Context-Dependent Modeling in a Segment-Based Speech Recognition System

by

Benjamin M. Serridge

B.S., MIT, 1995

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1997

Author . . . .
Department of Electrical Engineering
and Computer Science
August 22, 1997

Certified by . . . .
Dr. James R. Glass
Principal Research Scientist
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Departmental Committee on Graduate Theses

Context-Dependent Modeling in a Segment-Based Speech Recognition System

by

Benjamin M. Serridge

Submitted to the Department of Electrical Engineering and Computer Science on August 22, 1997 in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

The goal of this thesis is to explore various strategies for incorporating contextual information into a segment-based speech recognition system, while maintaining computational costs at a level acceptable for implementation in a real-time system. The latter is achieved by using context-independent models in the search, while context-dependent models are reserved for *re-scoring* the hypotheses proposed by the context-independent system.

Within this framework, several types of context-dependent sub-word units were evaluated, including word-dependent, biphone, and triphone units. In each case, deleted interpolation was used to compensate for the lack of training data for the models. Other types of context-dependent modeling, such as context-dependent boundary modeling and "offset" modeling, were also used successfully in the re-scoring pass.

The evaluation of the system was performed using the Resource Management task. Context-dependent segment models were able to reduce the error rate of the context-independent system by more than twenty percent, and context-dependent boundary models were able to reduce the word error rate by more than a third. A straight-forward combination of context-dependent segment models and boundary models leads to further reductions in error rate.

So that it can be incorporated easily into existing and future systems, the code for re-sorting N-best lists has been implemented as an object in Sapphire, a framework for specifying the configuration of a speech recognition system using a scripting language. It is currently being tested on Jupiter, a real-time telephone based weather information system under development here at SLS.

Acknowledgments

My experiences in the Spoken Language Systems group have been among the most valuable of my MIT education, and the respect I feel for my friends and mentors here has only grown with time. Perhaps more important than the help people have given me in response to particular problems is the example they set for me by the way they guide their lives. I would especially like to mention my advisor Jim, who has demonstrated unwavering support for me throughout the past year, and my office mates Sri and Giovanni, who have made room NE43-604 something more than an office over the past year. It is because of the people here that, mixed with my excitement about the future, I feel a tinge of sorrow at leaving this place.

Contents

List of Figures

List of Tables

Chapter 1

Context-Dependent Modeling

1.1 Introduction

Modern speech recognition systems typically classify speech into sub-word units that loosely correspond to phonemes. These phonetic units are, at least in theory, independent of task and vocabulary, and because they constitute a small set, each one can be well-trained with a reasonable amount of data. In practice, however, the acoustic realization of a phoneme varies greatly depending on its context, and speech recognition systems can benefit by choosing units that more explicitly model such contextual effects.

The goal of this thesis is to comparatively evaluate some strategies for modeling the effects of phonetic context, using a segment-based speech recognition system as a basis. The next section provides, by way of an overview of previous research on the topic, an introduction to several of the issues involved, followed by an outline of the remaining chapters of this thesis and a more precise statement of its objectives.

1.2 Previous Research

Kai-Fu Lee, in his description of the SPHINX system [17], presents a clear summary of the search for a good unit of speech, including a discussion of most of the units considered in this thesis. He frames the choice of speech unit in terms of a tradeoff between trainability and specificity: more specific acoustic models will, all else being equal, perform better than more general models, but because of their specificity they are likely to occur very rarely and are therefore difficult to train well. Very general models, on the other hand, can be well-trained, but are less likely to provide a good match to any particular token.

Since the goal of speech recognition is to recognize the words a person speaks, the most obvious choice of speech unit is the word itself. In fact, word models have been applied fairly successfully in small-vocabulary systems to problems such as the connected-digit recognition task [24]. Unfortunately, word models do not generalize well to larger vocabulary tasks, since the data used to train one word can not be shared by others. A more linguistically appealing unit of speech is the phoneme,

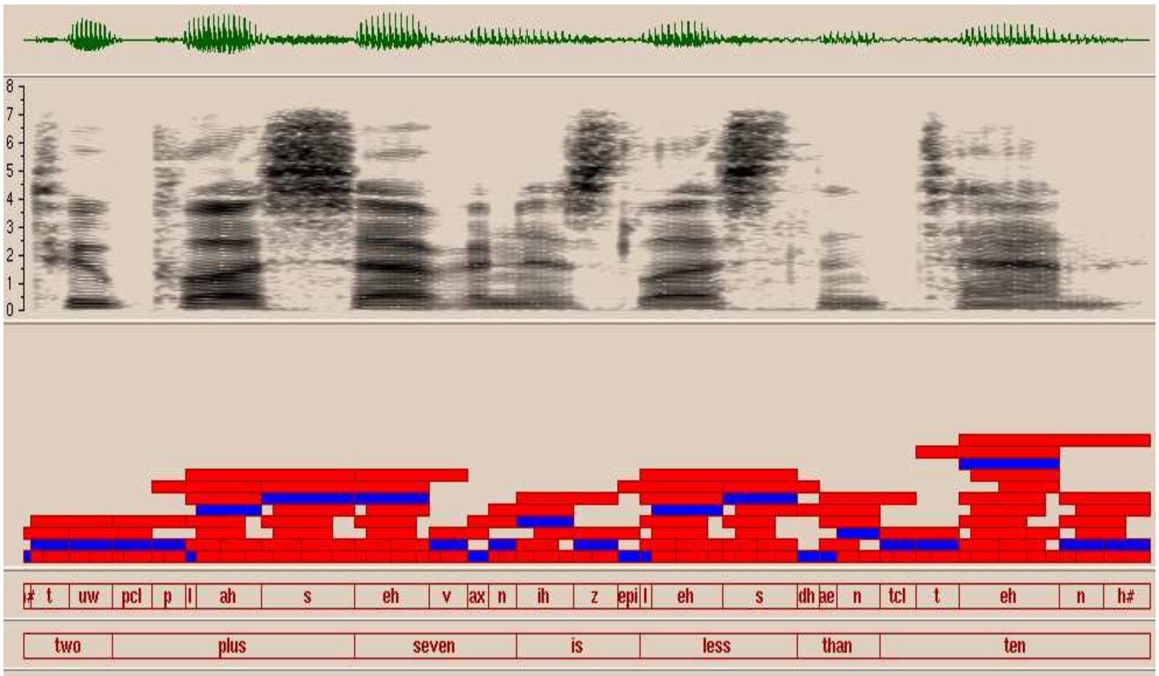

Figure 1-1: A spectrogram of the utterance “Two plus seven is less than ten.” Notice the variation in the realizations of the three examples of the phoneme /eh/: the first, in the word “seven,” exhibits formants (shown in the spectrogram as dark horizontal bands) that drop near the end of the phoneme as a result of the labial fricative /v/ that follows it; the second /eh/, in the word “less,” has a second formant that is being “pulled down” by the /l/ on the left; and the third /eh/, in the word “ten,” has first and third formants that are hardly visible due to energy lost in nasal cavities that have opened up in anticipation of the final /n/. If such variations can be predicted from context (as is believed to be the case), then speech recognition systems that do so will embody a much more precise model of what is actually occuring during natural speech than those that do not.

since a small set of units covers all possible utterances. This generality allows data to be shared across words, but at the same time forces each acoustic model to account for all the different possible realizations of a phoneme. Acoustic models can handle the variability within a phoneme implicitly if they are constructed as mixtures of several simpler component models. Previous research, however, has shown that superior performance can be obtained by handling the variation explicitly. Gender or speaker-dependent models, for example, create a separate model for each gender or speaker. Similarly, context-dependent models create a separate model for each context.

Many types of context-dependent models have been proposed in the literature. "Word-dependent" phone models, first proposed by Chow *et al* in 1986 [3], consider the context of a phone to be the word in which it occurs. Kai-Fu Lee applied such models in the SPHINX system to a small set of 42 "function words", such as *of*, *the*, and *with*, which accounted for almost 50% of the errors in the SPHINX system on the Resource Management task [17]. Adding these models to the context-independent system reduced the error rate by more than 25%, significantly decreasing the number of errors in both function words and non-function words.

More commonly used are phone models that are conditioned on the identity of the neighboring phones. A *left biphone* is dependent on the preceding phone, while a *right biphone* is dependent on the following phone. A *triphone* model depends on both the left and the right context. Such models were first proposed by Bahl *et al* in 1980 [1], and since then have been shown many times to improve the performance of various systems [26, 18]. The concept has even been extended to the use of quinphones which take into account the identity of the two following and preceding phones [29].

The aforementioned models all adhere to the same basic paradigm: the data that normally would contribute to the construction of just one model are grouped according to context, thus creating a separate model for each context. Unfortunately, if the number of possible contexts is large, the amount of data available to each model will be small.

This problem, known as the sparse data problem, can be dealt with in several ways. The simplest technique is to train models only for those units for which sufficient training data are available [16]. A more sophisticated (but not necessarily better) approach is to merge together contexts that have similar effects, thereby not only increasing the amount of training data per model, but also reducing the number of models that must be applied during recognition. The choice of models to be combined can be made either *a priori* (*e.g.*, using the linguistic knowledge of an expert [19]) or automatically (*e.g.*, using decision trees to split the data according to context [14], unsupervised clustering algorithms to merge the models themselves [17], or other methods).

Even after taking the above precautions, context-dependent models may still perform poorly on new data, especially if they have been trained from only a few examples. A technique known as "deleted interpolation" aleviates this problem by creating models as a linear combination of context-dependent and context-independent models. The extent to which each component contributes to the final model is calculated from the performance of each model on data that was "deleted" from the training set. This strategy was first applied to hidden Markov models by Jelinek and Mercer in 1980 [13] and has been described more recently by Huang *et al* [11].

Yet another issue raised by the use of context-dependent models is computational complexity, which can grow significantly if, during the search, the recognizer must postulate and test all possible contexts for a given region of speech. The "N-best search paradigm" [2] addresses this issue by using the standard recognizer to produce a list of the top N hypotheses, which are then re-evaluated and re-ranked using more sophisticated modeling techniques.

Most previous research has been performed on systems based on the use of hidden Markov models (HMM's) to perform recognition. The work presented in this thesis is based on SUMMIT [7], a *segment-based* continuous speech recognition system developed by the Spoken Language Systems group at MIT. Currently, the system used for real-time demonstrations and ongoing research is context-independent, although in the past context-dependent models have been used for evaluation purposes [22, 8]. The Sapphire framework [9] allows speech recognition systems to be constructed as a set of dependencies between individually configured components, and is used as a development platform for the systems described in this thesis. Evaluations are performed on the Resource Management task [23], which has been used extensively to evaluate the performance of several fledgling context-dependent systems [18, 15].

1.3 Thesis Objectives

The goal of this thesis is to evaluate different strategies for modeling contextual effects in a segment-based speech recognition system. Included in the evaluation are traditional methods such as word-dependent, biphone, and triphone modeling, as well as some more unusual approaches such as boundary modeling and context normalization techniques (offset models). In all cases, the basic approach is to use context-independent acoustic models to generate a list of hypotheses, which are then re-evaluated and re-ranked using context-dependent models.

The next chapter describes the components of the SUMMIT system relevant to this thesis, including an explanation of the Viterbi search, the A* search, and the algorithm used to re-score the hypotheses of the N-best list. Also included is a description of how the re-scoring algorithm is incorporated into the Sapphire framework.

Chapter 3 describes the context-independent baseline system and the Resource Management task. Some preliminary experimental results are presented for the baseline system, as well as some analysis which suggests that the system has the potential to achieve much higher performance, if it can somehow correctly select the best alternative from those in the N-best list.

