphones and 4500 senones in total are determined.

In Table 5.6(d), we see that the first four states of D(Z,EY) and D(S,EY) share the same senones because they are preceded by similar fricatives. However, their last states map to different senones because the D in *Tues<u>d</u>ay* is in an unstressed syllable while the D in *s<u>t</u>ate* is in a stressed one. This shows that the greatest difference between these two phones is at the transition from D $\rightarrow$ EY. The last two rows of Table 5.6(d) show that for D(S,EY) and D(S,AA) both distribution clustering and triphone clustering yield the same result. That is, these two triphones resemble each other so much that one model is adequate to represent both.

triphone	example	b_1	b_2	m	e_1	e_2	# distinct senones
EY(L,S)	~ *pl<u>a</u>ce*	EY.111	EY.112	EY.27	EY.109	EY.114	5
EY(L,SH)	~ *l<u>a</u>tion*	EY.111	EY.112	EY.27	EY.68	EY.21	5

(a)

triphone	example	b_1	b_2	m	e_1	e_2	# distinct senones
AE(K,S)	~ *c<u>a</u>stle*	AE.113	AE.16	AE.76	AE.105	AE.122	5
AE(K,Z)	*c<u>a</u>srep* ~	AE.113	AE.16	AE.76	AE.53	AE.52	5

(b)

triphone	example	b_1	b_2	m	e_1	e_2	# distinct senones
EH(R,TD)	*thr<u>ea</u>t*	EH.65	EH.65	EH.65	EH.61	EH.113	3
EH(R,TD)b	*are <u>a</u>t*	EH.63	EH.185	EH.63	EH.64	EH.207	4

(c)

triphone	example	b_1	b_2	m	e_1	e_2	# distinct senones
D(Z,EY)	*Tues<u>d</u>ay*	D.94	D.105	D.95	D.103	D.96	5
D(S,EY)	*s<u>t</u>ate*	D.94	D.105	D.95	D.103	D.87	5
D(S,AA)	*s<u>t</u>art*	D.94	D.105	D.95	D.103	D.87	5

(d)

Table 5.6: The senone labeling of several triphones. Triphones in the same sub-table are in the same generalized-triphone cluster. The notation $P.i$ indicates the i-th senone among all the senones for phone P.

In addition to comparing the clustering behaviors, we also examine the contents of several clusters. Table 3.2 in Section 3.1.4 and many other examples show that the k-th states of different triphone models often share the same senone. In other words, the distributions that are in the same cluster are mostly from the the same k-th states of different triphones. Interestingly, the first two output distributions of an HMM are never merged with the last two distributions. For the 4500-senone system, we observe that the average number of distinct senones a triphone uses is 4.66. This shows that most of the triphone models keep 5 unique output distributions as Table 5.6 demonstrates. As each triphone tends to keep more output distributions, the SPHINX three-distribution topology must not have had sufficient details. Simply increasing the number of states for each generalized-triphone model does not help as Table 5.3 shows, because the "mis-average" problem becomes more serious. Therefore, in addition to increasing the model complexity, we have to use distribution clustering for parameter reduction in order to achieve both robust and accurate acoustic representation.

5.2 The Shared-Distribution Model with Phone-Dependency and State-Dependency for Large Vocabulary Tasks

As we examined in the previous section, the k-th states of different triphone models often share the same senone. This suggests us to use the state-dependency constraint to guide and speed the clustering process. When we scale up to very large vocabulary tasks, the bottom-up clustering algorithm becomes intolerable due to the exponential computational complexity introduced by the shift step. To reduce the input size for the clustering algorithm, the state-dependency constraint is introduced in Section 3.1.3. With both the phone-dependency and state-dependency constraints, clusterings for different state locations of different phones are independent and can be executed in parallel. To relate the clusterings across different state locations for the same phone, we define the *entropy increase with neighboring-state information* in Section 3.1.3

$$\Omega^*(\mathcal{T}_1(s), \mathcal{T}_2(s)) = \sum_i w_{s,i}\, \Omega(\mathcal{T}_1(i), \mathcal{T}_2(i))$$

as the distance measure for comparing two distributions. In this section, we apply the bottom-up distribution clustering with the above distortion measure and both phone-dependency and state-dependency constraints to the 5000-word closed-vocabulary Wall Street Journal with non-verbalized punctuations (WSJ 5c-nvp) task. Further experiments in later sections will be based on the same system setup. Comparison between different techniques will be made as each new experiment is explained.

5.2.1 System Setup

The weighting function $w_{s,i}$ of the distance measure in the distribution clustering procedure is defined arbitrarily as

$\lvert s-i \rvert$	0	1	2	3	4
$w_{s,i}$ ratio	1.00	0.30	0.15	0.10	0.01

As $w_{s,i}$ is basically inversely proportional to the topological distance between state s and state i, $w_{s,i}$ thus emphasizes the importance of closer states. For example, for the middle state (s_3) of the 5-state Bakis model, the distortion measure is

$$\begin{aligned}\Omega^*(\mathcal{T}_1(s_3), \mathcal{T}_2(s_3)) &= \frac{0.15}{1.90}\,\Omega(\mathcal{T}_1(s_1), \mathcal{T}_2(s_1)) + \frac{0.30}{1.90}\,\Omega(\mathcal{T}_1(s_2), \mathcal{T}_2(s_2)) \\ &\quad + \frac{1.00}{1.90}\,\Omega(\mathcal{T}_1(s_3), \mathcal{T}_2(s_3)) + \frac{0.30}{1.90}\,\Omega(\mathcal{T}_1(s_4), \mathcal{T}_2(s_4)) \\ &\quad + \frac{0.15}{1.90}\,\Omega(\mathcal{T}_1(s_5), \mathcal{T}_2(s_5))\end{aligned}$$

The training corpus of the WSJ task consists of 7200 utterances, distributed equally between male and female speakers. There are more than 23,000 triphones, including position-dependent between-word triphones, in the training corpus. Triphones that occur only once or twice are discarded to avoid arbitrary and thus noisy clustering decisions. This leaves us 17,610 triphones, whose one-codebook DHMMs are estimated for the state clustering in Figure 4.8(a). During the following four-codebook SDM training, the discarded triphones are replaced by context-independent phones. Transition probabilities are inherited from the context-independent phones to save computation and memory. The 5-state Bakis topology is adopted.

For the WSJ 5c-nvp task, the lexicon consists of 4986 words whose pronunciations are derived from the Dragon System dictionary [4]. In addition, we put about 500 alternative pronunciations in our dictionary [5], such as the alternative pronunciations with parentheses in

NEW	N UW
NEW(2)	N Y UW
WHEN	HH W EH N
WHEN(2)	W EH N

The WSJ 5c-nvp task uses a backoff bigram language model with a perplexity of 118. In the backoff model, a word pair which does not occur in the bigram training texts is assumed to have a floor probability proportional to the unigram probability of the second word. Therefore for the purpose of decoding, all possible within-word triphones and between-word triphones crossing all word pairs are constructed in advance. This results in about 50,000 triphones. Among them, only 17,610 triphones have a mapping in the bottom-up senone mapping table; the rest are modeled by monophones. Among the three passes in the decoder (see Section 4.2.4), only the first pass (the left-to-right Viterbi beam search) is activated since no long-distance language model is used and thus A^* search in unnecessary. Note that unlike the decoder for Resource

[4]Dragon Systems Inc. is a company in Newton, MA. Most ARPA-supported systems derive the phonetic pronunciations for the WSJ 5c-nvp lexicon from the same dictionary supplied by Dragon.

[5]However, experimental results show that these manually added alternative pronunciations do not affect the recognition accuracy.

Management, where a language network is pre-compiled before search starts, the three-pass decoder expands the network as search is progressing. This minimizes the memory requirement for large vocabulary tasks, with a negligible increase in computation.

Two data sets are tested — the development set consists of 367 utterances and the November 1922 evaluation set has 330 utterances. For each utterance, the left-to-right Viterbi beam search is run separately on male SCHMMs and female SCHMMs, each producing a best-matched word sequence. The word sequence with the highest score is output as the recognized sentence.

5.2.2 Performance Evaluation

Based on the standard acoustic training corpus, we find the most robust and sensitive parameters by tuning the number of senones to achieve the best performance on the Nov92 test set, as Table 5.7 shows. The error increase from 7000 senones to 8000 senones illustrates the maximum number of parameters that the given training data can train reliably. The 7000-senone system, the best among the three configurations we experimented, performed with a word error rate of 7.5% on the development set.

# of senones	parameter reduction	Nov92	Dev
6000	(17610 * 5) / 6000 = 14.68	9.1%	–
7000	(17610 * 5) / 7000 = 12.58	8.2%	7.5%
8000	(17610 * 5) / 8000 = 11.01	8.8%	–

Table 5.7: Word error rates on the Nov92 WSJ 5c-nvp test set and the Development set, using bottom-up clustered SDMs with different numbers of senones. Unseen triphones are modeled by context-independent phones.

5.3 Decision-Tree Based Senones for Modeling Unseen Triphones

In the bottom-up clustering algorithm it is difficult, if not impossible, to find appropriate senones to model unseen triphones. Therefore, in the experiments described in the previous section, unseen triphones were modeled by monophones.

To be able to model unseen triphones with senones, this section presents the experiment using the decision-tree based senones. The corresponding technique explanation is in Section

3.2. The experiment is based on the same system setup as the previous section. 7000 senones are used since that is the best configuration from the previous experiments. Both seen (about 17,600 triphones) and unseen (about 33,000) triphones are modeled by senones in a unified fashion. Although the number of unseen triphones is almost double the number of seen triphones, a large portion of them are between-word triphones crossing word pairs with backoff language probabilities. In fact, there are on average only 2.6 unseen triphones per utterance in the November92 evaluation set. The development set has 2.2 unseen triphones per utterance on average. The Nov92 set contains a few sentences that need more unseen triphones than the remaining utterances. Figure 5.3 elaborates the steps to build the senonic decision-tree recognizer with unseen triphones represented by senones.

1. Train one-codebook DHMMs for the 17,610 seen triphones, using the forward-backward algorithm.

2. Generate the senone mapping table $\mathcal{M}(\mathcal{T}, s)$ by the top-down tree classification algorithm, with both phone-dependency and state-dependency constraints, using the one-codebook DHMMs generated at step 1. This table maps each state s of each seen triphone $\mathcal{T}$ to a senone label.

3. Train four-codebook sex-dependent SCHMMs with $\mathcal{M}(\mathcal{T}, s)$ for all of the triphones $\mathcal{T}$ in the training data (except the discarded ones).

4. Find the appropriate senones for the remaining 33,000 unseen triphones by traversing the trees created at step 2. This enlarges the mapping table $\mathcal{M}$ to be $\mathcal{M}'$ which includes mappings for unseen triphones.

5. Use the enlarged mapping table $\mathcal{M}'$ and the trained SCHMMs for decoding.

Figure 5.3: The steps to build the senonic decision-tree recognizer with unseen triphones represented by senones.

At Step 3 of Figure 5.3, the discarded infrequent triphones are replaced by monophones during the SCHMM training. Alternatively, they can be mapped to senones by traversing the decision trees to enable effective use of the training data. However, representing these discarded triphones with senones during training did not affect our experimental performance because the amount of data for these discarded triphones was very small. Table 5.8 shows the results on the Nov92 and Dev sets using the 7000 decision-tree based senones. To make a comparison with the bottom-up clustering approach, the baseline results presented in the previous section using 7000 senones are also included in the same table.

The last column of Table 5.8 lists the number of unique senone sequences for all the

clustering method	# triphones used	unseen-triphone modeling	Nov92 set	Dev set	overall	# unique senone sequences
bottom-up	17610	monophones	8.2%	7.5%	7.8%	14872
top-down tree	50000	senones	7.1%	7.5%	7.3%	13270

Table 5.8: Word error rates on the WSJ 5c-nvp task, using 7000 senones. Unseen triphones are represented by monophones in the bottom-up clustering approach, but are represented by decision-tree based senones in the tree-classification approach.

triphones used. This is equivalent to the true number of distinct triphone models in the system because the same senone sequence in different triphones actually defines the same model [6]. For the sake of discussion, let's name the model defined by a senone sequence a "sephone" model. The relationship between triphones and sephones is thus many-to-one, just like the relationship between triphones and generalized triphones. It is obvious that with the same number of parameters, the senonic system (with either bottom-up clustering or top-down classification) is able to form many more sephones (and thus more detailed models to distinguish any small dissimilarity between different triphones) than the number of generalized triphones. For example, had 7000/5 decision-tree based allophone models been used, the system would have still had 7000/5 allophone models even when unseen triphones were added. However, the number of unique sephones affects the computation time during decoding. When evaluating the scores for all the states of all surviving triphone models at each time frame, the decoder actually evaluates all the states of all sephones first. To obtain the scores for any triphone model, the corresponding sephone is accessed. Fortunately, the sephone (or generalized-triphone) evaluation occupies only a small portion of the decoding time. Furthermore, the beam actually becomes smaller when more accurate models are provided and thus decoding cost is only increased negligibly.

With about the same number of sephones, the top-down tree classification performs slightly better than the bottom-up clustering approach because of better modeling for unseen triphones. It performs significantly better than the bottom-up clustering on the Nov92 set because Nov92 requires more unseen triphones for the correct word sequences. The following subsection will make a fair comparison between the two clustering approaches by representing unseen triphones with monophones in both approaches. Section 5.3.2 will elaborate the effect of modeling unseen triphones with detailed parameters.

[6]Recall that transition probabilities are context-independent.

5.3.1 Bottom-Up versus Top-Down Clustering

To support our belief that the bottom-up clustering algorithm is able to find a better configuration for seen triphones than the top-down tree classification is, we compare the performance of both approaches, by representing unseen triphones with monophones. Results on Nov92 and Dev sets are shown in Table 5.9.

clustering method	unseen-triphone modeling	Nov92 set	Dev set	overall error rate	# unique senone sequences
bottom-up	monophones	8.2%	7.5%	7.8%	14872
top-down tree	monophones	8.4%	8.0%	8.2%	8921

Table 5.9: Word error rates on the WSJ 5c-nvp task, using 7000 senones. Unseen triphones are represented by monophones in both systems.

As expected, the top-down classification not only performs slightly worse, but the number of unique sephones is much smaller than with the bottom-up clustering approach. This demonstrates that the bottom-up clustering algorithm indeed provides greater freedom in reconfiguration in the hope of finding the optimal one. Although the tree classification does not perform as well as the bottom-up clustering on seen triphones, it reduces the degradation to a minimum. This is primarily accomplished through the help of linguistic questions which provide human expert knowledge when the HMM statistics are not reliable due to insufficient training data. As more complex composite questions are allowed (growing the "simple" tree as deep as possible), the decision-tree classification gains more and more freedom in reconfiguration. More importantly, the binary linguistic questions generalize the classification so that we are able to model new triphones easily, using the decision-tree based senones. The better modeling on unseen triphones increases the performance of the decision-tree based senones to be at least as good as that of the bottom-up clustering approach, as shown in the previous subsection.

5.3.2 Effects of Modeling Unseen Triphones

The strength of the senonic decision tree resides in its ability to model unseen triphones using detailed parameters. To understand the contribution of modeling unseen triphones with senones, here we analyze the results from the 7000 decision-tree based senones with unseen triphones being represented by monophones and senones respectively. Figure 5.4 illustrates the relative error rate reduction, in terms of the number of unseen triphones contained in each utterance. Table 5.10 shows the exact word error rates corresponding to the figure and the number of qualified utterances among these 697 utterances (Nov92 + Dev). For the overall set, the error rate reduction by modeling unseen triphones with senones is 11%. When there are at least 5 unseen triphones in each test utterance, we are able to reduce the error rate by more than 20% [7]. Even for utterances that contain no unseen triphones, we believe the improved modeling for unseen triphones helps the decoder prune those wrong paths in which unseen triphones are needed.

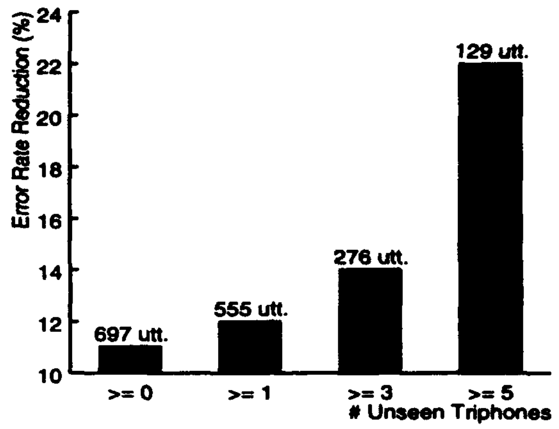


Figure 5.4: Error rate reduction using senones instead of monophones to represent unseen triphones.

# unseen triphones	# utterances	unseen triphones ⇒ monophones	unseen triphones ⇒ senones	error rate reduction
≥ 0	697	8.2%	7.3%	11%
≥ 1	555	8.5%	7.5%	12%
≥ 3	276	9.9%	8.5%	14%
≥ 5	129	9.7%	7.6%	22%

Table 5.10: Word error rates of the 7000 decision-tree senones on WSJ 5c-nvp, with unseen triphones being represented by monophones and senones respectively.