Chapter 4 introduces the technique of deleted interpolation, including a description of how it is applied to the models used in this thesis.

Chapter 5 evaluates the performance of word-dependent, biphone, and triphone models, both with and without deleted interpolation. The performance of word-dependent models in the Viterbi is compared with their performance in the resort pass, and results from some experiments with the backoff strategy are given.

Boundary models, described in Chapter 6, account for contextual effects by explicitly modeling the region of speech surrounding the transitions from one phonetic unit to another. Their use in the Viterbi search actually achieves the highest performance documented in this thesis, when combined with the word-dependent models in the resorting pass.

Finally, Chapter 8 summarizes the lessons derived from this thesis and presents some suggestions for future work in this area.

Chapter 2

The Search

2.1 Introduction

The goal of the search in a segment-based recognizer is to find the most likely word sequence, given the following information:

- the possible segmentations of the utterance and the measurement vector for each segment,
- acoustic phonetic models, which estimate the likelihood of a measurement vector, given the identity of the phonetic unit,
- a language model, which estimates the probability of a sequence of words, and
- a pronunciation network, which describes the possible pronunciations of words in terms of the set of phonetic units being used.

This chapter describes each of these four components separately, and then describes how the search combines them together to produce the final word sequence.

2.2 Components of the Search

2.2.1 Segmentation

The goal of the segmenter is to divide the signal into regions of speech called *segments*, in order to constrain the space to be searched by the recognizer. From a linguistic point of view, the segments are intended to correspond to phonetic units. From a signal processing point of view, a segment corresponds to a region of speech where the spectral properties of the signal are relatively constant, while the boundaries between segments correspond to regions of spectral change. The segment-based approach to speech recognition is inspired partly by the visual representation of speech presented by the spectrogram, such as the one shown in Figure 1-1, which clearly exhibits sharp divisions between relatively constant regions of speech. Below the spectrogram is a representation of the segmentation network proposed by the segmenter, in which the dark segments correspond to those eventually chosen by the recognizer to correspond to the most likely word sequence. The phonetic and word labels at the bottom are those associated with the path represented by the dark segments.

The segmenter used in this thesis operates heuristically, postulating boundaries at regions where the rate of change of the spectral features reaches a local maximum, and building the segment network S from the possible combinations of these boundaries. Since it is very difficult for the recognizer to later recover from a missed segment, the segmenter intentionally over-generates, postulating an average of seven segments for every one that is eventually included in the recognition output [7].

Mathematically, the segment network S is a directed graph, where the nodes in the graph represente the boundaries postulated by the segmenter and an edge connects node n_i to node n_j if and only if there is a segment starting at boundary b_i and ending at boundary b_j. The Viterbi search will eventually consider all possible paths through the segment network that start with the first boundary and end with the last.

A measurement vector $\mathbf{x}_i$ is calculated based on the frame-based observations contained within each segment s_i [7]. The measurements used in this thesis are a set of 40 proposed by Muzumdar [20]. They consist of averages of MFCC values over parts of the segment, derivatives of MFCC values at the beginning and end of the segment, and the log of the duration of the segment (see Appendix A). From this point onward, the measurement vectors and the segment network are the only information the recognizer has about the signal — gone forever are the frames and their individual MFCC values.

2.2.2 Acoustic Phonetic Models

Acoustic phonetic models are probability density functions over the space of possible measurement vectors, conditioned on the identity of the phonetic unit. A separate acoustic model is created for each phonetic unit, and each is assumed to be independent of the others. Therefore, the following discussion will refer to one particular model, that for the hypothetical phonetic unit $/\alpha/$, with the understanding that all others are defined similarly.

The acoustic models used in this thesis are mixtures of diagonal Gaussian models, of the following form:

$$p(\mathbf{x} \mid \alpha) = \sum_{i=1}^{M} w_i p_i(\mathbf{x} \mid \alpha),$$

where M is the number of mixtures in the model, $\mathbf{x}$ is a measurement vector, and each $p_i(\mathbf{x})$ is a multivariate normal probability density function with no off-diagonal covariance terms, whose value is scaled by a weight w_i. To *score* an acoustic model $p(\mathbf{x} \mid \alpha)$ is to compute the weighted sum of the component density functions at the given measurement vector $\mathbf{x}$. Note that this score is not a probability, but rather simply the value of the function evaluated at the measurement vector $\mathbf{x}$.[1] For pragmatic

[1]To speak of probability one must consider a *range* of possible vectors, over which the PDF is

reasons, the log of the value is used during computation, resulting in what is known as a *log likelihood score* for the given measurement vector.

The acoustic model for the phonetic unit $/\alpha/$ is trained from previously recorded and transcribed speech data. More specifically, it is trained from the set $\mathcal{X}_\alpha$ of measurement vectors corresponding to segments that, in the training data, were labeled with the phonetic unit $/\alpha/$.

The training procedure is as follows:

1. Divide the segments in $\mathcal{X}_\alpha$ into M mutually exclusive subsets, $\mathcal{X}_{\alpha 1} \ldots \mathcal{X}_{\alpha M}$, using the k-means clustering algorithm [5].

2. For each cluster $\mathcal{X}_{\alpha i}$, compute the sample mean $\mu_{\alpha i}$ and variance $\sigma^2_{\alpha i}$ of the vectors in that cluster.

3. Construct, for each cluster $\mathcal{X}_{\alpha i}$, a diagonal Gaussian model $p_i(\mathbf{x} \mid \alpha)$, using the sample mean and variance as its parameters, $p_i(\mathbf{x} \mid \alpha) \sim N(\mu_{\alpha i}, \sigma^2_{\alpha i}\mathbf{I})$. Estimate the weight of each cluster w_i as the fraction of the total number of feature vectors included in that cluster.

4. Re-estimate the mixture parameters by iteratively applying the EM algorithm until the total log prob of the data converges [5].

2.2.3 The Language Model

The language model assigns a probability P to a sequence of words $w_1 w_2 \ldots w_k$. For practical reasons, most language models do not consider the entire word sequence at once, but rather estimate the probability of each successive word by considering only the previous few words. An n-gram, for example, conditions the probability of a word on the identity of the previous $n-1$ words. A bigram conditions the probability of each successive word only on the previous word, as follows:

$$P(w_1 w_2 \ldots w_k) \approx \sum_{i=1}^{k} P(w_i \mid w_{i-1}).$$

The language model for the Resource Management task is a word-pair grammar, which defines for each word in the vocabulary a set of words that are allowed to follow it. This model is not probabilistic, so in order to incorporate it into the probabilistic framework of SUMMIT, it was first converted to a bigram model. The issues involved in this process are subtle, and are explained in more detail in Appendix B.

2.2.4 The Pronunciation Network

The pronunciation network defines the possible pronunciations of each word in terms of the available set of phonetic units, as well as the possible transitions from one word

integrated; the **true** probability of any *particular* measurement vector is precisely zero.

to another. Alternative pronunciations are expressed as a directed graph, in which the arcs are labeled with phonetic units (see Figure 2-1). In the work described in this thesis, the arcs in the graph are unweighted, and thus the model is not probabilistic. Analogous to the case of the word-pair language model, such a pronunciation network could be made probabilistic by considering the network to represent a first order Markov process, in which the probability of each phonetic unit depends only on the previous unit. These probabilities could be estimated from training data or adjusted by hand.

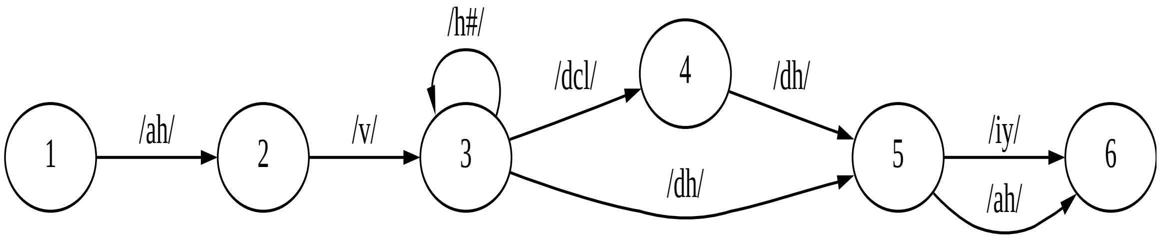

Figure 2-1: Part of a pronunciation network spanning the word sequence "of the."

The structure of the graph is usually fairly simple within a word, but the transitions between words can be fairly complex, since the phonetic context at the end of one word influence those at the beginning of the next. Since, in a large vocabulary system, a word can be followed by many other words, at word boundaries the graph has a very high degree of branching. This complexity, along with the associated computational costs, make the direct inclusion of context-dependent models into the Viterbi search difficult. In fact, many systems that include context-dependent models apply them only within words and not across word boundaries [16]. Those that do apply context-dependent models across word boundaries typically make simplifying assumptions about the extent of cross-word phonetic effects, allowing the acoustic models themselves to implicitly account for such effects.

2.3 The Viterbi Search

The Viterbi search has a complicated purpose. It must find paths through the segment network, assigning to each segment a phonetic label, such that the sequence of labels forms a legal sentence according to the pronunciation network. Of these paths, it must find the one with the highest likelihood score, where the likelihood of a path is a combination of the likelihood of the individual pairings of phonetic labels with segments and the likelihood of the entire word sequence according to the language model.

This task is accomplished by casting the search in terms of a new graph, referred to as the *Viterbi lattice*, which captures the constraints of both the segmentation

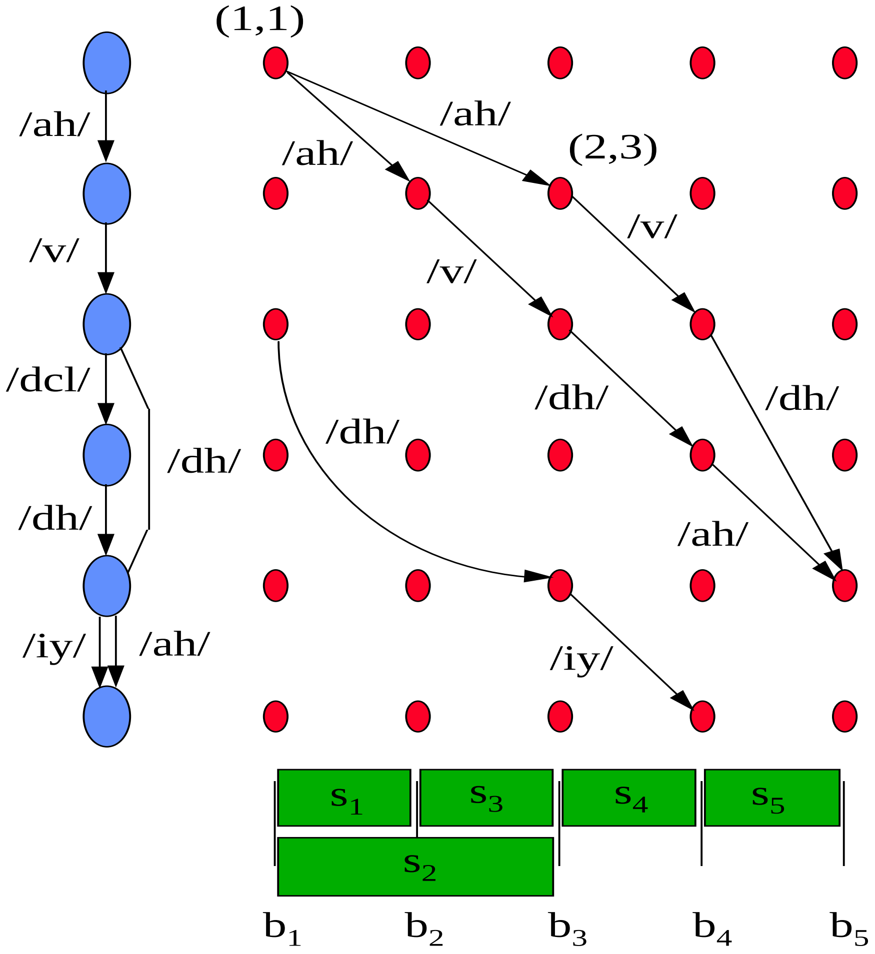

Figure 2-2: A sample Viterbi lattice, illustrating several concepts. An edge connects lattice nodes (1,1) and (2,3) because 1) there is an arc in the pronunciation network between the first and the second node, and 2) there is a segment between the first and the third boundaries. The edge is labeled with the phonetic unit /ah/, and its score is the score of the measurement vector for segment s_2 according to the acoustic model for /ah/. (Note that not all possible edges are shown.) Of the two paths that end at node $(5, 5)$, only the one with the higher score will be maintained.

network and the pronunciation network[2]. Figure 2-2 shows a part of an example Viterbi lattice. Columns in the lattice correspond to boundaries between segments. Rows correspond to nodes in the pronunciation network. There is an edge in the Viterbi lattice from node (i, j) to node (k, l) if and only if:

- there is an arc, labeled with a phonetic unit $/\alpha/$, from node i to node k in the pronunciation network, and
- there is a segment s (with associated measurement vector $\mathbf{x}$) starting at boundary j and ending at boundary l.