5.3.3 Tree Examples for Markov State Classification

After obtaining the encouraging results using decision-tree based senones, it is interesting to see what questions are asked in the tree. Recall that in the algorithm to automatically generate a decision tree in Figure 3.10, we manually locate a set of "basic" linguistic questions which can be asked of the left-context or right-context phone of a triphone. The set of basic linguistic

[7]Actually the three-pass decoder prunes at the first phone of a word for easy implementation, rather than at the second phone which is the correct pruning when between-word triphones are used. Since most unseen triphones are between-word triphones, some additional error reduction made by modeling unseen triphones with detailed parameters may be blocked due to this early pruning.

questions we used here is the same one as Hon used [42] and is listed in Appendix D for convenience. The tree growing algorithm expands the tree by automatically finding the best composite question for each node. In Chapter 3, we showed an example tree to classify the first state of all K-triphones in Figure 3.9. Figure 5.5 shows another tree to classify the last state of all K-triphones. It also highlights the path that triphone K(L,AX) goes through. Both trees are modified from the decision trees in the above experiments, by cutting the deeper layers. As expected by linguists, the left context is more important than the right context for the first state of a triphone and the right context has stronger influence for the last state of a triphone. This demonstrates that the entropy combining weights $w_{s,i}$ are able to select appropriate questions according to different state locations.

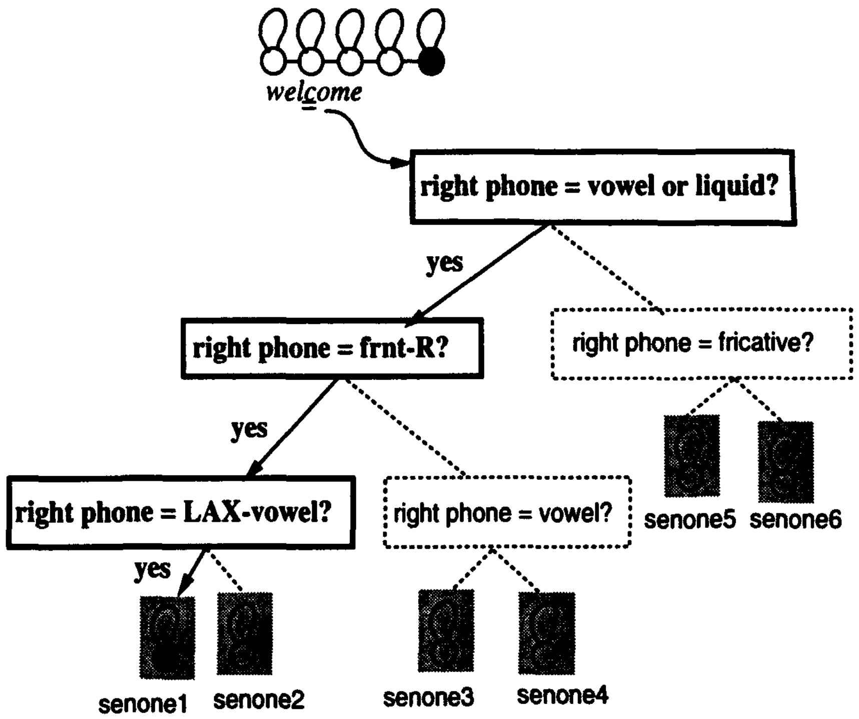


Figure 5.5: A decision tree for classifying the last state of K-triphones. The 5th state of the K-triphone in "welcome" is mapped to the first senone among all the senones in this tree.

For a complete comparison, Figure 5.6, 5.7 and 5.8 are the decision trees for classifying the second, third and fourth states of all K-triphones respectively. Again, the closer a state is to the left side, the more questions about the left-context phone there are at the top levels of the tree, and vice versa. We also observe that similar questions are asked for these middle states. Appendix E shows a complete tree for classifying the first state of all AA-triphones.

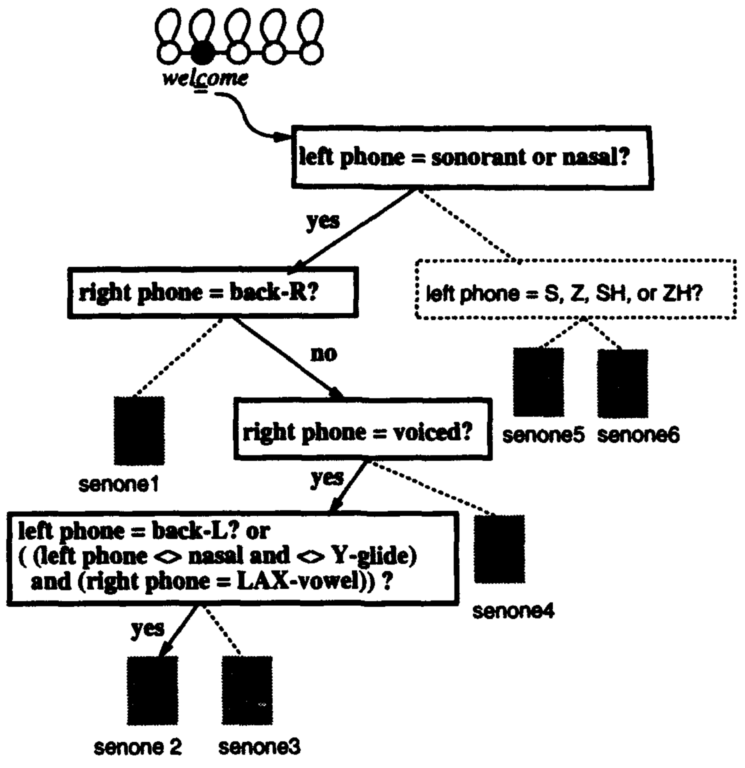

welcome
left phone = sonorant or nasal?
yes
right phone = back-R?
left phone = S, Z, SH, or ZH?
no
senone1
right phone = voiced?
senone5
senone6
yes
left phone = back-L? or
((left phone <> nasal and <> Y-glide)
and (right phone = LAX-vowel)) ?
senone4
yes
senone 2
senone3

Figure 5.6: A decision tree for classifying the second state of K-triphones.

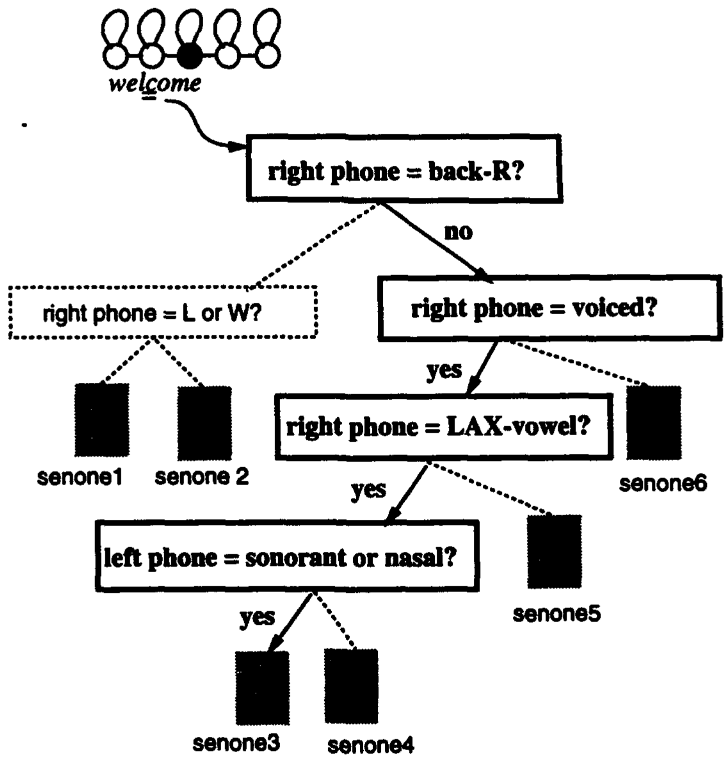


Figure 5.7: A decision tree for classifying the third state of K-triphones.

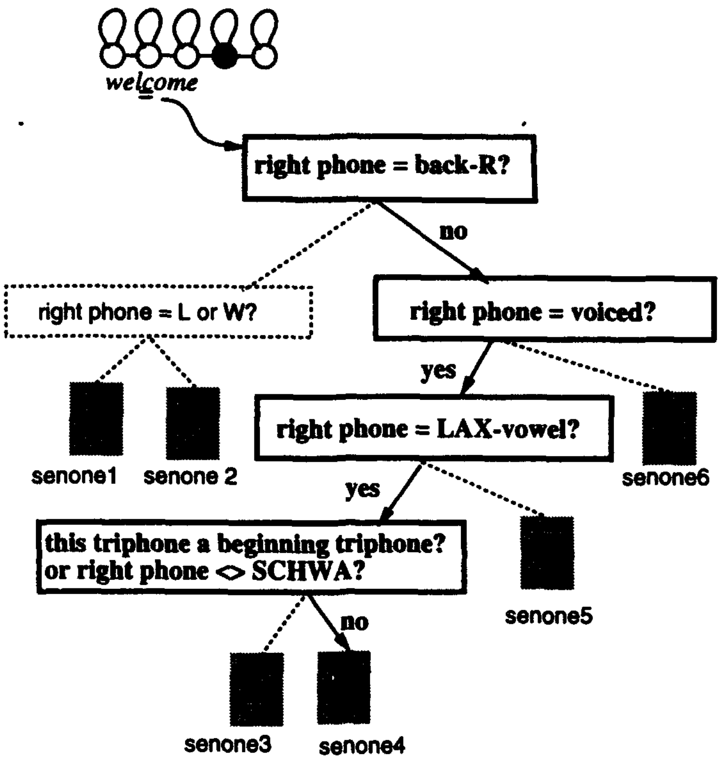


Figure 5.8: A decision tree for classifying the fourth state of K-triphones.

5.4 State Quantization for Modeling Unseen Triphones

The drawback of the top-down tree classification is the limited capability of re-configuration, which often results in a sub-optimal classification, as the experimental results illustrate in Table 5.9. Therefore, in Section 3.3.1 we proposed a hybrid approach in which seen triphones share senones according to the bottom-up clustering configuration while unseen triphones share senones through decision-tree based generalized allophone models whose states are "quantized" with the senone codebook created by the seen triphones.

In this section, we will present the experimental performance of the hybrid approach and make a comparison with the decision-tree based senones discussed in the previous subsection. In an effort to obtain optimal performance, we also experimented with four-codebook triphone SCHMMs for the clustering in order to provide more accurate models for computing the entropy increase. Comparison between the one-codebook and four-codebook clusterings will be made in Section 5.4.2.

5.4.1 Hybrid vs. Purely Decision-Tree Based Senones

Using the same system setup as Section 5.2.1 describes, the hybrid approach requires the complex steps illustrated in Figure 5.9.

Comparing Figure 5.3 and 5.9, we notice the system complexity in the hybrid approach. In our experiments, we first fixed the number of senones generated by the bottom-up clustering procedure to be 7000. Then we tuned the number of generalized allophone models created by the decision tree within the range of 2500 to 4000. By experimental result, the set of 3500 generalized allophone models was the best configuration for state quantization. Unfortunately, this complex hybrid system does not perform better than the purely decision-tree based senones, as shown in Table 5.11, especially when there are many unseen triphones in the test set (the Nov92 set). This is partly because the "mis-average" problem introduced in the decision-tree based generalized allophones is not recoverable. In addition, state quantization errors, which are unavoidable as in all quantization processes, may be a dominant factor.

clustering method	unseen-triphone modeling	Nov92 set	Dev set	# unique senone sequences
top-down tree	senones	7.1%	7.5%	13270
hybrid	senones	7.4%	7.3%	16300

Table 5.11: Word error rates on the WSJ 5c-nvp task, using 7000 senones. Unseen triphones are represented by senones in both systems. Both the purely decision-tree based senones and the hybrid approach yielded statistically the same performance.

5.4.2 Four-codebook SCHMMs for State Clustering

To offer more information for state clustering/classification in order to get the best performance, we experiment with 4-codebook SCHMMs for generating the mapping table. To make feasible the training of 17,610 four-codebook triphone SCHMMs, we segment the training data into phones using an existing set of HMMs and the Viterbi alignment algorithm. Then 50 phone-based training runs are executed separately instead of one sentence-based training. Each of these phone-based training runs train all the triphones representing the same phone.

Unfortunately, this 4-codebook mapping table did not yield a satisfactory improvement, as shown in Table 5.12, where the the word error rates using the 1-codebook mapping table are also included for a clear comparison. Perhaps the one-codebook DHMM has already offered enough information in terms of the similarity measure for output distributions, even though it is not as detailed as the 4-codebook SCHMM. We also notice that using 4-codebook SCHMMs

1. Train one-codebook DHMMs for the 17,610 seen triphones.
2. Generate the senone mapping table $\mathcal{M}$ by the bottom-up clustering algorithm, with both phone-dependency and state-dependency constraints, using the one-codebook DHMMs generated at step 1.
3. Generate decision trees for classifying *allophones*, with only the phone-dependency constraint, using the same one-codebook DHMMs.
4. Train one-codebook DHMM shared-distribution models with $\mathcal{M}$. This creates the senone codebook to be used for state quantization (step 6).
5. Train one-codebook DHMM generalized allophone models with the generalized-triphone mapping table created at step 3.
6. Quantize the Markov states of the generalized allophone models (step 5) with the senones trained at step 4.

   During decoding, the 33,000 unseen triphones first traverse the decision trees to find appropriate generalized allophone models. Then the corresponding sequence of senones resulting from state quantization is used to represent the unseen triphone. This enlarges the mapping table $\mathcal{M}$ to be $\mathcal{M}'$ to include senone mappings for unseen triphones.

   To quantize the Markov state of the generalized allophone model, the associated output distribution is used to represent the state since after tree classification we have enough data for training each generalized allophone model. Then the cross entropy measure described in Section 3.3.3 is used to compute the distance between the output distribution at state s of generalized allophone $\mathcal{T}$ and a senone $\mathcal{S}$. That is,

   $$\mathcal{M}'(\mathcal{T},s) = \underset{\mathcal{S}\ \in\ \Phi_{p,s}}{\operatorname{argmin}} \{\ \chi(\mathcal{T}(s),\mathcal{T}(s)+\mathcal{S})\ +\ \chi(\mathcal{S},\mathcal{T}(s)+\mathcal{S})\ \}$$

   where $\Phi_{p,s}$ is the set of senones for the s-th states of all the triphones which represent the same phone as $\mathcal{T}$.
7. Train four-codebook SCHMMs with $\mathcal{M}$.
8. Use $\mathcal{M}'$ and the SCHMMs for decoding.

Figure 5.9: The steps required to build the hybrid system for modeling unseen triphones with senones.

for the clustering in the hybrid approach yields more improvement than for the decision-tree classification. This may inform us that we should have used more composite questions to allow more configuration freedom to take full advantage of the more detailed acoustic model. In theory, given any configuration created by the bottom-up agglomerative clustering, we can always build an equivalent tree using an oracle, with each leaf corresponding to each cluster generated by the bottom-up clustering algorithm, as long as we allow the complexity of the composite questions to be unlimited and provide a sufficient number of detailed simple questions (e.g., each categorical question contains only one phone). At that extreme, the decision-tree clustering converges to the bottom-up agglomerative method. Since the performance of the hybrid and the decision-tree approaches here are essentially the same, we conjecture that the decision-tree based senones can be guaranteed to be no worse than the hybrid approach when we allow more composite questions for the decision tree.

clustering	unseen-triphone	1-codebook		4-codebook	
method	modeling	Dev	Nov92	Dev	Nov92
bottom-up	monophones	7.5%	8.2%	7.0%	8.0%
top-down tree	monophones	8.0%	8.4%	–	–
hybrid	senones	7.3%	7.4%	6.7%	7.0%
top-down tree	senones	7.5%	7.1%	7.1%	7.0%

Table 5.12: Word error rates on the WSJ 5c-nvp task, using respectively 1-codebook DHMMs and 4-codebook SCHMMs for the clustering procedure.

Interestingly, reading the above table we find that the November evaluation set favors unseen-triphone modeling. We believe it is mostly likely because there are more unseen triphones in certain utterances in the November set than in the development set.

Based on these intensive experiments, we conclude the following guidelines for parameter sharing:

- Use purely decision-tree based senones for modeling both seen and unseen triphones. To make the decision-tree based mapping table robust with seen triphones, more complex composite questions are preferred.
- Use 4-codebook DHMMs for constructing the decision tree. Four feature codebooks are favored since they offer more detailed models. DHMMs instead of SCHMMs are used because training SCHMMs is much more expensive both in computation and disk storage. To take full advantage of the more accurate models, again more complex composite questions or finer categorical simple questions should be used.
- Although we found no performance difference when cross validation was used, Hon showed that when a vocabulary-independent system is pursued (training on a general-

purpose speech corpus but testing on a specific task), cross validation is preferred to make the decision tree robust [42].

5.5 State Quantization for Senonic Baseforms

As shown in Figure 3.14, senones can be shared not only across phonetic models, but also across word models. In this section, we will present our experimental results which automatically learn senonic baseforms for words, when word training utterances are available. For the description of the senonic-baseform construction algorithm, see Section 3.3.2.

5.5.1 Pronunciation Learning with Many Training Samples

To start experiments for pronunciation learning, we used the speaker-independent Resource Management *spelling* task, whose utterances spell the words in RM in the continuous speech mode. That is, the vocabulary in this task contains only the 26 English letters. No language grammar is used. There are 1132 training sentences from 100 speakers and 162 testing sentences from 12 new speakers. In the decoder, the language network is pre-compiled as with RM since this is a small task and a pre-compiled network can speed the search. To test an utterance, the VQ-based distortion described in (4.1) is used to decide the gender first. Then the model with the selected sex is activated for Viterbi decoding.

The Baseline System

For the baseline system, we train *sex-dependent word-dependent* phonetic SCHMMs. Word-dependent phones [28] are a compromise between word modeling and phone modeling. Although phone models are used, these phone models are *word-dependent*. In other words, a phone model used for one word (e.g., the IY phone in word "*b*") has different parameters from the model of the same phone in another word (e.g., the IY phone in word "*c*"). Like word models, word-dependent phone models can model word-dependent phonological variations, but require considerable training and storage. However, word-dependent phone modeling is more robust than word modeling because when a word has not been observed frequently, its parameters can be interpolated with that of context-independent phone models. For example, although the IY phonemes in word "*b*", "*c*", "*d*", etc. are all represented by different models, they can all be interpolated with the context-independent IY model.