This edge is labeled with the phonetic unit $/\alpha/$, and its weight is the log likelihood score given by the acoustic model $p(\mathbf{x} \mid \alpha)$.

In a graph that obeys these constraints, any path that starts at the first boundary and ends at the last will have traversed the segment network completely, accounting for the entire speech signal, and will also have generated a legal path of equal length through the pronunciation network. The goal of the Viterbi search is to find the highest scoring such path, where the score for a path is the sum of the edge weights along that path.

The Viterbi search accomplishes this goal by considering one boundary at a time, proceeding from the first to the last. (The graph is not built in its entirety at the beginning, but rather is constructed as necessary as the search progresses.) To assist the search as it progresses, nodes in the Viterbi lattice are labeled with the score of the highest scoring partial path terminating at that node, as well as a pointer to the previous node in that path. At each boundary, the search considers all the segments that arrive at that boundary from some previous boundary. For each segment, say from boundary j to boundary l, there is a set of labeled edges in the Viterbi lattice that join the nodes in column j with nodes in column l. For each edge, if the score of the node at the start boundary, plus the acoustic model score of the segment across that edge, is greater than the score of the node at the end boundary (or if this node is not yet active), then the score at the end node is updated to reflect this new, better partial path. When such a link is created, a back pointer from the destination node to the source node must be maintained so that, when the search is finished, the full path can be recovered. Figure 2-3 summarizes the algorithm described above.

This sort of search is possible only because the edges in the Viterbi lattice all have the same direction. Once all edges that arrive at a boundary have been considered, the nodes for that boundary will never again be updated, as the search will have proceeded past it in time, never to return. This property suggests a method of pruning the search, which is essential for reducing the cost of the computation. Pruning occurs when, once a boundary has been completely updated, any node along that boundary whose score falls below some threshold is removed from the lattice. As a result, the search is no longer theoretically admissible (*i.e.,* guaranteed to find the optimal

[2]Mathematically, the Viterbi lattice is the graph intersection of the pronunciation network and the segment network.

```
for each boundary b_to in the utterance
    let best_score(b_to) = -∞
    for each segment s that terminates at boundary b_to
        let x be the measurement vector for segment s
        let b_from be the starting boundary of segment s
        for each node n_to in the pronunciation network
            for each pronunciation arc a arriving at node n_to
                let n_from be the source node of arc a
                if (n_from, b_from) has not been pruned from the Viterbi lattice
                    let α be the label on arc a
                    let acoustic_score = p(x | α)
                    if (score(n_from, b_from) + acoustic_score > score(n_to, b_to))
                        score(n_to, b_to) = score(n_from, b_from) + acoustic_score
                        make a back pointer from (n_to, b_to) to (n_from, b_from)
                        if score(n_to, b_to) > best_score(b_to)
                            let best_score(b_to) = score(n_to, b_to)
    for each node n_to in the pronunciation network
        if best_score(b_to) - score(n_to, b_to) > thresh
            prune node (n_to, b_to) from the Viterbi lattice
```

Figure 2-3: A more concise description of the (simplified) Viterbi algorithm.

path), since it is conceivable that a partial path starting from a pruned node might in fact have turned out to be the best one, but in practice pruning at an appropriate threshold reduces computation costs without significantly increasing the error rate.

Finally, because the search only considers a small part of the lattice at any given time, it can operate time-synchronously, processing each boundary as it arrives from the segmenter. This sort of pipelining is one of the primary advantages of the Viterbi search, since it allows the recognizer to keep up with the speaker. More general search algorithms that do not take advantage of the particular properties of the search space might fail in this regard.

2.4 The A* Search

A drawback of the Viterbi search is that, by keeping alive only the best partial path to each node, there is no information about other paths that might have been competitive but not optimal. This drawback becomes more severe if, as is often the case, more sophisticated natural language processing is to take place in a later stage of processing. Furthermore, the Viterbi makes decisions based only on local information. What if the best path from the Viterbi search makes no sense from a linguistic point of view? The system would like to be able to consider the next-best alternative.

Before understanding how this goal is achieved in the current system, it is important to first understand the basic algorithm employed by an A* search [28]. A* search is a modified form of best-first search, where the score of a given partial path is a combination of the distance along the path so far and an estimate of the remaining distance to the final destination. For example, in finding the shortest route from Boston to New York and using the straight-line distance as an estimate of the remaining distance, the A* search will avoid exploring routes to the north of Boston until those to the south have been proven untenable. A simple best-first search, on the other hand, would extend partial paths in an ever-expanding circle around Boston until finally arriving at one that eventually hits New York. In an A* search in which the goal is to find the path of minimal score (as in the above example), the first path to arrive at the destination is guaranteed to be the best one, so long as the estimate of the remaining distance is an underestimate.

In SUMMIT, the goal of the A* search is to search backward through the Viterbi lattice (after the Viterbi search has finished), using the score at each node in the lattice as an estimate of the remaining score [27]. Since the goal is to find paths of maximum score, the estimate used must be an overestimate. In this case it clearly is, since the Viterbi search has guaranteed that any node in the lattice is marked with the score of the *best* partial path up to that node, and that no path with a higher score exists.

As presented here, however, the A* search does not solve the problem described above, for two reasons:

1. In the case of two competing partial paths arriving at the same node, the lesser is pruned, as in the Viterbi search. Maintaining such paths would allow the

discovery of the N-best paths, but would lead to an explosion in the size of the search.

2. The system is only interested in paths that differ in their word sequence. Two paths that differ in the particular nodes they traverse but produce the same word sequence are no different from a practical point of view.

The goal of the system, therefore, is to produce the top N most likely word sequences, not simply the top N paths through the lattice. This goal is accomplished by a combination of A* and Viterbi searches as follows [10]: The A* search traverses the Viterbi lattice backward, extending path hypotheses by one *word* at a time, using the score from the forward Viterbi search at each node as an overestimate of the remaining score. In the case where two paths covering the same word sequence arrive at the same boundary in the lattice, the inferior path is pruned away. During the search, however, many paths encoding the same word sequence might exist at any given time, since they might terminate at different boundaries. Since all complete paths must end at the first boundary, no two *complete* paths will contain the same word sequence, and thus the A* search is able to generate, in decreasing order of score, a list of the top N distinct paths through the lattice.

The extension of a partial path by an entire word is accomplished by a mini backward Viterbi search. A partial path is extended backward by one word by activating only those nodes in the lattice belonging to the new word and performing a Viterbi search backward through the lattice as far as possible. Each backward search terminates promptly because, as in the Viterbi search, partial paths that differ from the best path by a large enough score are pruned. Once the backward Viterbi has finished, the original partial path is extended by one word to create new paths, one for every terminating boundary.

2.5 Resorting the N-best List

The output of the A* search is a ranked list of the N best scoring paths through the Viterbi lattice, where each path represents a unique word sequence. The next stage of processing re-scores each hypothesis using more refined acoustic models, and then re-ranks the hypotheses according to their new scores. The correct answer, if it is present in the N-best list, should achieve a higher likelihood score (using the more refined models) than the competing hypotheses, and will thus be placed in the first position in the list.

The re-scoring algorithm is fairly simple. For each segment in each path, identify the context-dependent model that applies to it, and increment the total score for the path by the difference between the score of the context-dependent model and the score of the context-independent model. In the case that no context-dependent model applies to a segment, skip it. In theory, when a context-dependent model *does* apply to a segment, the context-dependent model should score better than the context-independent model in cases where the context is assumed correctly, worse otherwise.

-398.5504	74.7509	**draw the chart of siberian sea**
-406.4904	74.1970	add the chart of siberian sea
-409.4144	76.4126	can the chart of siberian sea
-415.1426	75.1856	turn the chart of siberian sea
-420.1019	76.9099	count the chart of siberian sea

Figure 2-4: A re-sorted N-best list. The first value is the score given by the resorting procedure, while the second is the original score from the A* search. Notice that the correct hypothesis (in bold) was originally fourth in the N-best list according to its score from the A* search.

The alternative to re-scoring the hypotheses of the N-best list is to use the more sophisticated models in the first place, during the Viterbi or A* search. This approach can be difficult for two reasons. The first is computational: more sophisticated models are often more specific models, of which there are many, and scoring so many models for each segment of speech may be prohibitively expensive from a computational point of view. The second is epistemological: the more sophisticated models may require knowledge of the context surrounding a segment, which can not be known during the search, since the future path is as-of-yet undetermined. This second problem could be overcome in the search by postulating all contexts that are possible at any given moment, but this strategy leads back to the first problem, that of computational cost.

2.6 A Note about the Antiphone

The framework described above for comparing the likelihood of alternative paths through the Viterbi lattice is flawed (from a probabilistic point of view) in that it compares likelihoods that are calculated over different observation spaces. That is, two hypotheses that span the same speech signal but traverse different paths through the segment network are represented by different sets of feature vectors. For example, although the two paths shown in Figure 2-2 that end at node $(5, 5)$ both cover the same acoustics, one is represented by a series of four measurement vectors, while the other is represented by only three. To say that one of the paths is more likely than the other is misleading. More precisely speaking, one path can be said to be more likely than the other only *with respect to its segmentation.* Since the alternative segmentations of an utterance are not probabilistic, this comparison is not valid without some further mechanism.

This problem was apparent many years ago [21, 22], but has only recently been addressed in a theoretically satisfying manner [7]. The solution involves considering the observation space to be not only the segments taken by a path, but also those *not* taken by the path. Doing so requires the creation of an *antiphone* model, which is trained from all segments that in the training data do *not* map to a phonetic unit. In practice, this means that whenever one considers the likelihood of a phonetic unit for a segment, one must actually take the ratio of the likelihood of that phonetic unit to the likelihood of the antiphone unit. Otherwise, the components of the system interact as previously described.