Fifty-six (56) word-dependent phonetic models are used in total in the baseline system. No between-word triphones are used because we find between-word coarticulation modeling is not effective for a small task like this with short and stressed words only. This baseline system achieved a word error rate of 11.3% (including insertion errors), as shown in the first row of Table 5.13. This baseline system provides sex-dependent senones for learning senonic baseforms.

system	word error rate	error reduction %
phonetic baseform	11.3%	–
senonic baseform	9.6%	15%

Table 5.13: Word error rates on the spelling task, using phonetic baseforms and senonic baseforms.

The Automatically Learned Senonic Baseform

To prepare for the learning of senonic baseforms, the 1132 training sentences are segmented into words by a set of existing HMMs with the Viterbi alignment algorithm. For each word and each gender, we split its sex-dependent training data into several groups by a DTW clustering procedure according to their acoustic resemblance. Different groups represent different acoustic realizations of the same word. For each word group, we estimate the word model by the Baum-Welch formula and compute a senonic baseform as Figure 3.15 describes. The number of states of a word model is arbitrarily decided to be equal to 75% of the average duration. The simple Euclidean distance is used as the distortion measure in state quantization because we believe each letter must have enough occurrences. Here we allow arbitrary senone sequences for senonic baseforms in order to give the maximum freedom in the automatic learning process. Note that all the male output distributions in the baseline system constitute the male senones; female output distributions in the baseline system are the female senones.

Next we calculate the predicting ability of the senonic word model $M_{w,g}$, constructed from the g-th training group of word w:

$$\text{predict}(M_{w,g}) \;=\; \sum_{\mathbf{x}_w \notin \text{group}_g} \log P(\mathbf{x}_w|M_{w,g}) \;\;/\; \sum_{\mathbf{x}_w \notin \text{group}_g} |\mathbf{x}_w|$$

where $\mathbf{x}_w$ is an utterance of word w.

For each word and each gender, we pick two senonic baseforms that have the best predicting abilities. That is, each letter has four senonic baseforms in total; two for male, two for female.

The pronunciation of each word utterance in the training set is then labeled by:

$$\text{pronunciation}(\mathbf{x}_w) = \underset{M_{w,g}}{\text{argmax}} \; \{ P(\mathbf{x}_w | M_{w,g}) \}$$

After the training data are labeled in this way, we re-train the system parameters by using exclusively the learned senonic baseforms for all 26 letters. During testing, each letter for each gender is represented by two senonic baseforms in the search space. As shown in Table 5.13, the word error rate is reduced by 15% with this automatically learned senonic baseform [8]. This shows that when a sufficient number of training samples are available to estimate each word model, the SQ algorithm does not cause quantization errors. Note that the number of system parameters remains the same as that in the baseline system. No additional parameters are needed for additional baseforms. A task of this sort cannot be improved by adding alternative phonetic pronunciations, since there are none. However, it can be improved through the application of automatically learned senonic baseforms, which in this case are purely acoustic-data driven.

This result indicates that senonic baseforms can capture detailed pronunciation variations for speaker-independent speech recognition. It also illustrates one of the best applications of senonic baseforms — tasks containing words that have ample training tokens and do not have good alternative phonetic pronunciations. The above algorithm not only learns the pronunciations automatically, it also labels the training data automatically so that re-training can be done whenever necessary.

5.5.2 Pronunciation Learning with Few Training Samples

Dictation applications usually start with a speaker-independent system. However, new words often appear during different users' usage. Therefore, new-word samples are often speaker-dependent. A natural application of the senonic baseform is to generate speaker-dependent word models for new words. New-word detection in large vocabulary systems is a difficult problem and it alone could be another thesis, although reasonable solutions could be obtained for limited categories in small tasks [4]. Therefore, this section assumes that possible new words are already detected and a few utterances for the new word are given. When a new word occurs, it is reasonable to ask the user to speak the same word a few times in order to enable the system to learn its pronunciation. Therefore the goal here is to build speaker-dependent acoustic models for new words without increasing the number of system parameters.

Due to their close relationship with acoustic data, senonic baseforms are particularly suitable for jargon, acronyms such as IEEE (pronounced as *I-Triple-E*), CAT-2 (pronounced as *cat-two*)

[8]When no re-training was enforced, we still observed 12% error reduction.

aaw	arctic	asuw	asw	average
c1	c2	c3	c4	c5
capable	capacity	casrep	casreped	casreps
casualty	china	fleet	fleets	formosa
indian	japan	latitude	latitudes	link-11
longitude	longitudes	max	maximum	miw
mob	mozambique	ntds	nuclear	pacflt
pearl-harbor	propulsion	readiness	solomon	strait
thailand	tonkin			

Table 5.14: RM-ω — the selected new words in Resource Management.

and non-English names whose phonetic transcriptions are difficult to obtain through simple spelling-to-sound rules. These categories are all good candidates in which new words occur.

Experimental Speech Corpus

To verify the potential application of the senonic baseform to speaker-dependent new-word learning, we carried the following experiment [65]. We chose RM as the experimental task as it was the most popular common task we could access at the time of the experiments. Speaker-independent (SI) sex-dependent SDMs, trained from the 3990 SI utterances, were used as our baseline system for this study. The shared distributions constitute the senone codebook. New-word training and testing data are speaker-dependent (SD). The four speakers from the June-1990 test set supplied 2520 SD sentences each. Among the four speakers, bjw and jrm are female; lpn and jls are male. The SD sentences were segmented into words using the Viterbi alignment and treated as isolated speech in both the training and testing conditions.

From the RM lexicon we chose randomly 42 words, denoted as RM-ω, that occurred frequently in the SD speech corpus (so that we would have enough test data) as listed in Table 5.14. These 42 words are treated as new words [9]. For each speaker and each of these words, 4 utterances in the SD data were used as training samples to learn the senonic baseform, and at most 10 other SD utterances were used for testing as isolated speech (For words occurring fewer than 14 times, we used whatever was left for testing). During recognition on these selected new words, no grammar was used.

[9]Actually, they are not new words for the baseline SI system (i.e. the senone codebook) since they are covered by the SI training sentences. On the other hand, the SD data used to construct and test senonic baseforms is exclusive from the SI training corpus. Therefore, we consider the system configuration described here to be a quick prototype to study the senonic baseform on the new-word problem.

The Baseline Shared-Distribution Model

Since the SD data is tested in the isolated-speech mode, it is not necessary to model between-word co-articulation. Therefore, our baseline system used within-word triphone models only. The baseline SDM system was trained from the entire SI training corpus, with sex-dependent models. During testing, all the 997 words are represented by the phonetic SDM, without any grammar. With the hand-written phonetic baseform, the average word error rate was 2.67% for the four speakers, as shown in the first row of Table 5.15. Note that the SI training corpus covers quite a few training samples for all the words in RM-ω (on average 25 samples for each word, each gender). Therefore, this result not only gives the upper bound for the performance of phonetic baseforms, it is even better than a hand-written phonetic baseform without any training sample.

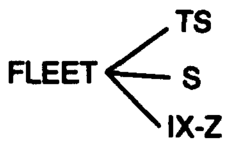
	Female		Male		
	bjw	jrm	lpn	jls	overall
hand-written phonetic baseform	4.4%	3.1%	1.4%	1.7%	2.67%
pilot senonic baseform	7.0%	7.9%	4.7%	5.3%	6.23%
blind suffix senonic baseform	2.3%	4.2%	3.0%	5.0%	3.63%
total testing tokens	385	355	363	357	1460

Table 5.15: Word error rates of the speaker-dependent senonic baseforms.

The Senonic Baseform

For each speaker and each word in RM-ω, we used 4 SD utterances to construct the speaker-dependent senonic baseform. The number of states was set to be 0.8 of the average duration. The SQ approach using semi-continuous output probabilities described in Section 3.3.3 was used during the senonic-baseform construction. Again we allow arbitrary senone sequences for the automatically learned senonic baseforms. The selected 42 words using senonic baseforms were used together with the remaining 955 words using phonetic baseforms to make a vocabulary size of 997. The original set of senones constructed by the baseline system was used directly with the learned senonic baseform; no re-training was imposed. The word error rate was 6.23%, as shown in the second row of Table 5.15. Most of the errors came from the confusion among word derivatives, like CASREP and CASREPED.

To understand how accurate the constructed senonic baseforms are without derivative confusion, we concatenated the senonic baseform of the word origin with all the possible suffix phones as the baseform for the derived words. For example, the baseform of *fleets* became



where the context-independent phone models /TS/, /S/, and the concatenated /IX-Z/ were appended in parallel with an equal probability after the senonic baseform of FLEET. In this way, no training data were used to learn the pronunciations of the derivatives. This blind *suffix senonic* approach significantly reduced the error rate to 3.63%, as the third row of Table 5.15 shows. Still, some errors continued to occur, such as the misrecognition of *casreped* as *casrep* and *max* as *next*. The *max-next* confusion reveals that the senonic-baseform construction algorithm was not sufficiently accurate when only a few training tokens were available. Confusions of the type of *casreped* for *casrep* could be due to improper segmentation during data preparation, in conjunction with the confusion between the phones PD and PD-TD. We are not yet close to reaching our goal of approaching the performance of hand-written phonetic pronunciations.

Interestingly we noticed that the suffix senonic baseforms for speaker bjw were significantly better than the phonetic baseforms. When we analyzed the error reduction, we found it was mainly due to the word *arctic*, which was transcribed as /AA R KD T IX KD/ in the phonetic baseform. The senonic baseform for bjw showed that there was none /KD/ senone in the middle of the word, implying that the middle /KD/ was completely deleted by the speaker. The deletable phone KD designed by linguists could not handle the complete deletion. This shows that senonic baseforms are useful for adapting the system according to speaker idiosyncrasy [10]. A simple way to take advantage of senonic baseforms is to choose whichever is better from the phonetic and senonic baseforms, instead of always replacing the phonetic baseform. The goodness of a baseform is decided by the joint probability of generating the given training utterances. Although this yields a heterogeneous pronunciation system, both baseforms utilize the same set of senones and thus almost no additional difficulty in the decoder is incurred. This way, improper phonetic baseforms can be replaced by senonic baseforms automatically.

Possible Further Improvements

Since we cannot expect many training utterances for new words, the phonetic information revealed from spelling is crucial [96, 13, 4]. Bahl [13] showed that given the spelling L of the new word and utterance samples U, the phonetic baseform of the new word could be derived by

[10]Some systems, such as LIMSI's recognizer in France, apply phonological rules during decoding. For example, *arctic* could be pronounced as /AA R [K] T IX K/, where the brackets mean that the first K phone is optional. However, the senonic baseform is still more flexible for capturing more complicated speaker idiosyncrasy.

decoding the phone sequence R which maximizes $\Pr(R|L, U)$, with decision trees determining the spelling-to-sound rule, $\Pr(R|L)$. With four utterances only, the automatically determined phonetic baseform performed as well as handwritten phonetic baseforms in a 20,000-word speaker-dependent isolated speech recognition.

When there are few training samples, the forward-backward algorithm falls easily to the local optimality closest to the initial model (therefore, uniform initial models are good in the sense that they are unbiased). When the estimated word model is sub-optimal, the Viterbi algorithm gives sub-optimal alignment for the training data, resulting in unrecoverable errors for the SQ algorithm using the aligned data frames. To help decide the senonic baseform, the phonetic transcription deduced from spelling may be used as the initial model, with the hope that the senonic baseform can escape from errors made by the spelling-to-sound rules. The advantage, in comparison with using the phonetic pronunciations directly, is the freedom (based on the acoustic data) to escape from the possibly wrong phonetic transcription which is derived purely from the word's spelling.

Other improvement possibilities for the senonic baseform include the use of state topological locations and senone bigrams during state quantization, and a search algorithm similar to the fenonic baseform defined by (2.1).

As shown in this study, the performance of new-word learning depends strongly on the selected words and the confusability of the vocabulary. For example, if all the selected words are long and acoustically easily distinguishable, it is easy for the automatically learned baseform to approach the performance of the hand-written phonetic baseform. Therefore, to study new-word learning, not only is spelling information important, a common and rich speech corpus of careful design is also needed. Since no such public corpus was available at the time of these experiments, our work on this problem was put on hold, where it remains.

5.6 Senonic SCHMMs with Phone-Dependent VQ Codebooks

To approach the performance of CHMMs without actually going to CHMMs, we relax the mixture-tying condition in SCHMMs from tying all the Markov states to the same set of mixtures (the VQ codebook) to tying only the states which belong to the same phone class. Phone classes can be easily determined manually since there are not many phones in English, or they can be determined automatically as Section 3.4.1. Alternatively, faced with a fixed training corpus, one way to reduce the number of parameters when the number of codebooks is increased is to reduce the VQ level. In this section, we will present our early results with both approaches.

5.6.1 SCHMMs with Eight Phone-Class VQ Codebooks

According to the phone-classification tree in Figure 3.19, we select the top 7 splits. This gives us the eight most distinguishable phone classes. They are:

0	SIL SILb SILe noises
1	AA AE AH AO AW AY EH
2	AX EY IH IX OW OY UH
3	B D DH F G K P TH
4	BD DD GD KD PD TD V
5	CH JH S SH T TS Z ZH
6	DX HH L W
7	M N NG
8	ER AXR R Y IY UW

Note that silences and background noises form a separate class. Markov states of all the triphones which represent the same phone class share the same set of mixture densities (the 256 VQ densities). For our experiments, the hybrid system with 7000 senones and one-codebook DHMMs for clustering in Table 5.12 is used as our baseline. This baseline system had a word error rate of 7.4% on the Nov92 set. To train these eight phone-class dependent VQ codebooks and senones, we initialize them with the universal VQ codebook and senones of the hybrid system. Then we run the forward-backward algorithm for two iterations to adjust both the senones and phone-class dependent codebooks. As mentioned in Appendix B.2, we normalize the density value $f_k^{(p)}(\mathbf{x}_k)$ for each phone class p by the average summation to force the normalized density value into a stable range:

$$\begin{aligned} c_t &= \frac{\sum_{p \in P} \sum_k f_k^{(p)}(\mathbf{x}_k)}{|P|} \\ f_k^{(p)'}(\mathbf{x}_k) &= \frac{f_k^{(p)}(\mathbf{x}_k)}{c_t} \end{aligned}$$

where P denotes the set of phone classes and $|P|$ is the number of phone classes. During training, computation can be saved by computing only the necessary densities as Section 3.4.3 describes, where Φ is a subset of P.

Table 5.16 shows our results on the male and female subsets of the Nov92 evaluation set.

Further increase in the number of phone classes does not give us additional improvement. In fact, when more phone classes were used, we found many codewords were not observed and thus their densities were not re-estimated. For those codewords that did get re-estimated, some of them possibly did not have sufficient training data, either. We thus hypothesize that with

system	Nov92	
	male	female
1 phone class	6.1%	8.6%
8 phone classes	5.8%	7.6%
error reduction %	5%	11%

Table 5.16: Word error rates of the senonic SCHMMs with 8 phone-class dependent VQ codebooks on the WSJ 5c-nvp task. The baseline with one VQ codebook is the hybrid system with the senone mapping table generated from the one-codebook DHMMs in Table 5.12.

either more training data or a reduced VQ size, we should be able to get additional improvement.

5.6.2 SCHMMs with a Reduced VQ Size

The second experiment we tried is to reduce the VQ size from 256 to 128. Since the VQ size was reduced (thus so was the number of parameters), we decided to move to 40 phone classes directly instead of staying at the eight phone classes. These 40 phone classes are determined partly by the phone classification tree in Figure 3.19 and partly by merging infrequent phones (such as BD, GD, and ZH) into similar ones in order to guarantee enough training data. The 40 phone classes are:

0	SIL SILb SILe noises		
1	AA AO	21	JH
2	AE	22	K
3	AH AW	23	L
4	AX	24	M N NG
5	AXR	25	OW
6	AY	26	OY
7	B	27	EY
8	BD PD	28	P
9	GD DD KD TD	29	R
10	CH	30	S
11	D	31	SH ZH
12	DX HH	32	DH
13	EH	33	T
14	ER	34	TS
15	F	35	UH
16	TH	36	UW

17	G	37	V
18	IH	38	W
19	IX	39	Y
20	IY	40	Z

When the mixture sharing constraint is relaxed from 8 phone classes to 40 classes, almost each phone has the freedom to estimate its own VQ prototypes. We hoped to see a significantly different behavior by the dramatic change in the number of phone classes.

The phone-class dependent VQ codebooks and context-independent DHMMs for VQ size 128 were initialized from the segmentation of the training data by Viterbi alignment using a set of existing HMMs. Unfortunately, the performance with this reduced VQ size was similar to the baseline which had only one VQ codebook; there was no improvement at all. Perhaps using 128 prototypes for each of the 40 phone classes still results in too much resolution in the cepstrum space under the 7000-utterance training data. To obtain substantial improvement, we have to find the equilibrium between the number of phone classes and the VQ size. A more complex approach is to have a variable VQ size for each different phone class. The VQ size for a certain phone class can be decided by examining the entries in the output distributions for that phone class (i.e., the weighting vectors in terms of CHMM terminology). Zero entries mean that the corresponding codewords are redundant and should be removed.

Although the work on phone-class or phone dependent VQ codebooks is not successful yet on speaker-independent speech recognition, it is useful for speaker adaptation and channel normalization, where the codebook means are the best candidates for adaptation given a small amount of adaptation data (e.g. speaker-dependent speech or secondary-microphone speech). Speaker adaptation on the universal VQ codebook of the traditional SCHMM had been successfully applied to the RM task [55]. Adapting the phone-dependent codebooks allows the system to learn the phone distribution in the cepstrum domain for a new speaker or a new environment and hence should be more powerful than adapting only one universal codebook. To limit the scope of the investigation, these further studies are left open.

5.7 Summary

In this chapter, we presented the experimental results for the subphonetic modeling techniques described in Chapter 3.

For the SDM created by the bottom-up state clustering procedure, we achieved a 20% error reduction over the generalized-triphone model. This demonstrates that the "mis-average" problem does indeed occur in the generalized-triphone approach. The SDM avoids the "mis-average" problem by clustering at a finer level.

To extend the SDM to very-large vocabulary tasks, we impose both phone-dependency and state-dependency constraints to speed the clustering procedure. Our experiments show that the bottom-up clustering procedure provides more freedom in re-configuration than the top-down tree classification and thus provides a slightly better performance when both systems represent unseen triphones with monophone models. The strength of decision-tree based senones resides in its ability to model unseen triphones using detailed parameters. When unseen triphones were modeled by the decision-tree based senones in the same way as seen triphones, the senonic decision tree outperformed the hybrid approach in which unseen triphones were also modeled by senones but with the state quantization technique. The hybrid approach failed to outperform the purely decision-tree based senones because of the "mis-average" problem in the generalized allophone models and because of possible state quantization errors. Modeling unseen triphones with detailed parameters helps not only identify the correct path when the correct path contains unseen triphones, but also helps prune those wrong paths in which unseen triphones are used. The more unseen triphones the correct path contains, the more error reduction modeling unseen triphones with senones provides. In our experiments, when there were at least 5 unseen triphones in the correct path, the error reduction was above 20%.

The above experiments used one-codebook DHMMs for state clustering for the sake of simplicity. We also tried using 4-codebook SCHMMs for state clustering, in the hope of improving the recognition performance. However, this added model accuracy only offered a small improvement, perhaps because the one-codebook DHMMs already provided a good distance measure for state clustering.

Next we applied the senonic baseform for automatic pronunciation learning. When there were ample training samples, the senonic baseform provided 15% error reduction over the hand-written phonetic baseform, with the same number of parameters. When there were few training samples in the new-word learning scenario, confusion among word derivatives was common. To avoid the derivative confusion, we proposed the incorporation of spelling information, but were not able to complete the line of work. One of the advantages of the senonic baseform is that it shares senones with all the rest existing phonetic models and thus the number of system parameters remains fixed. We can always choose the better model between the senonic and phonetic baseforms for any word. The heterogeneous system does not impose any difficulty in implementation because both baseforms share the same set of senones. In our experiments, we also showed that the senonic baseform was able to adjust to speaker idiosyncrasy, showing the potential application to speaker adaptation.

Finally, the senonic SCHMMs with phone-class dependent VQ codebooks are hopeful but as yet unproven for further acoustic improvement in speaker-independent speech recognition.

Another advantage of multiple VQ codebooks is the potential application to speaker adaptation and channel normalization, which will be our important work in acoustic modeling in the near future.

Chapter 6

Conclusions

This thesis presents several subphonetic acoustic modeling techniques for speaker-independent speech recognition. We will summarize our conclusions based on our experimental experiences with these techniques. Contributions of this thesis are discussed, and finally future directions in acoustic modeling based on this work are presented.

6.1 Conclusions on Subphonetic Acoustic Modeling

Here we summarize our conclusions for each of the subphonetic modeling techniques described in Chapter 3, based on our experimental experiences. We also discuss conditions under which each subphonetic modeling technique is most suitable.

- Parameter sharing at a subphonetic level provides more accurate models than sharing at the entire phonetic unit. Particularly, the shared-distribution model substantially outperformed the generalized-triphone model.
- The bottom-up state clustering algorithm to generate a senone mapping table is most suitable for medial-sized tasks (e.g., 1,000 – 2,000 words) because (1) The bottom-up clustering algorithm is able to find the optimal sharing structure for seen triphones, (2) the computation is tolerable, (3) it is not difficult to design and collect a speech training corpus which covers a high percentage of the triphones needed for a medial-sized task.
- For medial-sized tasks, the state-dependency constraint for state clustering is not necessary. For very large-vocabulary tasks, it can speed the the bottom-up clustering dramatically. The phone-dependency constraint is reasonable as long as there are not many

phones for a certain phoneme. For example, the phones T, TD, and DX all represent the phoneme T. It is more desirable to allow parameter sharing across the phones which represent the same phoneme.

- The strength of the senonic decision tree rests on its ability to model unseen triphones with detailed parameters (i.e. senones). Therefore, it is the method of choice when large-vocabulary dictation systems are built or when a backoff language model is employed. In both conditions, we expect a large number of unseen triphones. Modeling unseen triphones with senones helps not only in identifying the correct word sequence, but also in pruning the incorrect paths in which unseen triphones are needed.

- Cross validation is designed to enhance the robustness of decision trees, especially when a vocabulary-independent system is pursued or complex composite questions are constructed to grow the tree in which case the tree might become tailored too closely to the training corpus. In our experiments, there was neither a cross-vocabulary condition nor did we make the composite question complex. Therefore cross validation had no effect on our performance.

- Because of the complex procedure involved in the training phase, the hybrid approach for modeling unseen triphones by quantizing the Markov states of tree-based generalized allophone models with senones was not worthwhile, although it did offer slightly better performance than the purely decision-tree based senones when there were not many unseen triphones in the test set (e.g., fewer than 2 unseen triphones per utterance).

- It is always better to provide more accurate and detailed models for the state clustering algorithm, either in the bottom-up or top-down fashion, in order to provide a better distance measure. However, training the four-codebook SCHMMs before clustering is much more expensive than the training of one-codebook DHMMs. To reduce the cost and at the same time provide as detailed models as possible, we suggest training four-codebook DHMMs for the purpose of state clustering.

- An accurate senonic baseform can be automatically learned when there are many training samples. Senonic baseforms can be used together with phonetic baseforms without an increase in the number of system parameters because both baseforms share the same set of senones. Therefore frequent words, phrases, and syllables may be better modeled by the more detailed senonic baseforms than by phone-concatenated models.

- We have seen the potential application to speaker adaptation using senonic baseforms in the *arctic* example in Section 5.5.2. However, the senonic baseform learned from only a few training samples is not accurate enough for a large vocabulary task. Possible improvements include the incorporation of spelling information and a search strategy similar to the one used for determining a fenonic baseform, with a senone language model.

- The senonic SDM with phone-class dependent VQ codebooks offered small improvement. To observe further improvement, it requires more work on finding the equilibrium between the number of phone classes and the VQ size, based on a fixed training corpus.

6.2 Contributions

Senones

The most significant contribution of this thesis is the introduction of the subphonetic unit — the senone. By tying parameters across similar Markov states so that the number of parameters can be reduced, we not only make the parameters well trained with limited amount of training data, we also provide more accurate parameter sharing than generalized-triphone models, which sometimes mis-average dissimilar states. A natural effect of parameter reduction is the reduction in the memory requirement, which is important when useful application systems are built.

Given the strong results of this thesis, both BBN [111] and SRI [32] have recently successfully incorporated senones and similar approaches into their systems.

The senonic baseform

We present the concept of treating senones as Markov-state prototypes. With senones in the Markov-state space, any unit of model can be treated as a walk through or permutation of the prototypes. The sequence of the walk through the senones defines the senonic baseform for that unit of speech.

On the English spelling task, we demonstrated a promising result of using automatically-learned senonic baseforms for detailed word modeling. When sufficient training data for a word is available, accurate senonic baseforms can be determined automatically. Compared with hand-written phonetic baseforms, the advantages of the senonic baseform are

- The senonic baseform is purely acoustic-driven and can be derived automatically.
- The number of system parameters remains fixed, even though there are often more states per word in the senonic baseform than the phone-concatenated word model. More states usually imply the ability of accommodating more acoustic variabilities.
- More than one senonic baseform can be learned for each word. The training data can be re-labeled automatically with the senonic baseforms. The system parameters (i.e. the senones) can be re-trained using the re-labeled senonic pronunciations so that the senones can be optimized under the new sharing structure.
- Since the senonic baseform is purely acoustic-driven, it is very suitable for speaker adaptation.

Purely senonic decision trees ve sus Hybrid

The hybrid approach is motivated by better modeling for unseen triphones. However, when the unseen-triphone problem is serious, it is better solved using the decision-tree based senones. We show by experiments that the hybrid approach is unnecessary because it did not offer meaningfully better performance while it required a very complex training procedure.

5-state Bakis topology

With state sharing, we have the luxury of increasing the model complexity and relying on the sharing algorithm to tie redundant states. Most speech recognition systems use three-state phonetic HMMs as their basic models. We modified our phonetic models to have the 5-state Bakis topology when we incorporated the shared-distribution model into SPHINX. Recently BBN has confirmed that it is slightly helpful to increase the number of states per model when senones are used for very large vocabulary tasks [111]. Perhaps when the task scales up, the acoustic confusability is increased and thus more detailed model topologies are needed to accommodate more acoustic variabilities. However, had the complexity of the model topology been increased simple-mindedly without proper parameter sharing, the increased system parameters would have been either under-trained or "mis-averaged" somehow. Only when a proper parameter sharing technique like senones is coupled with the increased topology complexity do we expect a consistent improvement.

The phone-dependent VQ codebooks

We propose the senonic SDM with phone-dependent VQ codebooks with the hope of approaching CHMM performance without actually going to CHMMs. A small improvement was observed when eight phone-class dependent VQ codebooks were used.

Codebook adaptation has been studied for fast speaker adaptation by Huang and Lee [55]. The most important advantage of multiple VQ codebooks is its potential impact on speaker adaptation and channel normalization by adjusting the phone-dependent VQ prototypes. When there is only one VQ codebook as in the traditional SCHMMs, the effectiveness of adapting the codebook only becomes saturated after the number of adaptation utterances reaches the order of 20. To be more effective with respect to a modest amount of additional adaptation data, we can adjust the phone-dependent codebooks.

6.3 Future Work on Acoustic Modeling

Despite the high accuracy (93% word accuracy) on the WSJ 5c-nvp task, the recognition rate is still not satisfactory for real applications, especially when a task larger than 5000 words is considered. In recent years, researchers have been frustrated in trying to make improvement in speaker-independent acoustic modeling. The work in SCHMMs and senones have been the most successful recent improvements.

Our work on the decision-tree based senones builds a paradigm for subphonetic acoustic modeling. However, the state-dependency constraint is undesirable, even though we did observe that states in the same cluster were mostly from the same topological location. It would be better to allow the state clustering algorithm the full freedom to decide if tying some states in the same model would yield a better configuration and result in an improvement. Moreover, as discussed in Section 3.2.2, the tree growing algorithm runs much more efficiently than the bottom-up clustering algorithm which has essentially an exponential computational complexity. This violates the motivation for the state-dependency constraint, which is designed to speed the clustering procedure. Therefore one possible improvement is to release the state-dependency constraint in the tree growing algorithm. Questions like *"is the state under consideration the first state of a model?"* and their composites may need to be added. Carefully re-designing the set of basic linguistic questions and adding *criteria* to constrain or guide the composite questions could also improve the system. When more complex composite questions are used, we believe that cross validation will become more important lest the tree become tailored too closely to the training data. Finally, to consider more contextual effects, we propose to cluster allophones with longer contexts when more training data are available, as IBM [12] has been using the left five and the right five neighboring phonemes.

When a vocabulary-independent system is desired, a rich training corpus covering a larger number of triphones is needed. With the rich training corpus, vocabulary-independent senonic decision trees can be built. When a particular task is chosen, the vocabulary-independent senonic decision trees are used to generate a senone mapping table. Since the triphone set needed for the target task is different from the triphone set formed by the vocabulary-independent training corpus, some vocabulary adaptation techniques are helpful in getting the optimal senone mapping table for the target task. For example, pruning subtrees which do not contain any (or very few) triphone of the target task (these triphones are called *relevant allophones* in [42]), re-constructing the decision trees using only relevant allophones, and so on.

The concept of senonic baseforms can be extended to any unit of speech, not just the word model. It is well know that *there is no data like more data* [99]. However, acoustic modeling cannot be improved by simply adding more data without increasing the parameter complexity. For example, a few thousand utterances are enough to train monophone models well. If a few million utterances are available and monophone models are still used, then we will not be able to obtain increasingly accurate models or achieve higher recognition rates. To obtain significantly better systems, we have to use triphones or even longer-distance context units. The problem with long-context units or multi-phone units is the tremendous number of free parameters. To maintain the number of parameters at a manageable level, the Markov states of the long-context units could be clustered using the state clustering algorithm or they could be quantized with an existing set of senones using the state quantization algorithm. The senonic baseform created by state quantization works best when there are many training samples. Therefore, we can represent frequent multi-phone units, syllables, or phrases using their automatically-learned senonic baseforms without an increase in the number of parameters. The senonic baseforms can be used together with other existing phonetic baseforms with almost no implementation difficulty since they share the same set of senones.

When there are few training samples, the senonic baseform may be improved by incorporating spelling information. A senone language model and a search strategy similar to the one used to determine fenonic baseforms may also be incorporated to enhance the quality of the senonic baseform. The senonic baseform has a great potential application for speaker adaptation, as evidenced in our experiments in Chapter 5.

The strength of the senonic SDM with phone-dependent VQ codebooks relies on the quick codebook adaptation for speaker adaptation and channel normalization, where phone-dependent VQ prototypes can be tailored to the test speaker or recording environment by the Baum-Welch re-estimation formula. Adjusting the phone-dependent VQ codebooks is more effective and accurate than adjusting a single universal VQ codebook. It is well known that the performance of a speaker-independent system performs with about triple the error of a speaker-dependent system [35, 55, 80]. Speech recognition across different recording environments is even worse [2, 74]. Therefore, speaker adaptation and environment normalization are two crucial and probably related research issues for building useful, accurate and robust speech recognition systems. All of these directions mentioned above are important work for improving acoustic models in the near future.

Appendix A

Fundamentals of Hidden Markov Models

As a statistical modeling tool, hidden Markov models were first described in the classic paper by Baum [20]. Shortly afterwards, they were applied to automatic speech recognition independently at CMU [16] and IBM [18, 70]. Until recently, HMMs have become the predominant approach in speech recognition, subsuming dynamic time warping [70, 29, 85, 49] and outperforming neural networks [133, 38, 129, 40] in most speech recognition tasks.

An HMM can be used to represent a specific unit of speech, such as a phoneme or a word. Most speech recognition systems use phonetic HMMs. However, for small vocabulary tasks in which data collection does not require heavy efforts, word HMMs can be used efficiently to obtain high-accuracy [33] speech recognition systems.

This appendix is designed for readers who are not familiar with hidden Markov models. At the end of this appendix is a list of references for further studies of applying HMMs to speech recognition.

A.1 Definition of Hidden Markov Models

Formally, an HMM is defined as

$$\lambda = < S, I, F, \mathbf{A}, \mathbf{B}, >$$

where S is a set of states with transition arcs defined between the states. Associated with each transition from state i to state j is an output distribution, $b_{ij}(\mathbf{x}) \in \mathbf{B}$, which defines how likely a certain event $\mathbf{x}$ in the observation space is going to happen when the transition is taken [1]. That is, whenever a transition is taken, a piece of data is observed (or "output") with a certain probability. In addition, a transition probability $a_{ij} \in \mathbf{A}$ is associated with each arc and specifies the likelihood of transit to state j, given that we are currently at state i. I is the set of initial states, F is the set of final states. Any sequence of observations is output by starting from one of the initial states and ending with one of the final states. $\pi_i \; \forall \; i \in I$ is the probability that we start from state i. Since **A**, **B**, are all probabilities,

$$\begin{aligned}
\sum_{j \in S} a_{ij} &= 1 \quad \forall \, i \in S, \\
\int_{\mathbf{X}} b_{ij}(\mathbf{x}) d\mathbf{x} &= 1 \quad \text{for all transitions } i \text{ to } j \text{ if } \mathbf{x} \text{ is a continuous vector,} && \text{(A.1)} \\
\sum_{k} b_{ij}(k) &= 1 \quad \text{for all transitions } i \text{ to } j \text{ if } \mathbf{x} = \mathrm{k} \text{ is a discrete symbol,} && \text{(A.2)} \\
\sum_{i \in I} \pi_i &= 1
\end{aligned}$$

Starting from a certain initial state (unknown or hidden), we observe a sequence of data. We know these data are emitted (output) by the HMM and that the stochastic process ends at one of the final states. The observed event is the sequence of data output by the HMM; the *hidden* part is the sequence of states the observed data have gone through. In other words, we don't know which sequence of transitions output the observed data.

A.2 Null Transitions

Some transitions may be *null*, meaning no data is output when the transition is taken. A null transition has an associated transition probability, but no output distribution since no data is output by the transition. Null transitions are a convenient notation that can be used to reduce the topology complexity. For example, the two HMMs shown in Figure A.1 are equivalent, where the dotted arc is a null transition.

Null transitions can also be used to unify the initial states and/or unify the final states to simplify implementation, as illustrated in Figure A.2.

[1]Output distributions may also be associated with states instead of arcs. The theory is exactly the same, with only slight modification on notations.