2.7 Implementation in Sapphire

One of the goals of this thesis was not only to experiment with different types of context-dependent modeling techniques, but also to implement the code in such a way that the benefits, if any, would be available to others who wish to take advantage of them. The Sapphire framework, developed here at MIT [9], provides a common mechanism whereby different components of the recognizer can be specified as objects which can communicate with one another by established protocols. The procedure described above for re-scoring the hypotheses of the N-best list has been implemented as a Sapphire object, which fits nicely into the existing structure of SUMMIT for several reasons. First, the context-dependent models are applied as a distinct stage in the processing, independent of the implementation of previous stages. Incorporating context-dependent models directly into the Viterbi search, for example, would not enjoy such an advantage. Second, the application of context-dependent models is the last stage of recognition, and therefore is relatively free from the demands of later stages. (A change in the output of the classifiers, on the other hand, would wreak havoc in the Viterbi and A* searches.) Finally, its output, an N-best list, is of the same form as the output of the A* search, which was previously the last stage of recognition, and thus any components, such as natural language processing, that depend on the output of the A* search will function without change on the output of the resorting module.

The following is an example of the Tcl code that specifies the configuration of the Sapphire object that handles the context-dependent re-scoring procedure:

```
s_resort resort \
    -astar_parent astar \
    -ci_parent seg_scores \
    -cd_parent cd_seg_scores \
    -type TRIPHONE
```

This code instructs Sapphire to create a new object, called `resort`, that is a child of three parents: an A* object called `astar` and two classifier objects called `seg_scores` and `cd_seg_scores`, all of which are Sapphire objects declared previously in the file. These objects are parents of the `resort` object because `resort` requires their output before it can begin its own computation. The fourth argument, `-type`, on the other hand, is simply an argument which tells the `resort` object what type of context-dependent modeling to perform. In this case the models are to be applied as triphone models, but more complicated schemes might be possible, such as backoff strategies or other means of combination.

Sapphire forces the `resort` object to wait until its parents have completed their computation before proceeding. The only significant computation performed by the `resort` object is the scoring of context-dependent models, once for every segment in every path. Since, by the nature of the N-best list, the paths are for the most part identical, in the majority of cases these scores will be available directly from a cache that is maintained for such purposes, avoiding the need to re-compute them. Therefore, the most significant cost of using context-dependent models is the memory required to store them.

Finally, the `resort` object includes an output method that returns the re-sorted N-best list. This method has the same interface as that provided by the A* object, and thus can be used by any existing system that previously depended upon the output of the A* search. The full Sapphire specification of the recognizer used in this thesis is presented in Appendix D.

Chapter 3

Experimental Framework

3.1 Introduction

The research described in this thesis is driven by the goal of improving the performance of the recognizer. It is necessary, therefore, to characterize the performance of the current system, in order to establish a reference point against which progress can be measured. This chapter presents the details of the context-independent baseline configuration, as well some results it achieves on the test data that will be used to evaluate all systems. First, however, is a description of the corpus that forms the basis of the experiments reported in this thesis.

3.2 Resource Management

The DARPA Resource Management database was designed to provide a set of benchmark materials to train, test, and evaluate speech recognition systems [23]. The domain is the management of naval resources, and the utterances are queries a user might make to a computer system containing information about the location and state of various naval resources.[1] The data consist of read speech, where the sentences have been generated by a hand-crafted finite state grammar with a vocabulary of 997 words. Also provided is a word pair grammar (derived from the finite state grammar) which has a test set perplexity of approximately 60 (see Appendix B).

The Resource Management task is a useful framework for experimentation for two main reasons: first, it is a reasonably realistic simulation of the sorts of tasks we hope to accomplish with speech recognition; and second, as a result of annual DARPA evaluations, many papers have been published regarding performance on the task, including the gains reported by various systems as a result of incorporating context-dependent models [18, 15]. (See Table 3-1.)

The data in the Resource Management corpus are divided into separate sets for training and testing. Standard practice is to train on a 109-speaker training set,

[1] For example, “How many vessels are in Indian ocean?” and, for the fricative inclined, “Does sassafras have the largest fuel capacity of all Siberian Sea submarines?”

System	Configuration	% Error
HTK V1.4 (Woodland & Young '93 [30])	triphone HMM	4.1
SPHINX (Huang et al. '91 [12])	GD triphone HMM	3.8
SUMMIT (Hazen '97, work in progress)	CI + bounds	3.8

Table 3-1: The best results in the literature published for Resource Management's Feb. 1989 evaluation. (The error rates of 3.8% are actually for a slightly different, but harder test set.)

which I will call *train109*. This set contains a total of 3990 utterances, 2880 of which are from 72 speakers (40 utterances per speaker) and 1110 of which are from the remaining 37 speakers (30 utterances per speaker). Each speaker also recorded two "speaker-adaptation" utterances, which are not included in the training set. 78 of the speakers are male, and 31 are female.

The development test data used in the experiments described in this thesis are the data released for official evaluation purposes in February of 1989 (*test89*). The data consist of 300 utterances from 10 speakers, none of which is represented in the training data.

3.3 The Baseline Configuration

The baseline system classifies segments into one of 60 context-independent classes, corresponding to the units used to transcribe the TIMIT database [6], with the exception of /eng/. The majority of the experiments described in this thesis involve subdividing these classes into several according to context. Otherwise, in most cases, the configuration of the system remains as described below.

The measurements used throughout this thesis are 39 averages and derivatives of MFCC values across various portions of the segment, plus the log of the duration of the segment. They are due to Muzumdar [20] and are described more exactly in Appendix A. The acoustic models are mixtures of diagonal Gaussian models, with a maximum of 30 mixtures per model and a minimum of 25 training tokens per mixture.[2] In the case of the 60 context-independent classes, all but three (*em*, *en*, and *zh*) have sufficient training data to create all 30 mixtures. In the case of context-dependent models, very rarely does the number of mixtures approach the upper bound.

The language model is the bigram representation of the word-pair grammar, as described in Appendix B. This language model has a test-set perplexity of 62.26 on the 1989 evaluation set. The pronunciation network is created automatically by the application of rules to the baseform pronunciations for each word. There are no

[2] That is, the number of mixtures used to represent a class is either 30 or the number of mixtures possible with 25 tokens per mixture, whichever is smaller.

weights on arcs of the pronunciation network, and so all paths through the network are assigned equal likelihood.

The Viterbi search and the A* search are both characterized by pruning thresholds. Typically, the Viterbi threshold is fixed at 15.0 and the A* threshold at 30.0. Increasing either threshold decreases the amount of pruning, increases computation cost, and (usually) improves performance. The values presented here are used because they provide sufficient pruning to approach real-time performance, without sacrificing a great deal of accuracy.

The recognizer includes a variety of other configurable parameters, but these are held constant throughout the thesis and do not warrant discussion here.

3.4 Baseline Performance

Most of the experiments performed in this thesis evaluate models trained on the 109-speaker training set and tested on the 10-speaker 1989 evaluation set. As discussed in Appendix C, in order to accurately assess the performance of the system, it is necessary to run multiple trials. In the following experiment, 15 sets of context-independent models were trained on the 109-speaker training set. Each was evaluated on the 1989 test-set, yielding an average word error rate of 13.7%.

Trial	Word Error Rate	Substitutions	Insertions	Deletions
1	14.4 %	9.8 %	2.4 %	2.3 %
2	13.1 %	8.7 %	1.9 %	2.5 %
3	12.8 %	8.7 %	1.9 %	2.2 %
4	14.3 %	9.8 %	2.1 %	2.4 %
5	13.9 %	9.5 %	1.9 %	2.5 %
6	13.7 %	9.3 %	2.1 %	2.4 %
7	13.8 %	9.8 %	1.7 %	2.3 %
8	13.2 %	9.1 %	1.9 %	2.2 %
9	13.7 %	9.4 %	2.0 %	2.3 %
10	13.5 %	8.9 %	2.0 %	2.6 %
11	13.9 %	9.7 %	1.8 %	2.3 %
12	13.7 %	9.3 %	2.0 %	2.4 %
13	13.9 %	9.3 %	2.0 %	2.5 %
14	13.9 %	9.5 %	2.2 %	2.2 %
15	13.7 %	9.4 %	1.8 %	2.4 %
mean	**13.7 %**	**9.35 %**	**1.98 %**	**2.37 %**
range	1.60 %	1.10 %	0.70 %	0.40 %
uncertainty	0.21 %	0.19 %	0.09 %	0.06 %

Table 3-2: Baseline results for context-independent models on test89.

The word error rate is the sum of the substitutions, insertions, and deletions. The uncertainty in the measurement of the word error rate is calculated as described in Appendix C. In this case, for example, if one were to perform an infinite number of experiments, the word error rate averaged over all experiments would differ from 13.70% by more than 0.21% with a probability of 0.05. The range of possible results, however, is quite large (1.6%), clearly demonstrating the necessity of performing multiple experiments[3].

Note that the results presented here are not competitive with those published in the literature (Table 3-1). The reason is that the system presented above was designed not to maximize performance, but rather to provide a solid baseline with respect to which further enhancements can be evaluated. Performance can nominally be improved by increasing the number of measurements per segment, the number of mixtures per model, or the pruning thesholds of the Viterbi and A* searches. Futhermore, corrective training could be used to optimize weights on the arcs of the pronunciation network, and other parameters throughout the system could be tuned by hand. Doing so, however, would provide little benefit to the value of comparative experimentation and would be contrary to the goal of designing a system that keeps computational costs within the range acceptable in a real-time system.

3.5 N-best Performance

The strategy of re-scoring the hypotheses of the N-best list is worthless if the top N choices do not include the correct answer, or at least utterances that are "more" correct than the first choice. Furthermore, one would like to know what value of N should be used. Is the correct answer almost always within the top 10 hypotheses? Within the top 100? What is the word error rate that we could achieve if we could choose the best among the hypothesis from the N-best list?

An experiment was designed to answer these questions. The first step was to compute N-best lists of size $N = 100$ for each of the 300 utterances in the 1989 test set, using a set of models trained on the 109-speaker training set. Then, for each value of N between 1 and 100, two statistics were calculated:

1. the word error rate across all 300 utterances, where each utterance is represented by the *best* among the top N hypotheses, and

2. the sentence error rate on the same data; that is, the percent of utterances for which the correct answer is not included as one of the top N hypotheses.

Figure 3-1 is a plot of the two statistics, which shows that a great deal of improvement is possible if the recognizer can learn to extract the best among the hypotheses represented in the N-best list. (Table 3-3 gives the exact values.) In some ways, this problem is much simpler than the original problem presented to the recognizer, due

[3]All results published in this thesis are calculated as the average of several trials (usually 15), and the word error rate is typically accurate to within 0.25%

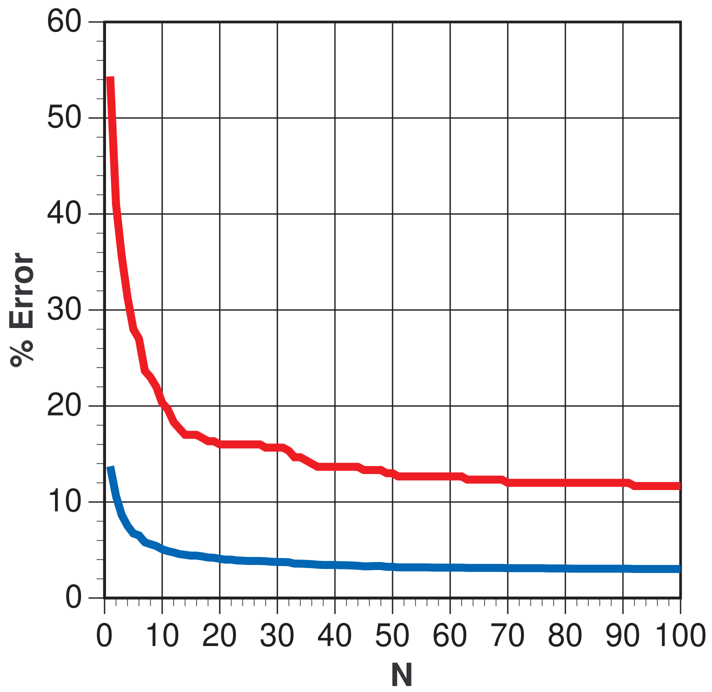

Figure 3-1: A plot of word and sentence error rate for an N-best list as a function of N. The upper curve is sentence error rate.

to the severely constrained search space embodied in the N-best list. In others, it is more difficult, since the differences between alternatives in the N-best list are often very slight, hinging on only a few segments of speech. But precisely because of the constraints imposed by the limited number of possibilities, the recognizer can afford to apply more computationally intensive techniques than during the first pass. One such technique is to use context-dependent acoustic models, the results of which are presented in the remaining chapters of this thesis.

N	**Word Error Rate**	**Sentence Error Rate**
1	13.7 %	54.3 %
2	10.6 %	41.0 %
10	5.1 %	20.3 %
25	3.9 %	16.0 %
50	3.2 %	13.0 %
100	3.0 %	11.7 %

Table 3-3: Word and sentence error rate on test89 as the length of the N-best list increases.

Chapter 4

Deleted Interpolation

4.1 Introduction

The training of context-dependent models is hindered by what is known as the "sparse data problem." That is, for very specific contexts, it is unlikely that there will be enough examples to train robust models; on the other hand, more general contexts lack the specificity that was the original goal of building context-dependent models. One solution to this tradeoff involves a technique called *deleted interpolation* [13], in which specific models are interpolated with more general models according to their performance on unseen data. This chapter introduces the concept of deleted interpolation and describes its implementation within a segment-based speech recognition system.

4.2 Deleted Interpolation

Given a particular context-dependent model $P_{CD}(\cdot)$ and its associated context-independent model $P_{CI}(\cdot)$, an interpolated model $P_{DI}(\cdot)$ can be created as a linear combination of the two as follows [11]:

$$P_{DI}(\cdot) = \lambda P_{CD}(\cdot) + (1 - \lambda) P_{CI}(\cdot). \tag{4.1}$$

The goal is to interpolate the two component models such that the resulting model approximates the context-dependent model ($\lambda \approx 1$) when it is well-trained but "backs off" to the context-independent model ($\lambda \approx 0$) otherwise. Deleted interpolation is a technique for estimating the values of λ by measuring the ability of each model to predict unseen data. The basic procedure is as follows [11]:

1. Partition the training set $\mathcal{T}$ into two disjoint sets, $\mathcal{T}_1$ and $\mathcal{T}_2$, such that $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$.

2. Train both the context-dependent model $P_{CD(\mathcal{T}_1)}(\cdot)$ and the context-independent model $P_{CI(\mathcal{T}_1)}(\cdot)$ on the data in $\mathcal{T}_1$.

3. Given the models trained in the previous step, estimate the value of λ for each model using the "held-out" data in $\mathcal{T}_2$. (See Equation 4.2 below.)

4. Train both the context-dependent model $P_{CD(\mathcal{T})}(\cdot)$ and the context-independent model $P_{CI(\mathcal{T})}(\cdot)$ on the entire training set $\mathcal{T}$.

5. Use the value of λ calculated in step 3 to create the deleted interpolation model $P_{DI(\mathcal{T})}(\cdot)$ as a linear combination of the models trained by the previous step, as per Equation 4.1.

In Step 3, the value of λ for the model is initially set to 0.5 and then iteratively re-calculated, according to the following formula, until it converges:

$$\lambda' = \frac{1}{N}\sum_{i=1}^{N}\frac{\lambda P_{CD(\mathcal{T}_1)}(\mathbf{x}_i)}{\lambda P_{CD(\mathcal{T}_1)}(\mathbf{x}_i) + (1-\lambda)P_{CI(\mathcal{T}_1)}(\mathbf{x}_i)}. \tag{4.2}$$

In Equation 4.2 the $\mathbf{x}_i$ are the N measurement vectors in $\mathcal{T}_2$ representing the segments for which the context-dependent model is applicable. An extension to the basic method (known as "jack-knifing") rotates $\mathcal{T}_2$ through the training data [11]. In other words, the training set $\mathcal{T}$ is partitioned into M disjoint sets $\mathcal{T}_1 \ldots \mathcal{T}_M$, and the calculation of λ is performed over the entire training set, applying to each feature vector $\mathbf{x} \in \mathcal{T}_j$ the models trained on $\mathcal{T} - \mathcal{T}_j$:

$$\lambda' = \frac{1}{N}\sum_{j=1}^{M}\sum_{i=1}^{N_j}\frac{\lambda P_{CD(\mathcal{T}-\mathcal{T}_j)}(\mathbf{x}_{i,j})}{\lambda P_{CD(\mathcal{T}-\mathcal{T}_j)}(\mathbf{x}_{i,j}) + (1-\lambda)P_{CI(\mathcal{T}-\mathcal{T}_j)}(\mathbf{x}_{i,j})},$$

where N_j is the number of vectors in $\mathcal{T}_j$, and $\mathbf{x}_{i,j}$ is the i^{th} feature vector in set $\mathcal{T}_j$.

In general, one can combine more than two distributions together in a similar manner [11]. In the case of J distributions $P_1(\cdot) \ldots P_J(\cdot)$, let $\lambda = \lambda_1 \ldots \lambda_J$ such that $P_{DI}(\cdot)$ is defined according to the following formula (a generalization of Equation 4.1):

$$P_{DI}(\cdot) = \sum_{j=1}^{J}\lambda_j P_J(\cdot),$$

where the values of λ are forced to sum to one:

$$\sum_{j=1}^{J}\lambda_j = 1.$$

The values for λ_j are solved by a formula that is a generalization of the two model case (Equation 4.2):

$$\lambda_j' = \frac{1}{N}\sum_{i=1}^{N}\frac{\lambda_j P_j(\mathbf{x}_i)}{P_{DI}(\cdot)}.$$

In this way, for example, one might consider interpolating a triphone model with both left and right biphone models and with the context-independent model.

4.3 Incorporation Into SUMMIT

The method described above for interpolating models is applicable regardless of the underlying nature of the original probability density functions themselves. In SUMMIT, however, the acoustic models are mixtures of diagonal Gaussian models, and therefore a new interpolated model of the same form can be constructed directly from the component models.

An acoustic model in SUMMIT, for the phonetic unit $/\alpha/$, is of the form:

$$p(\mathbf{x} \mid \alpha) = \sum_{i=1}^{M} w_i p_i(\mathbf{x} \mid \alpha),$$

where M is the number of mixtures in the model, $\mathbf{x}$ is a measurement vector, and each $p_i(\mathbf{x})$ is a multivariate normal probability density function with no off-diagonal covariance terms (see Section 2.2.2).

A new model, $p_{DI}(\mathbf{x} \mid \alpha)$, which interpolates component models $p_1(\mathbf{x} \mid \alpha) \ldots p_J(\mathbf{x} \mid \alpha)$ with interpolation parameters $\lambda_1 \ldots \lambda_J$, can be created as follows:

$$p_{DI}(\mathbf{x} \mid \alpha) = \sum_{j=1}^{J} \sum_{i=1}^{M_j} \lambda_j w_{i,j} p_{i,j}(\mathbf{x} \mid \alpha)$$

where $w_{i,j}$ and $p_{i,j}(\mathbf{x} \mid \alpha)$ refer to the i^{th} mixture component of the j^{th} model in the interpolation. The resulting model, just like its components, is a mixture of M diagonal Gaussian models, where

$$M = \sum_{j=1}^{J} M_j.$$

4.4 Chapter Summary

Deleted interpolation, as described above, provides a mathematical framework for compensating for the fact that some models will not perform well due to a lack of training samples. This is accomplished by creating, for each context, a new model that is a linear combination of the model for that context and the context-independent model. Th weights used to perform the interpolation are determined automatically by a procedure which measures the relative ability of each component model to predict unseen data. The following chapter will demonstrate the use of deleted interpolation with context-dependent segment models such as biphones, triphones, and word-dependent phone models.

Chapter 5

Traditional Context-Dependent Models

5.1 Introduction

The traditional strategy for modeling context-dependency has been to change the unit being modeled from *phone* to *phone-in-context* [18]. The context might be the word in which the phone occurs, as in the case of word-dependent phonetic models, or the phones immediately preceding or following the phone, as in the case of biphone and triphone models. Whatever the context, the issues involved with the creation and use of such models are largely the same. The following sections describe the motivation behind the use of such models, followed by a presentation of the experimental framework used to evaluate the performance of systems that incorporate such models. Finally, some results are given that summarize the performance when such models are used to resort the N-best list output by the context-independent system.

5.2 Word-Dependent Models

One of the simplest forms of context-dependent modeling considers the context of a phone to be the identity of the word in which it occurs, creating *phone-in-word* models by training, for each phone, a separate model for each word in which it occurs [3]. For example, rather than having just one model for the phone /ae/, as in the context-independent case, there would be many such models, one for each word in the vocabulary that contains an /ae/.

This strategy is unsatisfying in that it does not capture the underlying *cause* of the variability in the pronunciation of phones. Although it is true that the same phone may be realized differently in different words, the difference derives not from the fact that the words are different, but rather from the differences in the phonetic context surrounding the unit in question. Presumably, two phones with the same surrounding context will be realized similarly, whether or not they occur in the same word.

On the other hand, the naivete of word-dependent modeling is also part of its strength. Whereas context-dependent models typically account only for contextual effects from adjacent phones, word-dependent models can implicitly capture more far-reaching contextual effects, as well as other influences whose source might be more difficult to ascertain. In other words, word-dependent models can capture contextual effects that are not well understood precisely *because* they make few assumptions about the source of the variability.

5.3 Biphone and Triphone Models

Biphones and triphones are the most commonly used type of context-dependent model. A *left biphone* is dependent on the identity of the preceding phone, while a *right biphone* is dependent on the identity of the following phone. A *triphone* model is conditioned on the identities of *both* the preceding phone and the following phone.

The primary motivation behind such models is that the variation observed in the realization of a phone is thought to be due largely to the co-articulatory influences of its neighbors. A model trained only from examples that occur in a particular phonetic context is sheltered from this sort of variability, and thus should be able to more effectively discriminate itself from similar classes.

One advantage of triphones, over word-dependent models at least, is that they are independent of vocabulary and that, presumably, as the vocabulary size grows, the number of triphone models required should remain relatively constant, while the number of word-dependent models will continue to increase. This distinction is analagous to that between phone and word-based acoustic models discussed in the section on previous research.

5.4 Basic Experiments

The purpose of the following experiments is to compare the performance of a system that includes context-dependent models with that of the context-independent baseline system, keeping all other factors constant. In each case, whether the models be word-dependent or biphone or triphone models, the procedure for selecting, training, and testing a set of models is essentially the same.

5.4.1 Choosing a Set of Models

The above strategies work by partitioning the data normally reserved for a single model into many non-overlapping subsets. Because the number of possible contexts is large, the amount of data for any one context is likely to be very small. In fact, for the word-dependent models, 53% of the contexts that occur in the training data are represented by fewer than ten examples. Even worse, 2,495 of the contexts that are possible in the pronunciation network are not represented at all in the training data. This is known as the sparse data problem, and one must have a strategy for resolving it.

A common strategy, at least in the case of triphones, is to merge contexts, creating what are known as *generalized* triphone models [18]. For the sake of simplicity I have chosen not to do so, and to pursue instead the straightforward strategy of creating models only for those contexts that occur a sufficient number of times in the training data. A sufficient number, in the following experiments, is either 50 or 25.