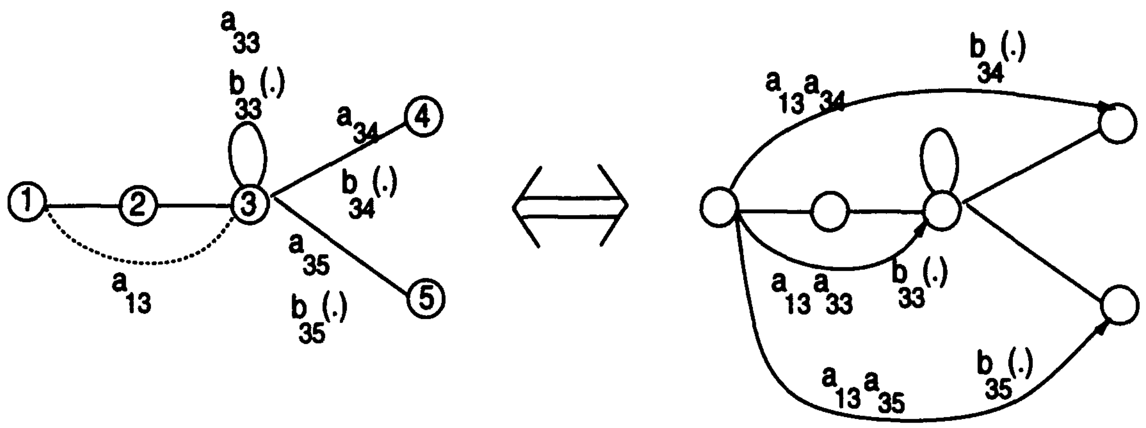


Figure A.1: Two equivalent HMMs, one with a null transition (the dotted arc), the other without. a_{ij} is the transition probability; $b_{ij}(.)$ is the output distribution. The parameters associated with those arcs without any mark in one HMM are the same as those associated with the corresponding arcs in the other HMM.

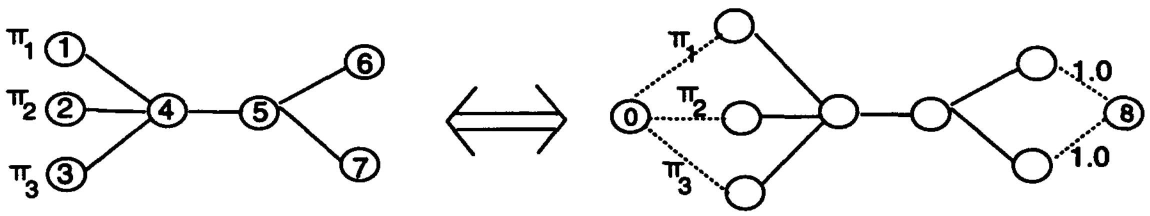


Figure A.2: Using null transitions to unify the initial states and to unify final states. π_i in the left HMM is the probability of being at state i initially; it becomes the transition probability of the null arc in the right HMM.

A.3 Three Types of HMMs

A.3.1 Continuous HMMs

When the observed data, **x**, are n-dimensional vectors in a continuous domain as the speech cepstra are [2], the output distributions become probability density functions (pdfs), as Equation A.1 shows. HMMs of this type are hence called continuous HMMs or CHMMs.

To enable parametric estimation of the pdfs, these pdfs are often assumed to be Gaussian (sometimes even with diagonal covariance matrices) for mathematical simplicity [107]. Other forms that have been explored include the Gaussian mixture density [114], the Gaussian autoregressive mixture density [73], the Richter mixture density [118], and the Laplacian mixture density [102]. For example, when the Gaussian mixture density is used, the probability of generating data **x** given the transition from state i to state j, $Pr(\mathbf{x}|ij)$, becomes

$$Pr(\mathbf{x}|ij) = \sum_{m=1}^{M} w_{ij}^{(m)} \mathcal{N}_{ij}^{(m)}(\mathbf{x}) \tag{A.3}$$

where $\mathcal{N}^{(m)}$ is the m-th Gaussian, $w^{(m)}$ is the mixture weight and $\sum_m w_{ij}^{(m)} = 1$ to make $\int_{\mathbf{X}} Pr(\mathbf{x}|ij)dx = 1$. Suppose there are n CHMMs in a system and each HMM has a arcs. The number of parameters is then na for transition probabilities, plus naM for the densities, assuming that there is one initial and one final state for all HMMs. In practice the number of mixtures, M, is something between 10 to 30.

When the assumptions about the forms of the distributions (e.g. the diagonality) are incorrect or when the distributions are not well behaved (e.g. a Gaussian model is used for distributions that are bimodal), a better training algorithm on the HMM parameters will not guarantee a better recognition accuracy during testing.

A.3.2 Discrete HMMs

Vector quantization (VQ) [39, 97] is a data reduction technique that maps a continuous vector onto a discrete symbol. To be able to perform VQ, a set of prototype vectors has to be created first, by techniques such as K-means clustering [94, 39, 97], based on an inventory of data vectors. A vector quantizer is defined by the prototype vectors together with a distortion measure that estimates the proximity of two vectors. Any input vector is quantized to the

[2]Speech cepstra are n-dimensional vectors that characterize the frequency spectra of the input speech within a short period of time, e.g. 10ms [31].

closest prototype. Thus, instead of dealing with the raw n-dimensional domain directly, we transform the data into finite symbols.

With the VQ technique, when the raw data $\mathbf{x}$ are n-dimensional vectors like speech cepstra, we can quantize the continuous vectors into a finite number of symbols. When the data emitted by an HMM are symbols from a finite set, the output distributions become real probability values, as Equation A.2 shows. HMMs of this type are called discrete HMMs or DHMMs.

DHMMs do not make any assumption about the form of the output distribution. Therefore, they can well represent all possible distributions. Moreover, computing the output probability $Pr(\mathbf{x}|ij)$ becomes only a table lookup, rather than the intensive multiplications and additions required for CHMMs. That is,

$$Pr(\mathbf{x}|ij) = b_{ij}(k)$$

where k is the vector quantized symbol for $\mathbf{x}$.

However, there are two disadvantages. First of all, many more parameters (and thus more training data) are necessary for DHMMs than for CHMMs. With DHMMs, the number of parameters is na plus naL, where n is the number of models in a system, a is the number arcs per HMM, and L is the size of the VQ codebook. Normally L is about 256 versus $M = 20$ in CHMMs. Secondly, VQ distortions are inevitable. A finite set of prototypes cannot represent all possible continuous vectors with 100% precision.

A.3.3 Semi-Continuous HMMs

To relieve the VQ distortion in DHMMs, semi-continuous HMMs (SCHMMs) [53] or tied-mixture HMMs [22, 21] were proposed independently by Huang and Bellegarda. When semi-continuous HMMs are used, the discrete output distribution $b_{ij}(k)$ is retained as the distribution representation. In addition, each VQ codeword k is modeled by a density function $f_k(\mathbf{x})$, which can be explained as the likelihood of vector $\mathbf{x}$ belonging to codeword k. The simple table lookup $b_{ij}(k)$ in DHMMs to find the probability of generating an acoustic vector $\mathbf{x}$ given the transition from state i to state j is then replaced by

$$Pr(\mathbf{x}|ij) = \sum_{k=1}^{L} b_{ij}(k) f_k(\mathbf{x}) \tag{A.4}$$

where L is the VQ level, i.e., the size of the VQ codebook. To save computation, SPHINX-II usually uses only the top 4 best-matched VQ prototypes to represent the input vector, rather than using all the L codewords. Each f_k is assumed to be a single Gaussian with a diagonal covariance in SPHINX-II. Moreover, we found that normalizing the $f_k(\mathbf{x})$'s such that

$$\sum_{k \in \text{top } 4} f_k(\mathbf{x}) = 1$$

increased the stability of our training routines. This is because the density values can be arbitrarily small or large, which may cause floating-point exceptions, especially when a double-precision floating-point representation is not used. The floating-point exception problem becomes serious when multiple independent features are used [52]. Appendix B proves mathematically that normalizing the density is just a scaling factor and will not affect either the Baum-Welch estimation [20] or the time-synchronous beam search [132].

When compared with DHMMs, SCHMMs can be viewed as reducing VQ errors by using the top few codewords instead of only the top one. We can call this "fuzzy" VQ, with the fact in mind that the codebook densities can be jointly optimized with the other parameters, rather than always being fixed as with DHMMs. From the viewpoint of CHMMs, all transitions in SCHMMs are tied with the same set of density functions $\{f_k\}$ to reduce the computation; i.e., $\{\mathcal{N}^{(m)}\}_1^M$ in (A.3) = $\{f_k\}_1^L$ in (A.4). Each transition has the discrete output probability as the weighting in the mixture; that is, $w_{ij}^{(k)}$ in (A.3) = $b_{ij}(k)$ in (A.4). It is the relationship with CHMMs and DHMMs that prompts us the name of "semi-continuous" or "tied-mixture". In this thesis, we use the name SCHMM, although the term "tied-mixture HMM" is interchangeable.

The number of free parameters in SCHMMs is na transition probabilities, plus naL output probabilities, plus L densities. Thus, the total number of parameters is only L more than DHMMs. With essentially no more parameters and modest additional computational costs than DHMMs, SCHMMs have been shown to provide more accurate models for speech recognition [44].

Compared with CHMMs which require the computation of naM densities for every piece of data $\mathbf{x}$, SCHMMs require only the computation of L densities since all transitions are tied with the same mixtures. With much less computation, SCHMMs perform at least as well as CHMMs in terms of recognition accuracy [103].

A.4 Application of HMMs to Speech Recognition

Because they provide a good model for time-series phenomena with stochastic influences, HMMs are a natural representation for speech. Each short segment of speech (e.g, 10ms) is represented by an n-dimensional (e.g. 12) cepstrum vector that characterizes the frequency spectra [31]. The n-dimensional cepstrum may be further quantized to a discrete symbol for DHMMs. The cepstrum or the discrete codeword is the data emitted by HMMs. The output distribution models the distribution of speech events at each transition, and the transition probability models the duration of these events. An HMM can be used to represent any unit of speech, such as a phoneme or a word. Since there are strong temporal constraints in speech, topologies that enforce left-to-right flow are usually used.

Given the definition of hidden Markov models, there are three problems of interest. Existing algorithms are available to solve each of them:

The Evaluation Problem Given a model (λ) and a sequence of observations ($\mathbf{X}$), what is the probability that the model generated the observations? That is, how to evaluate $Pr(\mathbf{X} = \mathbf{x}_1\mathbf{x}_2...\mathbf{x}_T|\lambda)$?

Application — isolated-word speech recognition.

Solution — the forward algorithm [116].

The Decoding Problem Given a model (λ) and a sequence of observations ($\mathbf{X}$), what is the most likely state sequence (Q) in the model that produced the observations? That is, how do we find $\underset{Q}{\operatorname{argmax}}\ Pr(\mathbf{X}, Q|\lambda)$?

Application — continuous speech recognition.

Solution — the Viterbi beam search algorithm [132].

The Learning Problem Given a model (λ) and a sequence of observations ($\mathbf{X}$), what should the model's parameters be so that it has a high (if not maximum) probability of generating the observations? That is, how to find $\operatorname{argmax}_\lambda Pr(\mathbf{X}|\lambda)$?

Application — training of the HMM parameters.

Solution — the iterative forward-backward algorithm or Baum-Welch reestimation [20, 7].

Interested readers are referred to [20, 17, 70, 7, 92, 116, 113, 85, 49] for full explanation of all the above algorithms and more detailed treatments on HMMs for speech recognition. Particularly, [116] is a good start for beginners.

Appendix B

Normalization of the Probability Density in SCHMMs

According to (A.4), the probability of output data $\mathbf{x}$ given a transition from state i to state j is

$$Pr(\mathbf{x}|ij) = \sum_{k=1}^{L} b_{ij}(k) f_k(\mathbf{x})$$

for SCHMMs, where symbols a_{ij}, b_{ij}, and $f_k(.)$ are defined in Appendix A. Since f_k are density values instead of probabilities, values may range from an epsilon small value to a gigantic number, especially when $\mathbf{x}$ is a high-dimensional vector. When multiple independent data features (e.g., cepstrum, delta cepstrum, and the second-order differenced cepstrum) are used, multiplication of multiple (A.4)'s could cause floating-point exceptions with underflow or overflow. To prevent floating-point exceptions, we can normalize these densities before multiplications take place. This appendix will prove that mathematically this affects neither the Baum-Welch reestimation nor the time-synchronous beam search.

This normalization is similar to the scaling [92, 73, 115] of the α and β values during the Baum-Welch computation. We will describe the scaling first before describing the normalization of the density functions. To be able to distinguish the same term under different scaling and/or normalization conditions, we will use superscript $*$ to denote the real mathematic value. Furthermore, to simplify the explanation, we assume that the first state (state 0) is the only initial state and the last state (state $n-1$) is the only final state in an HMM. Moreover, we assume there is only one data feature. Null arcs will be noted only in Section B.1.3 and B.2.3. It is easy to extend the proof to more generally-structured HMMs with multiple data features.

B.1 Scaling of the Baum-Welch Computation

B.1.1 The Forward Pass

In the forward pass of the Baum-Welch reestimation, we compute the α^* cells time-synchronously from the first frame to the last frame:

$$\begin{aligned}
\alpha_t^*(s) &= Pr(\text{reach state } s \text{ after } \mathbf{x}_1\mathbf{x}_2...\mathbf{x}_t \text{ are observed} \mid \mathbf{X}, \lambda) \\
&= \sum_j \alpha_{t-1}^*(j) a_{js} Pr(\mathbf{x}_t|js) \\
\alpha_0^*(0) &= 1 \\
\alpha_T^*(n-1) &= Pr(\mathbf{X}|\lambda)
\end{aligned}$$

assuming there are n states (from 0 to $n-1$) and the input utterance is T frames long (recall that the superscript $*$ is to mean the true mathematical value).

Due to the assumption that data frames are independent, the α^* values would underflow the floating point representation of almost any digital computer after only a few tens of frames have been processed. To prevent underflow, we can either use a logarithm representation for all probabilities or re-scale the probabilities whenever necessary. Section B.3 will mention the usage of the log representation. In this and the following sections, we will demonstrate in detail the usage of scaling. The most popular method of scaling to prevent underflow is to divide all the $\alpha_t^*(.)$ cells by the summation of them after they are computed. To elaborate, let's start from the initial condition and use γ_t to denote the scaling factor at time frame t, and α without the $*$ superscript to denote the scaled value. Because of

$$\begin{aligned}
\alpha_0^*(0) &= 1 \\
\alpha_0^*(s \neq 0) &= 0
\end{aligned}$$

the initial values for γ_0 and α_0 are

$$\begin{aligned}
\gamma_0 &= \sum_s \alpha_0^*(s) = 1 \\
\alpha_0(s) &= \frac{\alpha_0^*(s)}{\gamma_0} = \alpha_0^*(s) \ \forall s
\end{aligned}$$

Next for frame 1,

$$\alpha_1(s) \text{ before scaling} = \sum_j \alpha_0(j) a_{js} Pr(\mathbf{x}_1|js)$$

$$
\begin{aligned}
&= \sum_j \alpha_0^*(j) a_{js} Pr(\mathbf{x}_1|js) = \alpha_1^*(s) \\
\gamma_1 &= \sum_s \{\alpha_1(s) \text{ before scaling}\} \\
&= \sum_s \alpha_1^*(s) = \frac{\sum_s \alpha_1^*(s)}{\sum_s \alpha_0^*(s)} \\
\alpha_1(s) &= \frac{\alpha_1^*(s)}{\gamma_1} = \frac{\alpha_1^*(s)}{\sum_s \alpha_1^*(s)}
\end{aligned}
$$

Looking at one more frame, frame 2,

$$
\begin{aligned}
\alpha_2(s) \text{ before scaling} &= \sum_j \alpha_1(j) a_{js} Pr(\mathbf{x}_2|js) \\
&= \sum_j \frac{\alpha_1^*(j)}{\gamma_1} a_{js} Pr(\mathbf{x}_2|js) \\
&= \frac{\alpha_2^*(s)}{\gamma_1} \\
\gamma_2 &= \sum_s \{\alpha_2(s) \text{ before scaling}\} \\
&= \frac{\sum_s \alpha_2^*(s)}{\gamma_1} = \frac{\sum_s \alpha_2^*(s)}{\sum_s \alpha_1^*(s)} \\
\alpha_2(s) &= \frac{\alpha_2^*(s)}{\gamma_1 \gamma_2} = \frac{\alpha_2^*(s)}{\sum_s \alpha_2^*(s)}
\end{aligned}
$$

By induction it is easy to show that

$$
\begin{aligned}
\gamma_0 &= 1 \\
\gamma_t &= \frac{\sum_s \alpha_t^*(s)}{\gamma_1 \gamma_2 \ldots \gamma_{t-1}} = \frac{\sum_s \alpha_t^*(s)}{\sum_s \alpha_{t-1}^*(s)} \quad \text{for } t = 1..T \\
\alpha_t(s) &= \frac{\alpha_t^*(s)}{\gamma_1 \gamma_2 \ldots \gamma_t} = \frac{\alpha_t^*(s)}{\sum_s \alpha_t^*(s)} \quad \text{for } t = 1..T \qquad \text{(B.1)} \\
Pr(\mathbf{X}|\lambda) &= \alpha_T^*(n-1) = \alpha_T(n-1) \prod_{t=1}^{T} \gamma_t \qquad \text{(B.2)}
\end{aligned}
$$

To sum up, the true α^* value is the scaled α times the product of all the scaling factors from the first time frame through the current time frame. Or equivalently, each $\alpha_t(s)$ is effectively scaled by the sum over all states of $\alpha_t^*(s)$.