The set of all possible word-dependent models is the set of all phone/word combinations that occur in the pronunciation network.[1] In the Resource Management task, the pronunciation network allows 11,473 phone-in-word combinations, of which 8,978 occur at least once in the training data. 1,665 contexts occur at least 25 times in the training data, and only 690 occur at least 50 times. The statistics are similar for the biphones and triphones, and are summarized in Table 5-1.

	Number of Contexts in Train109		
Model Type	**Total**	**25 or More Times**	**50 or More Times**
Context-Independent	60	60	60
Word-Dependent	8978	1665	699
Biphone	1589	860	633
Triphone	9239	1662	882

Table 5-1: The number of contexts represented in the training data for each type of context-dependent model, with cut-offs of either 25 or 50 training tokens.

5.4.2 Training

Once the set of models has been determined, they are trained by the same procedure that is used to train the context-independent models, the only difference being the labeling of the data in the training set, which must be modified to reflect the context of each phone. Segments labeled with contexts not to be included in the final set of models are ignored. Each set of models is trained on the 109-speaker training set, which is the set typically used to train models evaluated in the literature.

5.4.3 Testing

The *test-set coverage* of a set of models is defined as the fraction of segments in the test data to which the models can be applied. In the case of context-independent models, the test-set coverage is 100%, since every segment in the test set is classified as belonging to one of the 60 context-independent phonetic classes. Triphones, biphones, and word-dependent models, on the other hand, do not account for every segment in the test-set, since many of the possible models of each type were removed from the model set because of a lack of training data.

[1] Others may occur in real speech, but their exclusion from the pronunciation network implicitly precludes them from being considered during recognition.

Table 5-2 summarizes the coverage of each type of context-dependent model on the Feb. 1989 10-speaker test set.

Type of Model	Test-Set Coverage	
	25+ Models	**50+ Models**
Context-Independent	100.0 %	100.0 %
Word-Dependent	68.6 %	51.2 %
Triphone	77.3 %	63.4 %
Biphone	96.8 %	92.9 %

Table 5-2: Test-Set Coverage of context-dependent models.

The most significant result is that the biphone models enjoy a very high coverage of the test set, despite using fewer models than either the triphones or the word-dependent models.

5.4.4 Results

Table 5-3 summarizes the performance achieved by each type of context-dependent model on the utterances in the 10-speaker 1989 test set. Each experiment was repeated 15 times, and the results given in the table are the averages of the results of each trial, accurate to within 0.25% word error rate.

Type of Model	Word Error Rate	
	25+ Models	**50+ Models**
Context-Independent	13.70 %	13.70 %
Word-Dependent	12.27 %	12.25 %
Left-Biphone	11.61 %	12.11 %
Right-Biphone	10.75 %	10.85 %
Triphone	12.11 %	13.24 %

Table 5-3: Summary of the performance of several different types of context-dependent models.

There are several interesting observations to be made regarding these results. First and foremost, the right biphone models perform significantly better than any other type of model. The fact that the 25+ performance is not significantly better than the 50+ performance is not surprising, since the extra 227 models account for an increase of less than four percent in the test-set coverage. What is surprising is the fact that the right biphone models perform so much better than the left biphones, despite the fact that both account for the same fraction of the test data. This result can be taken to suggest that anticipatory co-articulation is more significant than the influence of the previous phone.

A second interesting observation is that the word-dependent models, despite suffering from a particularly low test-set coverage, perform remarkably well. (Another way to look at it is that the triphone models perform particularly poorly when only those with 50 or more examples are used.) The lack of improvement in the word-dependent models, however, from the 50+ case to the 25+ case, is puzzling, and suggests that many of the 25+ models might be poorly trained. This problem, in fact, probably plagues, to a varying extent, all of the models described in this section. The technique of deleted interpolation, introduced in the next chapter, aleviates this problem by smoothing context-dependent models with their more general context-independent counterparts. See Table 5-4 for a summary of the performance of these models when they have first been interpolated with context-independent models.

5.5 Incorporating Deleted Interpolation

The purpose of this experiment is to measure the effect of deleted interpolation on the context-dependent models described in the previous chapter. The models were interpolated with context-independent models according to the procedure described above, using the jack-knifing technique to take advantage of the entire set of training data. The models were then tested on the February 1989 test set, using exactly the same procedure as before. As usual, the results presented in the table below are actually the average taken over 15 independent trials, and are accurate to within 0.25%.

Type of Model	**Non-Interpolated**		**Interpolated**	
	25+ Models	**50+ Models**	**25+ Models**	**50+ Models**
Context-Indep.	13.70 %	13.70 %	13.70 %	13.70 %
Word-Dependent	12.27 %	12.25 %	10.71 % (12.7 %)	11.42 % (6.8 %)
Left-Biphone	11.61 %	12.11 %	11.13 % (4.1 %)	11.15 % (7.9 %)
Right-Biphone	10.75 %	10.85 %	10.65 % (0.9 %)	11.12 % **(-2.5 %)**
Triphone	12.11 %	13.24 %	10.99 % (9.2 %)	11.74 % (11.3 %)

Table 5-4: Summary of the performance of several different types of context-dependent models. For comparison purposes, the first two columns are the results from Table 5-3, and the last two columns are the results of the same models, after being interpolated with context-independent models. The numbers in parentheses are the percent reduction in word error rate as a result of interpolation.

The effect of deleted interpolation on a set of models appears to be to suppress the influence of the bad among them while encouraging the good. Perhaps the best example of this effect the performance of the 25+ word-dependent models, which by rights of increased coverage should perform better than their 50+ counterparts.

This improvement due to increased coverage is only seen with the help of deleted interpolation.

Another interesting result is that the interpolated word-dependent models, despite their lack of coverage, outperform the triphone models, suggesting that there is value in their ability to capture contextual effects specific to particular words.

Finally, the most disappointing result is the decrease in performance of the right biphone models as a result of the application of deleted interpolation, though the difference is not terribly significant. This example serves as a reminder that, although deleted interpolation tries to maximize the performance of models on unseen data from the training set, there are never any guarantees regarding the performance on entirely new data.

5.5.1 Generalized Deleted Interpolation

Typically, deleted interpolation is applied to a combination of specific models, such as triphones, with general models, such as context-independent models. As described in Section 4.2, it is possible to interpolate more than two kinds of models together. One possibility is to interpolate triphone models not only with context-independent models, but also with left and right biphone models, leaving to the deleted interpolation the task of determining which component models are valuable and which are not. As before, examining the weights given by the deleted interpolation can provide us with some insights into the relative value of various types of context-dependent models.

Table 5-5 summarizes the results of experiments where the triphone models were interpolated with context-independent and biphone models, in various combinations.

TRI	RB	LB	CI	Word Error Rate
X				13.24 %
X			X	11.74 %
X	X	X	X	11.77 %
X	X	X		11.76 %
	X	X	X	12.07 %

Table 5-5: Results of experiments in which triphone models were interpolated with left and right biphone models and context-independent models, in various combinations. In no case did the word error rate improve over the simple interpolation with the context-independent models only.

The last row of the table indicates an interesting experiment where triphone models are created as an interpolated combination of biphone models and the context-independent model, but without including the triphone model itself. Mathematically, there is nothing special about this case, since the procedure simply evaluates the relative performance of the component models (whatever they might be) on the segments that apply to the interpolated model. This strategy would be ideal for the creation of triphones that are poorly represented by the training data, except for the fact that the triphone must appear relatively frequently in order to serve as a basis for the estimation of the interpolation weights. In any case, the performance is surprisingly good in this case, though worse than cases that do include the triphone model itself.

The overall results of this experiment show that interpolation with the context-independent models is sufficient, and that adding intermediate types of models to the interpolation does not significantly change the results.

5.6 Back-off Strategies

As discussed previously, the method used in this thesis to choose each set of context-dependent models does not guarantee complete coverage of the tokens in the test set. In fact, triphone models chosen with the 50+ criterion cover only 63.4% of the test data—no wonder they do not perform as well as the biphone models, which achieve a test-set coverage of 92.9%. Presumably, triphone models are better models, on the tokens to which they do apply, than the biphone models. Their drawback is a lack of coverage.

Table 5-6 is an attempt to quantify the notion that a model's true worth must somehow take into account its test-set coverage. The table gives, for each type of model, the percent decrease in word error rate from the baseline system, as well as this percentage divided by the test-set coverage for that type of model. The latter can be regarded as a metric of the "per-token" improvement provided by a set of context-dependent models. The numbers given are for the interpolated version of each type of model, which, except in the case of the right biphones (50+), are superior to the non-interpolated versions.

Models	**# Models**	**Coverage**	**Reduction**	**Adj. Reduction**
LB (25+)	860	96.8 %	18.76 %	19.38 %
LB (50+)	633	92.9 %	18.61 %	20.04 %
RB (50+)	633	92.9 %	18.83 %	20.27 %
RB (25+)	860	96.8 %	22.26 %	23.00 %
TRI (50+)	882	63.4 %	14.31 %	22.57 %
TRI (25+)	1662	77.3 %	19.78 %	25.59 %
WD (25+)	1665	68.6 %	21.82 %	31.81 %
WD (50+)	690	51.2 %	16.64 %	32.50 %
WD+TRI (25+)	1127	40.5 %	14.53 %	35.87 %
WD+TRI (50+)	544	31.0 %	11.17 %	36.03 %

Table 5-6: Percent reduction in word error rate, adjusted to account for test-set coverage. The model sets are all interpolated with the context-independent models. (The last two rows refer to triphone-in-word models, not previously discussed.)

As expected, the most specific models (triphone-in-word) yield the highest adjusted reduction in word error rate, while the least-specific models, the biphones, yield the least adjusted improvement. An interesting comparison is between the word-dependent models and the triphone models, both of which have similar coverage statistics. The word-dependent models exhibit clearly superior performance than the triphones, presumably because they can capture coarticulatory effects that are specific to the word in question, including effects from phones more distant than the immediate neighbors.

The backoff strategy is based on the assumption that one would like to use, whenever possible, the most specific model available. Whenever a very specific model is not applicable to a segment, the algorithm substitutes a less specific model, and so on, until finding some type of model that applies to the segment, or, if none is found, skipping the segment. The order of application suggested by the table is first to use the triphone-in-word models, followed by the word-dependent models, followed by the triphone models, the right biphone models, and finally the left biphone models. The results of several experiments using different possible orderings are summarized in Table 5-7.

Models	**# Models**	**Word Error Rate**	**Reduction**
WD+TRI, WD, TRI, RB, LB	3382	11.11 %	18.9 %
WD+TRI, TRI, RB, LB	2692	10.93 %	20.2 %
WD, TRI, RB, LB	2838	11.13 %	18.8 %
WD+TRI, WD, RB, LB	2500	10.86 %	20.7 %

Table 5-7: The results of various combinations of backoff strategies. The performance is essentially the same for all combinations, and does not represent an improvement over the increased coverage that can be obtained by decreasing the required number of tokens per model.

5.7 Performance in the Viterbi Search

The results presented in this chapter have shown that context-dependent segment models can decrease the word-error rate of the baseline system when used to re-score the hypotheses of the N-best list. However, even the best possible re-sorting mechanism is doomed to some level of failure, since 11.7% of the N-best lists do not even include the correct answer. Perhaps a better strategy would have been to apply the context-dependent segment models directly in the Viterbi search to avoid accidentally pruning away the correct answer.

In the following experiments word-dependent models have been evaluated directly in the Viterbi search. Word-dependent models, rather than triphone or biphone models, were chosen for this experiment because with such models the context is known unambiguously during the search. More specifically, the labels in the pronunciation network can be replaced by inspection of the network itself, whereas in the case of biphone and triphone models, the context of a given arc in the pronunciation network is not known until the search is in progress. Right biphone and triphone models suffer from the further inconvenience that the right context is not available during the search, making their application impossible without a significant increase in computation costs.

Type of Model	Viterbi	Resort
Context-Independent	13.70 %	13.70 %
WD+25 (interpolated)	10.65 %	10.71 %
WD+50 (interpolated)	11.05 %	11.42 %

Table 5-8: A comparison of the performance of word-dependent models in the Viterbi search and in the re-sorting pass. Performance is slightly better in the Viterbi search, though the differences are not very statistically significant (each result is significant to within 0.25%).

Table 5-8 summarizes the results of a comparison between the performance of the word-dependent models in the Viterbi search with their performance in the resorting pass. In these experiments, although there is a slight edge in performance in the Viterbi search, the differences are not very significant compared to the uncertainty in the measurement of the word error rate.

5.8 Chapter Summary

This chapter presented results from several experiments using context-dependent segment models. The most important conclusions are summarized by the following statements:

- Right-biphone models significantly outperform left-biphone and triphone models.
- Deleted interpolation with context-independent models improves performance.
- Context-dependent segment models achieve similar performance when used in the Viterbi search as when used to resort the hypotheses of the N-best list.
- The differences in performance among the different types of models can be attributed in part to the differences in their coverage of the test-set, but applying a backoff strategy from more to less specific models does not improve performance.

The following chapters will present results for boundary models and offset models, both of which are alternative ways of capturing the effects of phonetic context.

Chapter 6

Boundary Models

6.1 Introduction

The segmentation stage that precedes classification actually works by proposing likely *boundaries* between segments, and from these boundaries the segment network is created. Traditionally, models have been trained from measurements calculated for each segment, but in theory there is no reason they should not be trained from measurements calculated at each boundary. Furthermore, the measurements taken at a boundary need not be the same as those taken within a segment, and can be designed specifically to capture the salient features that distinguish among different types of boundaries. As an added benefit, recognition based on boundary models does not suffer from the same observation space disparities that plague the segment modeling strategy, since all paths include the same number of boundaries, so long as boundaries within a segment are included as well as those between segments.

6.2 Boundary Models

There are two types of boundary models: internal boundary models and transition boundary models. The former are trained from examples of those boundaries that, in the forced path, were not chosen by the search to be the endpoint of a segment. That is, they are the boundaries that were proposed by the segmenter but that, after the search, were found to be internal to some segment. The transition boundaries are the remaining ones, those that in the forced path were chosen to separate two segments from one another.

Boundary models are in some sense inherently context-dependent, since the models are conditioned on the identity of the two segments surrounding the boundary. More strictly speaking, however, the boundary models considered here are context independent, since they depend only on the identity of the *boundary* under consideration, and not on any neighboring boundaries or more distant segments, just as context-independent segment models depend only on the identity of the *segment* in question.

Boundary models do, however, share with context-dependent segment models the explicit modeling of co-articulatory effects, as they directly measure the features associated with the transition taken by the vocal tract from one phone to another. They also fit well within the re-scoring framework developed in this thesis for context-dependent models, since boundary models can be scored at a much lower cost in the resorting pass than in the Viterbi search.

The primary difficulty in training transition boundary models is the same one that arises when training context-dependent models: the sparse data problem. If the segments are divided into N distinct classes, then the number of possible types of transitions from one segment to another is N^2, which severely reduces the amount of training data available for each one. In practice, of course, the number is much lower, but still the problem remains that the training data may provide few, if any, examples of transitions that might occur in the test data.

The possible solutions to the problem are much the same as those for context dependent models. Various strategies include data-driven clustering of similar models, the a priori combination of classes by an expert, and the interpolation of poorly trained models with well trained ones. The following sections will explain the details of the two strategies implemented in this thesis.

6.3 Basic Experiments

The purpose of the following experiments is to evaluate the performance of a system that uses boundary models in addition to the normal context-independent segment models. Unlike context-dependent segment models, boundary models can be included in the Viterbi search, and therefore the following experiments the boundary models will be evaluated both in the Viterbi search and as a method for re-sorting the N-best list.

6.3.1 Choosing a Set of Models

Of the 3600 theoretically possible transitions from one phone to another, fewer than half (1589) occur in the training set. Of these, only 634 are represented by 50 or more examples. Those that did not occur in the training set (but still may in the test data) must be accounted for somehow. In the following experiments, two different strategies have been adopted for dealing with such cases.

The first is simple, and involves training one model for each transition that occurs sufficiently often (50 times or more) in the training data, plus one additional "catch-all" model trained from all remaining transitions that actually occur in the training data. In the 109-speaker training set for Resource Management, 634 transitions occur at least 50 times. Since the definition of the pronunciation network allows a total of 2168 transitions, the "catch-all" model covers the remaining 1534 transitions that do not occur with sufficient frequency. Many of these, of course, do not appear at all in the training data, but the "catch-all" model must cover them in case they arise during testing. Although at first glance it appears that the "catch-all" model accounts for the majority of transitions, it actually represents only seven percent of the transitions that occur in the training data, due to the relative infrequency of most of these transitions.

The second strategy is to group transitions together that share a common place of articulation. One such grouping, for example, might include all transitions from some type of silence to another, or between front vowels and retroflexed phones. The particular groupings used in this thesis are those suggested by T.J. Hazen (work in progress), using *a priori* linguistic knowledge, resulting in a set of 547 models covering a total of 2838 types of internal and transition boundaries.[1] The following are some examples of how transitions between similar places of articulation have been grouped together:

- t(r→l) t(r→el) t(er→l) t(er→el) t(axr→l) t(axr→el)
- t(ey→bcl) t(iy→bcl) t(ih→bcl) t(ix→bcl) t(ay→bcl) t(oy→bcl)
- t(h#→f) t(iwt→f) t(pcl→f) t(tcl→f) t(kcl→f) t(dcl→f) t(gcl→f) t(bcl→f) t(q→f)
- i(ih) t(ih→ih) t(ih→ix) t(ix→ih) i(ix) t(ix→ix)

In the last example, internal boundary models are grouped with transition boundary models. The reasoning is that a transition between a vowel and itself, if it ever occurs, is likely to be very similar to boundaries postulated by the segmenter within a segment of that vowel. Furthermore, since /ix/ is more or less simply a shorter realization of /ih/, any transition that is possible between /ih/ and itself can be grouped with those between /ih/ and /ix/.

Both strategies ensure that the set of boundary models created is such that in the test set every boundary that occurs is covered by some model. In the case of context-dependent segment models, complete coverage was not so important, since the scores of the context-dependent models were normalized by the context-independent scores in the re-scoring algorithm. In the case of boundary models, however, scores for each boundary are simply added to the total score for the path. If coverage were not 100%, paths with more boundaries covered by the model set would score better than paths with less frequent boundaries.

6.3.2 Training

The set of measurements used to represent a boundary is a set of eight averages of the first 14 MFCC's over successive speech windows 5, 10, 20, and 40 msec. long, symmetric about the boundary. PCA was used to normalize the feature space and reduce the dimensionality to 50 [7].

The training of boundary models follows exactly the same pattern as that for segment models, the only difference being that, in many cases, one model is trained

[1] It is not necessary to cover all 3600 possible transitions, because many of these are disallowed by the pronunciation network, and will never be considered during the search.

to represent many different types of boundaries. (The "catch-all" model, for example, corresponds to any of 1534 different classes of transition boundaries.) In these cases, the model is trained from the set of measurement vectors collected from all of the boundaries included in the model definition.

6.3.3 Testing

The purpose of the following experiments is the following:

- to compare the performance of the two strategies presented above for choosing the set of boundary models, and
- to compare the performance of the boundary models in the resorting pass with their performance in the Viterbi search.

One set of boundary models was trained according to each strategy and then tested both in the Viterbi search and in the resorting phase. As usual, in order to calculate the statistical significance of the results, multiple trials of each experiment were performed, and the final averages are accurate to within 0.25%.

Boundary models are incorporated into the Viterbi search simply by adding to the score of a segment the score for the boundary model joining the previous segment to it. Since the Viterbi search maintains only one partial path up to any given node in the Viterbi lattice, the identity of the previous segment, and thus the boundary between the two, is known unambiguously. Finally, the sum of the scores of the internal boundary models is also included in the score for a given segment.

Resorting according to boundary models is even simpler. Each path in the N-best list includes the sequence of boundaries traversed by that path, and the new score for a path is simply the original score plus the sum of the boundary scores along that path. The final score assigned to a path is the *same* as would have been assigned it had the boundary models been used directly in the Viterbi search.

6.3.4 Results

Table 6-1 summarizes the results from the experiments described above. The most important observation is that the addition of boundary models provides a phenomenal gain in performance over the use of segment models alone. A near fifty percent reduction in the word error rate is far more significant than any results achieved by the use of context-dependent segment models.

The second result of interest is that the performance of the boundary models is significantly better when they are included directly in the Viterbi search than when they are used to resort the N-best list. In fact, the best results achieved in this thesis have come from the use of boundary models directly in the Viterbi search.

Finally, the "catch-all" models (25+) outperform the set of models defined by Hazen, though the differences are not very significant. In future experiments, I have chosen to use the "catch-all" strategy for reasons of simplicity.

	Re-Score Results		Viterbi Results	
Model Set	$prune = 15$	$prune = 30$	$prune = 30$	$prune = 50$
Context-Independent	13.70 %	13.16 %	13.16 %	13.17 %
Catch-All (50+)	8.59 % (37.3 %)	7.43 % (43.5 %)	5.98 %	5.39 %
Catch-All (25+)	8.19 % (40.2 %)	7.03 % (48.7 %)	5.77 %	5.13 %
Hazen	8.52 % (37.8 %)	7.33 % (44.3 %)	5.95 %	5.04 %

Table 6-1: Summary of results from boundary model experiments. The numbers in parentheses are the percent reduction in word error rate achieved by the re-scoring over the results of the context-independent system. For comparison purposes, results are also presented for the 25+ version of the catch-all models, defined similarly to the 50+ models described above, except that only 25 examples are required to make a separate model.

6.4 Combining Boundary and Segment Models

The use of either boundary models or context-dependent segment models results in significant performance gains with respect to the context-independent system. Since the two types of models can be applied fairly independently, presumably a combination of the two would result in higher performance than either one can achieve alone. The experiments described in this section explore three possible combinations:

1. Use the boundary models in the Viterbi and A* searches and context-dependent segment models to re-score the N-best list.

2. Use context-dependent segment models in the Viterbi and A* search and boundary models to re-score the N-best list.

3. Use both boundary models and context-dependent segment models in the re-sorting pass.

The context-dependent segment models are the interpolated, word-dependent phone models that occur 25 times or more in the 109-speaker training set. The boundary models were chosen according to the 50+ "catch-all" strategy, as described in Section 6.3.1. In all three experiments, the combination of the models is achieved in a straight-forward manner, applying each sort of model as if it were independent of the other[2]. In the case where both models are used to re-score the N-best list, the contributions from the boundary models and the context-dependent segment models are simply added together.

Table 6-2 summarizes the results. The row identifies the models used in the first two recognition passes (Viterbi and A* searches), and the column identifies the models

[2]One could imagine other alternatives—for example, making models of the boundaries between word-dependent phonetic units—but this is not what was done.