B.1.2 The Backward Pass

In the backward pass of the Baum-Welch reestimation, we compute the β^* cells time-synchronously from the last frame to the first frame:

$$\begin{aligned}
\beta^*_t(s) &= Pr(\text{reach state } s \text{ after } \mathbf{x}_{t+1}\mathbf{x}_{t+2}\ldots\mathbf{x}_T \text{ are observed} \mid \mathbf{X}, \lambda) \\
&= \sum_j a_{sj} Pr(\mathbf{x}_{t+1}|sj)\beta^*_{t+1}(j) \\
\beta^*_T(n-1) &= 1 \\
\beta^*_0(0) &= Pr(\mathbf{X}|\lambda) = \alpha^*_T(n-1)
\end{aligned}$$

The same scaling factors γ computed in the forward pass are used in the backward pass. That is, the β_t cells are scaled by γ_t after they are computed. Therefore, the initial condition for computing β is

$$\beta_T(s) = \frac{\beta^*_T(s)}{\gamma_T} \ \forall s$$

Next for frame T-1,

$$\begin{aligned}
\beta_{T-1}(s) \text{ before scaling} &= \sum_j a_{sj} Pr(\mathbf{x}_T|sj)\beta_T(j) \\
&= \sum_j a_{sj} Pr(\mathbf{x}_T|sj)\frac{\beta^*_T(j)}{\gamma_T} \\
&= \frac{\beta^*_{T-1}(s)}{\gamma_T} \\
\beta_{T-1}(s) &= \frac{\beta^*_{T-1}(s)}{\gamma_{T-1}\gamma_T}
\end{aligned}$$

And for frame T-2,

$$\begin{aligned}
\beta_{T-2}(s) \text{ before scaling} &= \sum_j a_{sj} Pr(\mathbf{x}_{T-1}|sj)\beta_{T-1}(j) \\
&= \sum_j a_{sj} Pr(\mathbf{x}_{T-1}|sj)\,\frac{\beta^*_{T-1}(s)}{\gamma_{T-1}\gamma_T} \\
&= \frac{\beta^*_{T-2}(s)}{\gamma_{T-1}\gamma_T} \\
\beta_{T-2}(s) &= \frac{\beta^*_{T-2}(s)}{\gamma_{T-2}\gamma_{T-1}\gamma_T}
\end{aligned}$$

Once again, by induction it can be shown that

$$\begin{aligned}\beta_t(s) &= \frac{\beta_t^*(s)}{\gamma_t\gamma_{t+1}\cdots\gamma_T} \\ Pr(\mathbf{X}|\lambda) &= \beta_0^*(0) = \beta_0(0)\prod_{t=1}^{T}\gamma_t\end{aligned} \tag{B.3}$$

The true β^* value is the scaled β times the product of all the scaling factors from the last time frame back through the current time frame.

After scaling, $\beta_0(0) = \alpha_T(n-1)$ just as $\beta_0^*(0) = \alpha_T^*(n-1)$. However, the true probability that the given model generates the observed data sequence is

$$\beta_0(0)\prod_{t=1}^{T}\gamma_t \quad \text{or} \quad \alpha_T(n-1)\prod_{t=1}^{T}\gamma_t$$

It is this value ($Pr(\mathbf{X}|\lambda)$) that is guaranteed to converge by the iterative Baum-Welch re-estimation. However, since the true value $\alpha_T^*(n-1)$ is almost always out of the dynamic range of the machine, practically its logarithm is computed instead:

$$\log Pr(\mathbf{X}|\lambda) = \{\log \alpha_T(n-1)\} + \sum_{t=1}^{T}\log\gamma_t$$

B.1.3 Re-estimation

Non-null Arcs

To compute the re-estimation for SCHMMs, we compute

$$\begin{aligned}\xi_t^*(i,j,k) &= Pr(\text{at time frame } t, \text{ transition } ij \text{ is taken, and } \mathbf{x}_t \text{ is quantized to codeword } k \mid \mathbf{X}, \lambda) \\ &= \frac{\alpha_{t-1}^*(i)a_{ij}f_k(\mathbf{x}_t)b_{ij}(k)\beta_t^*(j)}{Pr(\mathbf{X}|\lambda)}\end{aligned}$$

for non-null arcs. According to the Baum-Welch re-estimation formula [20], all the re-estimations related to the output distribution, the transition probability and the VQ density functions are summations of $\xi_t^*(i,j,k)$ across different dimensions. Therefore, if we can prove that $\xi_t(i,j,k) = \xi_t^*(i,j,k)$, we prove that scaling does not affect Baum-Welch re-estimation.

In the main memory at run time, we have the scaled α and β values instead of the true mathematical values α^* and β^*. To compute $\xi_t(i,j,k)$, we use (B.1), (B.2) and (B.3):

$$\xi_t(i,j,k) = \frac{\alpha_{t-1}(i)a_{ij}f_k(\mathbf{x}_t)b_{ij}(k)\beta_t(j)}{\alpha_T(n-1)}$$

$$
\begin{aligned}
&= \frac{\alpha^*_{t-1}(i)/(\gamma_1\gamma_2...\gamma_{t-1})\ a_{ij}f_k(\mathbf{x}_t)b_{ij}(k)\ \beta^*_t(j)/(\gamma_t\gamma_{t+1}...\gamma_T)}{\alpha^*_T(n-1)/(\gamma_1\gamma_2...\gamma_T)} \\
&= \xi^*_t(i,j,k)
\end{aligned}
$$

We see that using the scaled α and β, we are able to re-produce the same re-estimation as originally defined.

Null Arcs

For null transitions, it is easy to see that (B.1) and (B.3) are still valid. When a transition is null, no data is output and only the transition probability has to be re-estimated.

$$
\begin{aligned}
\xi^*_t(i,j) &= Pr(\text{at time frame } t,\ \text{transition } ij \text{ is taken} \mid \mathbf{X}, \lambda) \\
&= \frac{\alpha^*_t(i)a_{ij}\beta^*_t(j)}{Pr(\mathbf{X}|\lambda)} \\
&= \frac{\alpha_t(i)\gamma_1\gamma_2...\gamma_t\ a_{ij}\beta_t(j)\ \gamma_t\gamma_{t+1}...\gamma_T}{\alpha_T(n-1)\ \gamma_1\gamma_2...\gamma_T} \\
&= \frac{\gamma_t\ \alpha_t(i)a_{ij}\beta_t(j)}{\alpha_T(n-1)} = \gamma_t\ \xi_t(i,j)
\end{aligned}
$$

Thus for null arcs, we have to be careful that the constant γ_t has to be multipled with $\xi_t(i,j)$ to obtain the correct re-estimation value.

B.2 Scaling and Density Normalization Simultaneously

The next issue is to normalize the density values $f_k(\mathbf{x}_t)$ to be $f'_k(\mathbf{x}_t)$ so that the normalized densities are stable. The normalized density for time frame t is

$$
f'_k(\mathbf{x}_t) = \frac{f_k(\mathbf{x}_t)}{c_t}
$$

The normalization factor c_t at time frame t can be any constant as long as (1) it makes f'_k stable and (2) the same constant (c_t) applies to the same data frame ($\mathbf{x}_t$) *aligned with any Markov state*. Since f_k can range from very small to very large, the normalization constant should be carefully designed. One way to choose a stable c_t is to set

$$
c_t = \sum_k f_k(\mathbf{x}_t)
$$

to make $0 \leq f_k'(\mathbf{x}_t) \leq 1$. For SCHMMs with phone-class dependent VQ codebooks presented in Section 3.4, a stable normalization factor is the average of the summations of densities over all phone classes. Condition (2) is important to guarantee the same re-estimation value as the Baum-Welch re-estimation formula defines. This will be clear later when we derive the relationship.

To prove that normalization does not affect Baum-Welch re-estimation, we assume the scaling procedure described in Section B.1 is still adopted since scaling is crucial to prevent floating-point exceptions. It is easy to derive a similar proof without the scaling assumption. Since the density is normalized, the new scaling factor for each time frame will be different. We distinguish the new α, β, and γ by adding a superscript $'$.

B.2.1 The Forward Pass

The initial values for γ_0' and α_0' are

$$\begin{aligned}
\gamma_0' &= 1 \\
\alpha_0'(s) &= \frac{\alpha_0^*(s)}{\gamma_0'} \\
&= \alpha_0^*(s) = \alpha_0(s) \ \forall s
\end{aligned}$$

For frame 1,

$$\begin{aligned}
\alpha_1'(s) \text{ before scaling} &= \sum_j \alpha_0'(j) a_{js} \sum_k f_k'(\mathbf{x}_1) b_{js}(k) \\
&= \sum_j \alpha_0^*(j) a_{js} \sum_k \frac{f_k(\mathbf{x}_1)}{c_1} b_{js}(k) \\
&= \frac{\alpha_1^*(s)}{c_1} \\
\gamma_1' &= \sum_s \{\alpha_1'(s) \text{ before scaling}\} \\
&= \frac{\sum_s \alpha_1^*(s)}{c_1} = \frac{\gamma_1}{c_1} \\
\alpha_1'(s) &= \frac{\alpha_1^*(s)}{c_1 \gamma_1'} = \frac{\alpha_1^*(s)}{c_1 \frac{\gamma_1}{c_1}} \\
&= \frac{\alpha_1^*(s)}{\gamma_1} = \alpha_1(s)
\end{aligned}$$

For frame 2,

$$\begin{aligned}
\alpha_2'(s) \text{ before scaling} &= \sum_j \alpha_1'(j) a_{js} \sum_k f_k'(\mathbf{x}_2) b_{js}(k) \\
&= \sum_j \frac{\alpha_1^*(j)}{c_1\gamma_1'} a_{js} \sum_k \frac{f_k(\mathbf{x}_2)}{c_2} b_{js}(k) \\
&= \frac{\alpha_2^*(s)}{c_1 c_2 \gamma_1'} \\
\gamma_2' &= \sum_s \{\alpha_2'(s) \text{ before scaling}\} \\
&= \frac{\sum_s \alpha_2^*(s)}{c_1 c_2 \gamma_1'} = \frac{\sum_s \alpha_2^*(s)}{c_2 \gamma_1} \\
&= \frac{\gamma_2}{c_2} \\
\alpha_2'(s) &= \frac{\alpha_2^*(s)}{c_1 c_2 \gamma_1' \gamma_2'} \\
&= \frac{\alpha_2^*(s)}{\gamma_1 \gamma_2} = \alpha_2(s)
\end{aligned}$$

By mathematic induction, the following relationship is derived:

$$\gamma_t' = \frac{\gamma_t}{c_t} \quad \forall\, t = 0..T \text{ with } c_0 = 1 \qquad \text{(B.4)}$$

$$\begin{aligned}
\alpha_t'(s) &= \frac{\alpha_t^*(s)}{c_1 c_2 \ldots c_t \gamma_1' \gamma_2' \ldots \gamma_t'} \\
&= \frac{\alpha_t^*(s)}{\gamma_1 \gamma_2 \ldots \gamma_t} = \alpha_t(s) \qquad \text{(B.5)}
\end{aligned}$$

$$P(\mathbf{X}|\lambda) = \alpha_T^*(n-1) = \alpha_T'(n-1) \prod_{t=1}^{T} \{c_t \gamma_t'\} \qquad \text{(B.6)}$$

It is actually easy to understand that $\alpha_t(s) = \alpha_t'(s)$ since the ratio of these $\alpha_t^*(.)$ cells is fixed and scaling simply enforces them to sum to 1. The new scaling factor is the old scaling factor divided by the normalization factor at that time frame.

B.2.2 The Backward Pass

According to the scaling algorithm, the same scaling factors γ' computed in the forward pass are used in the backward pass. Therefore, by (B.4) initially we have

$$\beta_T'(s) = \frac{\beta_T^*(s)}{\gamma_T'} \quad \forall s$$

$$= \frac{c_T \beta_T^*(s)}{\gamma_T} = c_T \beta_T(s)$$

For frame T-1,

$$\begin{aligned}
\beta'_{T-1}(s) \text{ before scaling} &= \sum_j a_{sj} \{\sum_k f'_k(\mathbf{x}_T) b_{sj}(k)\} \beta'_T(j) \\
&= \sum_j a_{sj} \{\sum_k \frac{f_k(\mathbf{x}_T)}{c_T} b_{sj}(k)\} \frac{\beta_T^*(j)}{\gamma'_T} \\
&= \frac{\beta_{T-1}^*(s)}{c_T \gamma'_T} \\
\beta_{T-1}(s) &= \frac{\beta_{T-1}^*(s)}{c_T \gamma'_{T-1} \gamma'_T} \\
&= \frac{c_{T-1} \beta_{T-1}^*(s)}{\gamma_{T-1} \gamma_T} = c_{T-1} \beta_{T-1}(s)
\end{aligned}$$

For frame T-2,

$$\begin{aligned}
\beta'_{T-2}(s) \text{ before scaling} &= \sum_j a_{sj} \{\sum_k f'_k(\mathbf{x}_{T-1}) b_{sj}(k)\} \beta'_{T-1}(j) \\
&= \sum_j a_{sj} \{\sum_k \frac{f_k(\mathbf{x}_{T-1})}{c_{T-1}} b_{sj}(k)\} \frac{\beta_{T-1}^*(s)}{c_T \gamma'_{T-1} \gamma'_T} \\
&= \frac{\beta_{T-2}^*(s)}{c_{T-1} c_T \gamma'_{T-1} \gamma'_T} \\
\beta_{T-2}(s) &= \frac{\beta_{T-2}^*(s)}{c_{T-1} c_T \gamma'_{T-2} \gamma'_{T-1} \gamma'_T} \\
&= \frac{c_{T-2} \beta_{T-2}^*(s)}{\gamma_{T-2} \gamma_{T-1} \gamma_T} = c_{T-2} \beta_{T-2}(s)
\end{aligned}$$

By mathematic induction,

$$\begin{aligned}
\beta'_t(s) &= \frac{\beta_t^*(s)}{c_{t+1} c_{t+2} \ldots c_T \gamma'_t \gamma_{t+1}' \ldots \gamma'_T} \\
&= \frac{c_t \beta_t^*(s)}{\gamma_t \gamma_{t+1} \ldots \gamma_T} = c_t \beta_t(s) \qquad \text{(B.7)} \\
Pr(\mathbf{X}|\lambda) &= \beta_0^*(0) \\
&= \beta'_0(0) \prod_{t=1}^{T} \{c_t \gamma'_t\}
\end{aligned}$$

The new $\beta'_t(.)$ cell is the old one $\beta_t(.)$ times the normalization factor at time t.

Therefore, after simultaneous scaling and normalization, $\beta_0'(0) = \alpha_T'(n-1)$ just as $\beta_0^*(0) = \alpha_T^*(n-1)$. However, the true probability that the given model generates the observed data sequence is

$$\beta_0'(0) \prod_{t=1}^{T} \{c_t\gamma_t'\} \quad \text{or} \quad \alpha_T'(n-1) \prod_{t=1}^{T} \{c_t\gamma_t'\}$$

It is this value ($Pr(\mathbf{X}|\lambda)$) that is guaranteed to converge by the iterative Baum-Welch re-estimation.

B.2.3 Re-estimation

As in the previous case, in memory at run time, we have only α', β' and γ' available, instead of the true values.

Non-null Arcs

To compute $\xi_t'(i,j,k)$, we will use (B.5), (B.7) and (B.6):

$$\begin{aligned}
\xi_t'(i,j,k) &= \frac{\alpha_{t-1}'(i)a_{ij}f_k'(\mathbf{x}_t)b_{ij}(k)\beta_t'(j)}{\alpha_T'(n-1)} \\
&= \frac{\alpha_{t-1}^*(i)/(\gamma_1\gamma_2...\gamma_{t-1}) \;\; a_{ij}\frac{f_k(\mathbf{X}_t)}{c_t}b_{ij}(k) \;\; c_t\beta_t^*(j)/(\gamma_t\gamma_{t+1}...\gamma_T)}{\alpha_T^*(n-1)/(\gamma_1\gamma_2...\gamma_T)} \\
&= \xi_t(i,j,k) \;=\; \xi_t^*(i,j,k)
\end{aligned}$$

Null Arcs

For null transitions, it is easy to see that (B.5) and (B.7) are still valid. To update the transition count of a null arc, we compute

$$\begin{aligned}
\xi_t^*(i,j) &= Pr(\text{at time frame } t,\ \text{transition } ij \text{ is taken} \mid \mathbf{X}, \lambda) \\
&= \frac{\alpha_t^*(i)a_{ij}\beta_t^*(j)}{Pr(\mathbf{X}|\lambda)} \\
&= \frac{c_1c_2...c_t\gamma_1'\gamma_2'...\gamma_t'\alpha_t'(i)a_{ij}\beta_t'(j)c_{t+1}c_{t+2}...c_T\gamma_t'\gamma_{t+1}'...\gamma_T'}{c_1c_2...c_T\gamma_1'\gamma_2'...\gamma_T'\alpha_T'(n-1)} \\
&= \frac{\gamma_t'\,\alpha_t'(i)a_{ij}\beta_t'(j)}{\alpha_T'(n-1)} \;=\; \gamma_t'\,\xi_t'(i,j)
\end{aligned}$$

Note that for null arcs, the constant γ_t' has to be multipled with $\xi_t'(i,j)$ to obtain the correct re-estimation value.

B.2.4 Conclusion

The relationships among true values, the scaled values, and the *normalized and scaled* values can be summarized as

$$\begin{aligned}
\alpha_t(s) &= \alpha_t'(s) = \frac{\alpha_t^*(s)}{\gamma_1\gamma_2\cdots\gamma_t} \\
\gamma_t &= \frac{\sum_s \alpha_t^*(s)}{\gamma_1\gamma_2\cdots\gamma_{t-1}} \\
\gamma_t' &= \frac{\gamma_t}{c_t} \\
\beta_t(s) &= \frac{\beta_t^*(s)}{\gamma_t\gamma_{t+1}\cdots\gamma_T} \\
\beta_t'(s) &= c_t\beta_t(s)
\end{aligned}$$

With the above relationship, the re-estimation value $\xi_t^*(i,j,k)$ can be computed using either the scaled or *normalized and scaled* cells. That is,

$$\xi_t^*(i,j,k) = \xi_t(i,j,k) = \xi_t'(i,j,k)$$

The probability that the given model generates the training data is

$$\begin{aligned}
Pr(\mathbf{X}|\lambda) &= \alpha_T^*(n-1) \\
&= \alpha_T(n-1)\prod_{t=1}^{T}\gamma_t \\
&= \alpha_T'(n-1)\prod_{t=1}^{T}\{c_t\gamma_t'\}
\end{aligned}$$

B.3 Beam Search

During beam search, most systems use log-scale integer representation instead of the floating-point probabilities. All probability multiplications become additions in the log domain. When representing probability P with $\log_b P$, we set b to be close to one in order to get maximal precision [1]. Scaling is unnecessary since after taking the logarithm, underflow becomes unlikely.

[1]SPHINX uses b = 1.0001 [87].

It is easy to show that normalizing the VQ density value will not affect the rank of hypotheses during the time-synchronous beam search.

To see this, recall that in Section 4.2.4 we define the total score for a path up to the current frame to be

$$\text{total score} = \log(\text{acoustic probability}) + w\ \log(\text{language probability})$$

where w is the language weight used to combine the two knowledge sources. When the acoustic probability for every time frame t is normalized by c_t, the normalized total score becomes

$$\text{normalized total score} = \log(\text{acoustic probability}) + w\ \log(\text{language probability}) - \sum_{i=1}^{t} \log c_i$$

Since the same constant ($-\sum_{i=1}^{t} \log c_i$) is applied to all the states at the same time frame t and the search is time-synchronous, the relative rank of all hypotheses remains unchanged.

Appendix C

Entropy and Cross Entropy

Assume that a source generator emits the observed data with a probability distribution $\mathcal{P}$. Then the entropy of the source is defined as [1]

$$H(\mathcal{P}) \triangleq -\sum_{i=1}^{L} p_i \log_b p_i \quad \text{when the observed data are among a finite set of } L \text{ symbols, or}$$

$$H(\mathcal{P}) \triangleq -\int_{\mathbf{X}} p(\mathbf{x}) \log_b p(\mathbf{x}) d\mathbf{x} \quad \text{when the observed data are continuous vectors.}$$

When a logarithm to the base 2 is used, the resulting unit of entropy is called the *binary unit* or *bit*. Entropy measures the expected uncertainty of the source that generates the possible events. The more uncertainty is present, the bigger the entropy. Entropy is also the amount of information we can expect when we observe a piece of data emitted from the source. For example, when $\mathcal{P}$ is discrete and $p_k = 1,\ p_{j \neq k} = 0$, then there is no surprise that we will definitely observe the symbol k (i.e., $H(\mathcal{P}) = 0$). On the other hand, when $\mathcal{P}$ is uniform, $H(\mathcal{P})$ reaches its maximum, $\log_2 L$. This means that any symbol emitted gives us the maximal information since we are not sure in advance which symbol is coming out next. Entropy is also the minimum number of bits required per symbol to encode any sequence of the symbols into a uniquely decodable binary code [58].

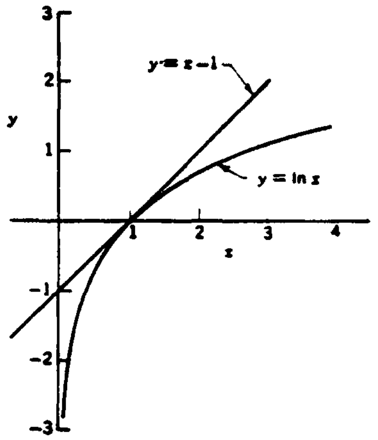
Cross entropy is known to be a good measure in determining the distance between two probability distributions [69, 81, 76, 75]. The cross entropy (directed divergence) [81] between two distributions, $\mathcal{P}$ and $\mathcal{Q}$, is defined as

$$\chi(\mathcal{P}, \mathcal{Q}) \triangleq \sum_i p_i \log_2 \frac{p_i}{q_i} \quad , \text{ or} \tag{C.1}$$

$$\chi(\mathcal{P}, \mathcal{Q}) \triangleq \int_{\mathbf{X}} p(\mathbf{x}) \log_2 \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} \tag{C.2}$$

Mathematically, p has to be zero whenever q is in order to make (C.1) and (C.2) converge. Practically, we can always use an arbitrarily small value to represent zero probability.

Cross entropy is always non-negative. The proof is as follows: Note that $\ln(x) \leq x - 1$ for non-negative x,



In other words, $\ln(1/x) \geq 1 - x$ and thus

$$\begin{aligned}
\chi(\mathcal{P}, \mathcal{Q}) &= \sum_i p_i \log_2 \frac{p_i}{q_i} \\
&= \frac{1}{\ln 2} \sum_i p_i \ln \frac{p_i}{q_i} \\
&\geq \frac{1}{\ln 2} \sum_i p_i (1 - \frac{q_i}{p_i}) \\
&= \frac{1}{\ln 2} (\sum_i p_i - \sum_i q_i) \\
&= 0 \qquad \text{(C.3)}
\end{aligned}$$

since $\sum_i p_i = \sum_i q_i = 1$. The same derivation applies in the continuous case, where the summation is replaced by an integral. Therefore, cross entropy is always non-negative. The greater the cross entropy is, the more divergent these two distributions are.

According to its definition, cross entropy is apparently not symmetric (and is thus *directed*). To make the distance measure symmetric, we can take the average of both directions:

$$\chi_s(\mathcal{P}, \mathcal{Q}) = \frac{\chi(\mathcal{P}, \mathcal{Q}) + \chi(\mathcal{Q}, \mathcal{P})}{2}$$

To understand the meaning of cross entropy, consider the probability of generating event i for p_i times independently, given discrete distribution $\mathcal{Q}$:

$$Pr(\ p_i \text{ occurrences of data } i \mid \text{distribution } \mathcal{Q}) = q_i{}^{p_i}$$

In other words,

$$\begin{aligned} Pr(\text{data } \mathcal{P} \mid \text{distribution } \mathcal{Q}) &= \prod_i q_i{}^{p_i} \\ \log Pr(\text{data } \mathcal{P} \mid \text{distribution } \mathcal{Q}) &= \sum_i p_i \log q_i \;\leq\; \sum_i p_i \log p_i \qquad \text{by (C.3)} \end{aligned}$$

with equality being true when distribution $\mathcal{Q}$ mimics data $\mathcal{P}$'s distribution, i.e. when $\mathcal{P} = \mathcal{Q}$. The cross entropy of $\mathcal{Q}$ with respect to data $\mathcal{P}$ thus measures how well distribution $\mathcal{Q}$ resembles distribution $\mathcal{P}$.

Appendix D

Linguistic Questions for Constructing the Decision Tree

beginning-word triphone	
ending-word triphone	
single-phone triphone	
silence	SIL
aspiration	HH
dental	DH TH
LW	L W
S/SH	S SH
S/Z/SH/ZH	S Z SH ZH
affricate	CH TS JH
nasal	M N NG
schwa	AX IX AXR
voiced-fric	DH Z ZH V
voiceless-fric	TH S SH F
fricative	DH TH S SH Z ZH V F
liquid	L R
lqgl-back	L R W
liquid-glide	L R W Y
w-glide	UW AW OW W
y-glide	IY AY EY OY Y
diphthong	UW AW AY EY IY OW OY
round-vocalic	UH AO UW OW OY W AXR ER

labial	W M B P V F
palatl	Y CH JH SH ZH
alvelr	N D T S Z DX TS
alveolar-stop	D T
velar	NG G K
velar-stop	G K
labial-stop	B P
delete-stop	PD TD KD BD DD GD
oral-stop1	B D G P T K CH TS JH BD DD GD PD TD KD
oral-stop2	P T K PD TD KD
oral-stop3	B D G BD DD GD
front-r	AE EH IH IX IY EY AH AX Y AW
back-r	UH AO UW OW AA ER AXR OY L R W AY
back-l	UH AO UW OW AA ER AXR L R W AW
front-l	AE EH IH IX IY EY AH AX Y OY AY
retro-flex	R ER AXR
retro-vowel	ER AXR
high-vowel	IH IX IY UH UW Y
lax-vowel	EH IH IX UH AH AX
low-vowel	AE AA AO AW AY OY
tense-vowel	IY EY AE UW OW AA AO AY OY AW
vowel	AE EH IH IX IY UH AH AX AA AO UW AW AY EY OW OY ER AXR
sonorant	AE EH IH IX IY EY AH AX OY AY UH AO UW OW AA ER AXR AW L R W Y
voiced	AE EH IH IX IY UH AH AX AA AO UW AW AY EY OW OY L R W Y ER AXR M N NG JH B D DH G V Z ZH DX

Appendix E

A Complete Senonic Decision Tree

This appendix shows a complete decision tree in one of our experimental runs to classify the first state of all AA-triphones. There were 99 AA-triphones used to construct this decision tree. Note the amount of entropy reduction is weighted by counts as (3.1) defines. The question associated with each node is a compound question of the basic ones listed in the previous appendix. As there are 48 splits in the tree, there are 49 leaves or senones. These leaves are underlined in the table.

entropy reduction (bits)	node id	left child	right child	question
4909.84882	0	1	2	Is the left-phone voiced?
2543.01673	1	3	4	Is the left-phone a back-l or a labial?
1178.99827	3	5	6	Is the left-phone a back-l?
1052.33468	2	7	8	Is the left-phone an oral-stop1?
959.88344	5	9	10	Is the left-phone a retro-flex?
660.57011	8	11	12	Is the left-phone S/SH or not voiceless-fric?
556.69071	4	13	14	Is the left-phone voiced-fric?
494.29429	7	15	16	Is the right-phone a liguid?
487.93191	6	17	18	Is this triphone a beginning-word triphone?
437.52164	11	19	20	Is the left-phone S/SH?
382.88286	10	21	22	Is the left-phone a w-glide?

entropy reduction	node	left	right	question
(bits)	id	child	child	
320.12406	14	23	24	Is the left-phone a front-l or a nasal?
318.95499	20	25	26	Is this triphone a beginning-word triphone or is the right-phone LW?
284.42239	15	27	28	Is this triphone a beginning-word triphone or is the left-phone not an oral-stop2?
205.68148	23	29	30	Is the left-phone a nasal?
193.61281	25	31	32	Is the left-phone aspiration or the right-phone not a liquid?
174.15782	9	33	34	Is the right-phone a liquid?
171.41922	21	35	36	Is this triphone a beginning-word triphone?
156.20514	27	37	38	Is this triphone a beginning-word triphone?
152.50721	22	39	40	Is the right-phone a lqgl-back?
139.39036	16	41	42	Is the right-phone a nasal?
129.00981	18	43	44	Is the right-phone a liguid?
111.16402	30	45	46	Is this triphone a beginning-word triphone?
107.06924	42	47	48	Is the right-phone a velar-stop?
102.41503	37	49	50	Is the left-phone a delete-stop?
90.45528	34	51	52	Is this triphone a beginning-word triphone or the right-phone a alvelr?
88.21653	45	53	54	Is the left-phone a y-glide?
86.80684	40	55	56	Is this triphone an ending-word triphone?
85.40782	46	57	58	Is the right-phone a nasal?
82.00750	48	59	60	Is the right-phone S/Z/SH/ZH?
73.31391	39	61	62	Is this triphone a beginning-word triphone?
70.06145	36	63	64	Is the right-phone a liquid?
66.88791	31	65	66	Is this triphone a beginning-word triphone?
66.87074	29	67	68	Is this triphone a beginning-word triphone?
63.73759	33	69	70	Is this triphone a beginning-word triphone?
63.50981	19	71	72	Is this triphone a beginning-word triphone?
60.64476	51	73	74	Is this triphone a beginning-word triphone or the right-phone a nasal?
57.39469	49	75	76	Is the left-phone an oral-stop2?

entropy reduction (bits)	node id	left child	right child	question
48.44302	13	77	78	Is this triphone a beginning-word triphone?
42.40140	44	79	80	Is the right-phone an affricate?
39.36533	64	81	82	Is the right-phone a nasal?
36.84698	56	83	84	Is the right-phone a voiceless-fric?
36.32877	24	85	86	Is the right-phone a liquid?
22.97764	73	87	88	Is this triphone a beginning-word triphone?
21.86084	66	89	90	Is this triphone an ending-word triphone?
19.80240	60	91	92	Is this triphone a beginning-word triphone?
17.12642	55	93	94	Is the right-phone silence?
13.61258	82	95	96	Is this triphone an ending-word triphone?

Bibliography

[1] Abramson, N. **Information Theory and Coding**. McGraw-Hill, New York, 1963, 11-16 pp.

[2] Acero, A. and Stern, R. *Robust Speech Recognition by Normalization of Acoustic Space.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1990, pp. 849–852.

[3] Alleva, F., Huang, X., and Hwang, M. *An Improved Search Algorithm for Continuous Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1993.

[4] Asadi, A., Schwatrz, R., and Makhoul, J. *Automatic Modeling For Adding New Words To A Large-Vocabulary Continuous Speech Recognition System.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1991, pp. 305–308.

[5] Aubert, X., Ney, H., and Haeb-Umbach, R. *Philips Research System for Continuous-Speech Recognition Overview and Evaluation on the DARPA RM Task.* in: **DARPA Continuous Speech Recognition Workshop**. DARPA Microelectronics Technology Office, Stanford, CA, 1992.

[6] Bahl, L. R., Bakis, R., Cohen, P. S., Cole, A. G., Jelinek, F., Lewis, B. L., and Mercer, R. L. *Further Results on the Recognition of a Continuously Read Natural Corpus.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1980.

[7] Bahl, L. R., Jelinek, F., and Mercer, R. *A Maximum Likelihood Approach to Continuous Speech Recognition.* **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. PAMI-5 (1983), pp. 179–190.

[8] Bahl, L., Brown, P., De Souza, P., and Mercer, R. *A New Algorithm for the Estimation of Hidden Markov Model Parameters.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1988.

[9] Bahl, L., Brown, P., de Souza, P., Mercer, R., and Picheny, M. *A Method for the Construction of Acoustic Markov Models for Words.* no. RC 13099 (#58580), IBM Thomas J. Watson Research Center, September 1987.

[10] Bahl, L., Brown, P., de Souza, P., Mercer, R., and Picheny, M. *A Method for the Construction of Acoustic Markov Models for Words.* **IEEE Transactions on Speech and Audio Processing**, vol. 1 (1993), pp. 443–452.

[11] Bahl, L., Brown, P., de Souze, P., and Mercer, R. *A Tree-Based Statistical Language Model for Natural Language Speech Recognition.* **IEEE Transactions on Acoustics, Speech, and Signal Processing**, vol. ASSP-37 (1989), pp. 1001–1008.

[12] Bahl, L., de Souza, P., Gopalakrishnan, P., Nahamoo, D., and Picheny, M. *Decision Trees for Phonological Rules in Continuous Speech.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1991, pp. 185–188.

[13] Bahl, L. and et. al. *Automatic Phonetic Baseform Determination.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1991, pp. 173–176.

[14] Bahl, L., Souza, P., Gopalakrishnan, P., and Picheny, M. *Context Dependent Vector Quantization For Continuous Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1993, pp. 632–635.

[15] Baker, J. and etc.. *Large Vocabulary Recognition of Wall Street Journal Sentences at Dragon Systems.* in: **DARPA Speech and Language Workshop**. 1992.

[16] Baker, J. K. *The DRAGON System – An Overview.* **IEEE Transactions on Acoustics, Speech, and Signal Processing**, vol. ASSP-23 (1975), pp. 24–29.

[17] Baker, J. K. *Stochastic Modeling as a Means of Automatic Speech Recognition.* Computer Science Department, Carnegie Mellon University, April 1975.

[18] Bakis, R. *Continuous Speech Recognition via Centisecond Acoustic States.* in: **91st Meeting of the Acoustical Society of America**. 1976.

[19] Bamberg, P. and Gillick, L. *Phoneme-in-Context Modeling for Dragon's Continuous Speech Recognizer.* in: **DARPA Speech and Language Workshop**. 1990.

[20] Baum, L. E. *An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes.* **Inequalities**, vol. 3 (1972), pp. 1–8.

[21] Bellegarda, J. and Nahamoo, D. *Tied Mixture Continuous Parameter Modeling for Speech Recognition,.* **IEEE Transactions on Acoustics, Speech, and Signal Processing**, vol. ASSP-38 (1990), pp. 2033–2045.

[22] Bellegarda, J. and Nahamoo, D. *Tied Mixture Continuous Parameter Models for Large Vocabulary Isolated Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1989, pp. 13–16.

[23] Bernstein, J., Cohen, M., Murveit, H., and Weintraub, M. *Linguistic Constraints in Hidden Markov Model Based Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1989.

[24] Breiman, L., Friedman, J., Olshen, R., and Stone, C. **Classification and Regression Trees**. Wadsworth, Inc., Belmont, CA., 1984.

[25] Brown, P. *The Acoustic-Modeling Problem in Automatic Speech Recognition*. Computer Science Department, Carnegie Mellon University, May 1987.

[26] Chase, L. *Personal Communication*. unpublished, 1993.

[27] Chen, F. and Shrager, J. *Automatic Discovery of Contextual Factors Describing Phonological variation*. in: **DARPA Speech and Language Workshop**. 1989.

[28] Chow, Y. L., Schwartz, R., Roucos, S., Kimball, O., Price, P., Kubala, F., Dunham, M., Krasner, M., and Makhoul, J. *The Role of Word-Dependent Coarticulatory Effects in a Phoneme-Based Speech Recognition System*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1986.

[29] Chow, Y. and et. al. *BYBLOS: The BBN Continuous Speech Recognition System*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1987, pp. 89–92.

[30] Cohen, M., , Murveit, H., Bernstein, J., P., P., and Weintraub, M. *The DECIPHER Speech Recognition System*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1990.

[31] Davis, S. and Mermelstein, P. *Comparison of Parametric Representations of Monosyllabic Word Recognition in Continuously Spoken Sentences,*. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, vol. ASSP-28 (1980), pp. 357–366.

[32] Digalakis, V. and Murveit, H. *An Algorithm for Optimizing the Degree of Mixture-Tying in a Large-Vocabulary HMM-Based Speech Recognizer*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1994.

[33] Doddington, G. *Phonetically Sensitive Discriminants for Improved Speech Recognition*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1989, pp. 556–559.

[34] D'Orta, P., Ferretti, M., and Scarci, S. *Phoneme Classification for Real Time Speech Recognition of Italian*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1987, pp. 81–84.

[35] Feng, M., Kubala, F., and Schwartz, R. *Improved Speaker Adaptation Using Text Dependent Mappings*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1988, pp. 131–134.

[36] Fisher, W., Doddington, G., and Goudie-Marshall, K. *The DARPA Speech Recognition Research Database: Specifications and Status*. in: **DARPA Speech Recognition Workshop**. 1986.

[37] Fisher, W., Zue, V., Bernstein, J., and Pallett, D. *An Acoustic-Phonetic Data Base*. in: **113th Meet[illegible] Acoustical Society of America**. 1987.

[38] Franzini, M., Lee, K., and Waibel, A. *Connectionist Viterbi Training for Continuous Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1990.

[39] Gray, R. *Vector Quantization.* **IEEE ASSP Magazine**, vol. 1 (1984), pp. 4–29.

[40] Hild, H. and Waibel, A. *Multi-Speaker/Speaker-Independent Architectures For the Multi-State Time Delay Neural Network.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. vol. II, 1993, pp. 255–258.

[41] Holmes, W., Wood, L., and Pearce, D. *Allophone Modeling For Vocabulary-Independent HMM Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. vol. II, 1993, pp. 487–490.

[42] Hon, H. *Vocabulary-Independent Speech Recognition: : The VOCIND System.* School of Computer Science, Carnegie Mellon University, February 1992.

[43] Hon, H. and Lee, K. *CMU Robust Vocabulary-Independent Speech Recognition System.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. Toronto, Ontario, CANADA, 1991, pp. 889–892.

[44] Huang, X. *Phoneme Classification Using Semicontinuous Hidden Markov Models.* **IEEE Transactions on Signal Processing**, vol. 40 (1992), pp. 1062–1067.

[45] Huang, X. *A Study on Speaker-Adaptive Speech Recognition.* in: **DARPA Speech and Language Workshop**. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[46] Huang, X., Alleva, F., Hon, H., Hwang, M., Lee, K., and Rosenfeld, R. *The SPHINX-II Speech Recognition System: An Overview.* **Computer Speech and Language**, vol. 2 (1993), pp. 137–148.

[47] Huang, X., Alleva, F., Hwang, M., and Rosenfeld, R. *An Overview of the SPHINX-II Speech Recognition System.* in: **ARPA Human Language Technology Workshop**. 1993.

[48] Huang, X., Alleva, F., Hayamizu, S., Hon, H., Hwang, M., and Lee, K. *Improved Hidden Markov Modeling for Speaker-Independent Continuous Speech Recognition.* in: **DARPA Speech and Language Workshop**. Morgan Kaufmann Publishers, Hidden Valley, PA, 1990, pp. 327–331.

[49] Huang, X., Ariki, Y., and Jack, M. **Hidden Markov Models for Speech Recognition**. Edinburgh University Press, Edinburgh, U.K., 1990.

[50] Huang, X., Belin, M., Alleva, F., and Hwang, M. *Unified Stochastic Engine (USE) for Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1993.

[51] Huang, X., Hon, H., Hwang, M., and Lee, K. *A Comparative Study of Discrete, Semicontinuous, and Continuous Hidden Markov Models.* **Computer Speech and Language, in press**, 1993.

[52] Huang, X., Hon, H., and Lee, K. *Multiple Codebook Semi-Continuous Hidden Markov Models for Speaker-Independent Continuous Speech Recognition*. Technical Report, no. CMU-CS-89-136, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, May 1989.

[53] Huang, X. and Jack, M. *Semi-Continuous Hidden Markov Models with Maximum Likelihood Vector Quantization*. in: **IEEE Workshop on Speech Recognition**. 1988.

[54] Huang, X., Lee, K., and Waibel, A. *Connectionist speaker normalization and its applications to speech recognition*. in: **IEEE Workshop on Neural Networks for Signal Processing**. 1991.

[55] Huang, X. and Lee, K. *On Speaker-Independent, Speaker-Dependent, and Speaker-Adaptive Speech Recognition*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1991, pp. 877–880.

[56] Huang, X., Lee, K., and Hon, H. *On Semi-Continuous Hidden Markov Modeling*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. Albuquerque, NM, 1990, pp. 689–692.

[57] Huang, X., Lee, K., Hon, H., and Hwang, M. *Improved Acoustic Modeling for the SPHINX Speech Recognition System*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. Toronto, Ontario, CANADA, 1991, pp. 345–348.

[58] Huffman, D. *A Method for the Construction of Minimum Redundancy Codes*. **Proc. IRE**, vol. 40 (1952), pp. 1098–1101.

[59] Hunt, M. J., Lennig, M., and Mermelstein, P. *Experiments in Syllable-Based Recognition of Continuous Speech*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1980, pp. 880–883.

[60] Hwang, M. *Personal Experience*. unpublished, 1990.

[61] Hwang, M., Hon, H., and Lee, K. *Between-Word Coarticulation Modeling for Continuous Speech Recognition*. Technical Report, no. CMU-CS-89-141R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, April 1989.

[62] Hwang, M., Hon, H., and Lee, K. *Modeling Between-Word Coarticulation in Continuous Speech Recognition*. in: **Proceedings of Eurospeech**. Paris, FRANCE, 1989, pp. 5–8.

[63] Hwang, M., Hon, H., and Lee, K. *Modeling Inter-Word Coarticulation Using Generalized Triphones*. in: **The 117th Meeting of the Acoustical Society of America**. Syracuse, NY, 1989.

[64] Hwang, M. and Huang, X. *Shared-Distribution Hidden Markov Models for Speech Recognition*. **IEEE Transactions on Speech and Audio Processing**, vol. 1 (1993), pp. 414–420.

[65] Hwang, M. and Huang, X. *Subphonetic Modeling for Speech Recognition*. in: **DARPA Speech and Language Workshop**. Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[66] Hwang, M., Huang, X., and F., A. *Senones, Multi-Pass Search, and Unified Stochastic Modeling in SPHINX-II*. in: **Proceedings of Eurospeech**. 1993.

[67] Hwang, M. and Huang, X. *Subphonetic Modeling with Markov States — Senone*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1992.

[68] Hwang, M., Huang, X., and Alleva, F. *Predicting Unseen Triphones with Senones*. in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1993.

[69] Jeffreys, H. **Theory of Probability**. Oxford University Press, 1948.

[70] Jelinek, F. *Continuous Speech Recognition by Statistical Methods*. **Proceedings of the IEEE**, vol. 64 (1976), pp. 532–556.

[71] Jelinek, F. *Self-Organized Language Modeling for Speech Recognition*. unpublished, 1987.

[72] Jelinek, F. and Mercer, R. *Interpolated Estimation of Markov Source Parameters from Sparse Data*. in: **Pattern Recognition in Practice**, edited by E. Gelsema and L. Kanal. North-Holland Publishing Company, Amsterdam, the Netherlands, 1980, pp. 381–397.

[73] Juang, B. H. and Rabiner, L. R. *Mixture Autoregressive Hidden Markov Models for Speech Signals*. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, vol. ASSP-33 (1985), pp. 1404–13.

[74] Juang, B. *Speech Recognition in Adverse Environments*. vol. 5, 1991, pp. 275–294.

[75] Juang, B. and Rabiner, L. *A Probabilistic Distance Measure for Hidden Markov Models*. **The Bell System Technical Journal**, vol. 64 (1985), pp. 391–408.

[76] Kailath, T. *The Divergence and Bhattacharyya Distance Measures in Signal Selection*. **IEEE Transactions on Communication Technology**, vol. COM-15 (1967), pp. 52–60.

[77] Kimball, O. and Ostendorf, M. *On the Use of Tied-Mixture Distributions*. in: **ARPA Human Language Technology Workshop**. 1993.

[78] Kimball, O., Price, P., Roucos, S., Schwartz, R., Kubala, F., Chow, Y.-L., Haas, A., Krasner, M., and Makhoul, J. *Recognition Performance and Grammatical Constraints*. in: **DARPA Speech and Language Workshop**, edited by L. S. Baumann. 1986, pp. 53–59.

[79] Klatt, D. *Problem of Variability in Speech Recognition and in Models of Speech Perception*. in: **Variability and Invariance in Speech Processes**, edited by J. Perkell and D. Klatt. Lawrence Erlbaum Assoc, Hillsdale, N.J., 1986, pp. 300–320.

[80] Kubala, F. and Schwartz, R. *A New Paradigm for Speaker-Independent Training and Speaker Adaptation.* in: **DARPA Speech and Language Workshop**. Morgan Kaufmann Publishers, San Mateo, CA, 1990.

[81] Kullback, S. **Information Theory and Statistics**. Dover, New York, 1959.

[82] Lamel, L. and Gauvain, J. *Cross-Lingual Experiments with Phone Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1993.

[83] Lee, C., Giachin, E., Rabiner, R., L. P., and Rosenberg, A. *Improved Acoustic Modeling for Continuous Speech Recognition.* in: **DARPA Speech and Language Workshop**. Morgan Kaufmann Publishers, San Mateo, CA, 1990.

[84] Lee, C., Rabiner, L., Pieraccini, R., and Wilpon, J. *Acoustic Modeling for Large Vocabulary Speech Recognition.* **Computer Speech and Language**, vol. 4 (1990), pp. 127–165.

[85] Lee, K. **Automatic Speech Recognition: The Development of the SPHINX System**. Kluwer Academic Publishers, Boston, 1989.

[86] Lee, K. *Context-Dependent Phonetic Hidden Markov Models for Continuous Speech Recognition.* **IEEE Transactions on Acoustics, Speech, and Signal Processing**, April 1990, pp. 599–609.

[87] Lee, K. *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System.* Computer Science Department, Carnegie Mellon University, April 1988.

[88] Lee, K., Hayamizu, S., Hon, H., Huang, C., Swartz, J., and Weide, R. *Allophone Clustering for Continuous Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. Albuquerque, NM, 1990, pp. 749–752.

[89] Lee, K. and Hon, H. *Large-Vocabulary Speaker-Independent Continuous Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. New York, NY, 1988.

[90] Lee, K., Hon, H., and Reddy, R. *An Overview of the SPHINX Speech Recognition System.* **IEEE Transactions on Acoustics, Speech, and Signal Processing**, January 1990, pp. 35–45.

[91] Lee, K. and Mahajan, S. *Corrective and Reinforcement Learning for Speaker-Independent Continuous Speech Recognition.* Technical Report, no. CMU-CS-89-100, Carnegie Mellon University, January 1989.

[92] Levinson, S. E., Rabiner, L. R., and Sondhi, M. M. *An Introduction to the Application of the Theory of Probabilistic Functions on a Markov Process to Automatic Speech Recognition.* **The Bell System Technical Journal**, vol. 62 (1983).

[93] Liberman, M. *Text on Tap: the ACL/DCI.* in: **DARPA Speech and Natural Language Workshop**. 1989, pp. 173–188.

[94] Linde, Y., Buzo, A., and Gray, R. *An Algorithm for Vector Quantizer Design.* **IEEE Transactions on Communication**, vol. COM-28 (1980), pp. 84–95.

[95] Lowerre, B. and Reddy, D. *The Harpy Speech Understanding System.* in: **The Harpy Speech Understanding System**, by B. Lowerre and D. Reddy, edited by W. Lee. Prentice-Hall, Englewood Cliffs, NJ, 1980.

[96] Lucassen, J. and Mercer, R. *An Information Theoretic Approach to the Automatic Determination of Phonemic Baseforms.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1984.

[97] Makhoul, J., Roucos, S., and Gish, H. *Vector Quantization in Speech Coding.* **Proceedings of the IEEE**, vol. 73 (1985), pp. 1551–1588.

[98] Markel, J. D. and Gray, A. H. **Linear Prediction of Speech**. Springer-Verlag, Berlin, 1976.

[99] Mercer, R. L. *Language Modeling for Speech Recognition.* in: **IEEE Workshop on Speech Recognition**. 1988.

[100] Nadas, A., Nahamoo, D., and Picheny, M. A. *Adaptive Labeling: Normalization of Speech by Adaptive Transformations based on Vector Quantization.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1988.

[101] Nagai, A., S., S., and Kita, K. *Phoneme-context-dependent LR parsing algorithms for HMM-based continuous speech recognition.* in: **Proceedings of Eurospeech**. 1991, pp. 1397–1400.

[102] Ney, H. and Noll, A. *Phoneme Modelling Using Continuous Mixture Densities.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1988, pp. 437–440.

[103] NIST. *Nov92 WSJ Evaluation.* in: **DARPA Spoken Language System Technology Workshop**. 1993.

[104] Noll, H., A. N. *Training of Phoneme Models in a Sentence Recognition System.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1987, pp. 1277–80.

[105] Office, A. M. T. *Continuous Speech Recognition Workshop.* 1991.

[106] Pallet, D. *WSJ Pilot Corpus.* National Institute of Standards and Technology, Limited distribution CD-ROM, 1991.

[107] Paul, D. B., Lippmann, R. P., Chen, Y., and Weinstein, C. *Robust HMM-Based Techniques for Recognition of Speech Produced under Stress and in Noise.* in: **Proceedings of the Speech Technology Conference**. 1986.

[108] Paul, D. *The Lincoln Robust Continuous Speech Recognizer.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1989, pp. 449 – 452.

[109] Paul, D. and Baker, J. *The Design for the Wall Street Journal-based CSR Corpus.* in: **DARPA Speech and Language Workshop**. Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[110] Paul, D. and Martin, E. *Speaker Stress-Resistant Continuous Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1988.

[111] Placeway, P. and Schwartz, R. *Personal Communication.* unpublished, 1993.

[112] Price, P., Fisher, W., Bernstein, J., and Pallett, D. *A Database for Continuous Speech Recognition in a 1000-Word Domain.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1988, pp. 651–654.

[113] Rabiner, L. R. *A Tutorial on Hidden Markov Models and Selected Applications Speech Recognition.* **IEEE Proceedings**, 1988.

[114] Rabiner, L. R., Juang, B. H., Levinson, S. E., and Sondhi, M. M. *Recognition of Isolated Digits Using Hidden Markov Models With Continuous Mixture Densities.* **AT&T Technical Journal**, vol. 64 (1985), pp. 1211–33.

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

[116] Rabiner, L. and Juang, B. *An Introduction to Hidden Markov Models.* **IEEE ASSP Magazine**, vol. 3 (1986), pp. 4–16.

[117] Reddy, D. and Zue, V. *Recognizing Continuous Speech Remains an Illusive Goal.* **IEEE Spectrum**, November 1983, pp. 84–87.

[118] Richter, A. *Modeling of Continuous Speech Observations.* in: **Advances in Speech Processing Conference, IBM Europe Institute**. 1986.

[119] Rosenberg, A. E., Rabiner, L. R., Wilpon, J., and Kahn, D. *Demisyllable-Based Isolated Word Recognition System.* **IEEE Transactions on Acoustics, Speech, and Signal Processing**, vol. ASSP-31 (1983), pp. 713–726.

[120] Sagayama, S. *Phoneme Environment Clustering for Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1989.

[121] Schwartz, e., R. *Comparative Experiments on Large Vocabulary Speech Recognition.* in: **ARPA Human Language Technology Workshop**. 1993.

[122] Schwartz, R., Austin, S., Kubala, F., and Makhoul, J. *New Uses for the N-Best Sentence Hypotheses Within the Byblos Speech Recognition System.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1992, pp. 1–4.

[123] Schwartz, R., Chow, Y., Kimball, O., Roucos, S., Krasner, M., and Makhoul, J. *Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1985, pp. 1205–1208.

[124] Schwartz, R., Klovstad, J., Makhoul, J., and Sorensen, J. *A Preliminary Design of a Phonetic Vocoder Based on a Diphone Model.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1980, pp. 32–35.

[125] Schwartz, R. M., Chow, Y. L., Roucos, S., Krasner, M., and Makhoul, J. *Improved Hidden Markov Modeling of Phonemes for Continuous Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1984.

[126] Soong, F., Rosenberg, A., Rabiner, L., and Juang, B. *A Vector Quantization Approach to Speaker Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1985, pp. 387–390.

[127] Soong, F. and Huang, E. *A Tree-Trellis Based Fast Search for Finding the N-Best Sentence Hypothesis.* in: **DARPA Speech and Language Workshop**. 1990.

[128] Takami, J. and Sagayama, S. *A Successive State Splitting Algorithm For Efficient Allophone Modeling.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1992, pp. I573–I576.

[129] Tebelskis, J. *Performance Through Consistency: Connectionist Large Vocabulary Continuous Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. vol. II, 1993, pp. 259–262.

[130] Tomita, M. **Efficient Parsing for Natural Language**. Kluwer Academic Publishers, Boston, 1986.

[131] Vintsyuk, T. *Element-wise recognition of continuous speech composed of words from a specified dictionary.* **Kibernetika**, vol. 7 (1971), pp. 133–143.

[132] Viterbi, A. J. *Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm.* **IEEE Transactions on Information Theory**, vol. IT-13 (1967), pp. 260–269.

[133] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. *Phoneme Recognition Using Time-Delay Neural Networks.* no. TR-1-0006, ATR International, Osaka, Japan, October 1987.

[134] Weide, R. *Personal Communication.* unpublished, 1989.

[135] Weintraub, M., Murveit, H., Cohen, M., Price, P., Bernstein, J., Baldwin, G., and Bell, D. *Linguistic Constraints in Hidden Markov Model Based Speech Recognition.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1989.

[136] Wood, L., Pearce, D., and Novello, F. *Improved Vocabulary-Independent Sub-Word HMM Modeling.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1991, pp. 181–184.

[137] Zue, V., Glass, M., Goodine, Leung, McCandless, Philips, M., Polifroni, and Seneff, S. *Recent Progress on the Voyager System.* in: **DARPA Speech and Language Workshop**. 1990.