Viterbi and A*	Type of Model(s) Used in Resort Pass			
	None	Boundary	Word-Dep	Both
Context-Independent	13.70 %	8.59 %	10.71 %	8.00 %
Word-Dependent	10.65 %	7.55 %	–	–
Boundary	5.93 %	–	5.47 %	–

Table 6-2: Word error rates resulting from the possible combinations of boundary models with word-dependent models.

used to resort the N-best list. In all cases the combination of the two types of models yields better performance than either type alone. As expected, the use of boundary models in the Viterbi and A* searches, followed by the use of word-dependent phones to resort the N-best list, provides the lowest word error rate of all, presumably because the boundary models when used in the Viterbi search provide such a good starting point for resorting. Unfortunately, in the course of these experiments, it was discovered that the use of boundary models in the A* search often causes the search to consume an inordinate amount of computation on some of the utterances, making the use of boundary models in the first pass impractical in these cases. The problem can probably be overcome with an appropriate implementation, but as it stands the most practical alternative for a real-time system is to combine the two types of models in the resorting pass.

6.5 Chapter Summary

In the experiments described in this chapter, acoustic models of the boundaries proposed by the recognizer (rather than the segments) were added to the baseline system, with impressive gains in performance. The most significant improvements came from the use of boundary models in the Viterbi search, although the reduction in error rate by their application in the resort pass far surpassed that achieved by the context-dependent segment models. In particular, the experiments evaluated two strategies for creating models to cover every possible transition, one in which boundaries with phonetically similar contexts were clustered together, and another in which a threshold was used to determine which boundaries to train, while the others were automatically merged into one class. Finally, some results were presented that demonstrate that the use of boundary models and context-dependent segment models together yields better performance than either strategy alone.

Chapter 7

Offset Models

7.1 Introduction

Up to this point, the context-dependent models considered in this thesis have been of the same form, in which the training data for each context-independent phonetic unit are divided into sub-classes based on the identity of the surrounding context. Each of these sub-classes is then trained (and applied during recognition) as if it were a phonetic unit unto itself. In effect, the standard algorithm reduces to using a larger set of phonetic units, relying on the apparent identity of the context as a fast way of ruling out most of them.

This paradigm is unsatisfying because, although it is motivated by the goal of capturing the co-articulatory effects of the surrounding context, it makes no attempt to explicitly model these effects. This deficiency is most evident in the word-dependent phone models, where all pretense at understanding the cause of the contextual variation is abandoned.

The strategy presented in this chapter, based on work by Phillips *et al* [22] here at MIT, is a tentative step toward building more advanced context-dependent models that learn how to compensate for co-articulatory effects by applying transformations to the measurement vectors themselves.

The basic idea behind "offset" modeling is that the effects of context can be approximated as a translation of the measurement vector, which can be applied in reverse during testing to "undo" the effects of context. The remainder of this chapter presents the mathematical framework for this technique, describes the procedure by which it is applied in SUMMIT, and presents experimental results that document its performance.

7.2 Mathematical Framework

As before, since each phonetic unit is treated independently, the following descriptions are for the hypothetical phonetic unit α, with the understanding that all other phonetic units are treated equally.

The "ideal" measurement vector $\mathbf{x}_\alpha$ is calculated as an average over the measurement vectors $\mathcal{X}_\alpha$ that apply to the phonetic unit α:

$$\mathbf{x}_\alpha = \frac{1}{\mid \mathcal{X}_\alpha \mid} \sum_{\mathbf{x} \in \mathcal{X}_\alpha} \mathbf{x} \tag{7.1}$$

Similarly, if $\mathcal{C}$ is the set of contexts that can apply to a phonetic unit, then the set $\mathcal{X}_\alpha$ can be partitioned into non-overlapping subsets as follows:

$$\mathcal{X}_\alpha = \bigcup_{c \in \mathcal{C}} \mathcal{X}_{\alpha,c}$$

where $\mathcal{X}_{\alpha,c}$ is the set of measurement vectors that apply to α in the context c. Then, for each context, the ideal measurement vector *for that context* can be calculated just as in Equation 7.1.

$$\mathbf{x}_{\alpha,c} = \frac{1}{\mid \mathcal{X}_{\alpha,c} \mid} \sum_{\mathbf{x} \in \mathcal{X}_{\alpha,c}} \mathbf{x}$$

Given the ideal measurement vector both for the phonetic unit as a whole and for each context, the "effect" of each context c, represented by the translation vector $\mathbf{t}_{\alpha,c}$, can be estimated as the difference between the two ideal measurement vectors:

$$\mathbf{t}_{\alpha,c} = \mathbf{x}_\alpha - \mathbf{x}_{\alpha,c}$$

If the context of a measurement vector $\mathbf{x}$ is known to be c, then one can compensate for the effects of context by applying the translation in reverse, as follows:

$$\mathbf{x}' = \mathbf{x} + \mathbf{t}_{\alpha,c}$$

Applying this transformation to the measurement vectors in the training set $\mathcal{X}_\alpha$ yields a new training set $\mathcal{X}'_\alpha$ of measurement vectors that have been adjusted to "compensate" for contextual effects. From this new training set, the new acoustic model $p'(\mathbf{x} \mid \alpha)$ can be trained as described in Section 2.2.2. To apply this model during testing to a measurement vector $\mathbf{x}$ in context c, the vector must first be adjusted by adding to it the appropriate translation vector.

7.3 Application in SUMMIT

The application of offset models requires two separate components: 1) the offset vectors for each context-dependent unit, and 2) the context-independent phonetic models that have been trained from translated measurement vectors. The procedure for creating these two components is summarized in the following steps:

1. Calculate the "ideal" measurement vector for each context-independent phonetic unit as the average of the measurement vectors labeled with that unit.

2. Similarly, calculate the "ideal" measurement vector for each context as the average of the measurement vectors in the training data to which that context applies.

3. Calculate a translation vector for each context as the difference between the ideal vector of the phonetic unit and the ideal vector of the context itself.

4. Adjust each measurement vector in the training data by adding to it the translation vector corresponding to its context. This adjustment is intended to "un-do" the co-articulatory influences of the surrounding context.

5. Train a new set of context-independent models from the training set of adjusted measurement vectors.

The following procedure is followed when *applying* offset models during recognition to score a segment:

1. Determine the context of the segment and add the translation vector for that context to the measurement vector for the segment.

2. Score the segment with the appropriate adjusted context-independent model.

Just as in the case of normal context-dependent models, the application of offset models takes place in the resorting stage, so that the recognizer can take advantage of the knowledge of the surrounding context.

7.4 Experimental Results

7.4.1 Basic Experiments

The contexts that can be defined for offset modeling are the same as those that can be used by traditional techniques, including word-dependent, biphone, and triphone contexts. One advantage of offset models, however, is that for each context only a single vector must be trained, without estimating the shape of a probability distribution. Another is that the memory required to store offset vectors is significantly less than that required to store a similar number of acoustic models. For these reasons, the threshold for the number of tokens required to train an offset vector can be much lower than that required to successfully train acoustic models.

The experiments summarized in Table 7-1 follow the procedure described above, with different ways of defining the context and different thresholds for the number of tokens required to train an offset vector.

7.4.2 Modeling Unseen Triphones

One interesting possibility that arises with the use of offset models is that independent contextual effects can be modeled as a sum of the translations due to each context.

Type of Model	Min # Tokens	Coverage	Word Error Rate
Word-Dependent	50	51.2 %	12.35 %
Word-Dependent	10	85.2 %	11.84 %
Right-Biphone	50	92.9 %	11.87 %
Right-Biphone	10	99.1 %	11.04 %
Left-Biphone	10	99.1 %	12.26 %
Triphone	2	97.5 %	11.24 %

Table 7-1: Summary of the performance of several variations of the offset model strategy.

For example, one might estimate the contextual effects of a triphone context to be the sum of the effects of the left context and the right context. By doing so, one can *predict* the contextual effects of unseen triphones, as long as translation vectors for both the left and the right contexts have been calculated.

In the following experiment, the translation vectors for the left and right biphones with a minimum of 10 examples in the training data were added together in all possible combinations to produce translation vectors for the resulting triphones. These vectors were then used to adjust the training data, as per the normal procedure, in order to create new context-independent models. These models were evaluated in the resorting pass, just as in the first experiment. Table 7-2 provides a summary of the results.

Type of Model	# Models	Coverage	Word Error Rate
Left-Biphone (10+)	1105	99.1 %	12.26 %
Right-Biphone (10+)	1105	99.1 %	11.04 %
Normal Triphone (2+)	7127	97.5 %	11.24 %
Triphone Combination	22767	98.1 %	12.23 %

Table 7-2: The performance of triphone models, both in the normal case and as a combination of left and right biphone models.

Unfortunately, the simple combination of left and right contexts yielded the performance of the worse of the two. In most cases, it is probably the case that either the right or the left context dominates the co-articulatory effects, and that it is inappropriate to imagine that both apply independently to any one token.

7.4.3 Context-Dependent Offset Models

Finally, instead of measuring the expected difference between context-dependent and context-independent phonetic units, one could measure a similar difference between two types of context-dependent units, where one of the types is more general than the other. In the following experiments, two different sets of offset vectors were trained. The first set contained offset vectors between triphone and context-independent phonetic units, and the second between triphone and right-biphone contexts. These offset vectors were applied separately to two identical sets of training data, and from these two sets of data were trained two sets of right-biphone models. During recognition, the application of such models follows the procedure described above, whereby the offset vector is applied depending on the triphone context and the appropriate right-biphone model (trained from data adjusted by the same offset vectors) is applied. Unfortunately, none of these more complicated strategies yields performance that is superior to the normal procedure of training offset vectors.

Table 7-3 provides a summary of the results.

Type of Offset	**Type of Model**	**Word Error Rate**
Triphone$\Longrightarrow$CI	CI	11.24 %
Triphone$\Longrightarrow$CI	Right-Biphone	11.61 %
Triphone$\Longrightarrow$Right-Biphone	Right-Biphone	11.62 %

Table 7-3: Performance of right biphone models when tested on data adjusted according to offset vectors. The first row is the normal case, where offsets between triphone contexts and context-independent units are used to train adjusted context-independent models, which are applied in the resorting pass as usual. The second row uses the same offset vectors, but instead trains right-biphone models from the adjusted training data, applying these right biphones in the resorting pass. Finally, the third row trains offset vectors between triphone and right biphone contexts, applies these offsets to the training data, from which are trained right biphone models. These offsets and their corresponding right biphone models are applied in the resorting pass as per the usual procedure.

7.5 Chapter Summary

The offset strategy makes several assumptions, all of which are invalid, in order to make the formulation mathematically tractable.

1. That each phonetic unit has an “ideal” realization that it would adhere to in the absence of contextual effects (one possible interpretation of the phonemic theory of speech). Furthermore, that this ideal realization can be estimated as the average over all examples in the training set.

2. That the co-articulatory influences of neighboring phonetic units can be represented as a transformation applied to measurement vectors. Futhermore, that this transformation takes the form of an additive offset to the measurement vector in question.

3. Finally, that this offset depends only on the *identities* of the surrounding phonetic units, and not on the values of any measurement vectors.

Because of these simplifications, the potential for success as a result of the application of such models is limited. Furthermore, there is no technique analagous to deleted interpolation to increase the robustness of the individual offset estimations. The most promising result of the offset model investigations is that, despite their simplicity, they were able to reduce the error rate of the context-independent system by almost twenty percent, demonstrating that the goal of reversing the effects of context is tangible, and that perhaps more sophisticated techniques might enjoy substantial success.

Chapter 8

Conclusions and Future Work

8.1 Thesis Overview

This thesis explored the use of context-dependent models to re-score and re-rank the hypotheses of a context-independent recognizer. Part of the motivation behind such a strategy, as opposed to incorporating context-dependent models directly into the Viterbi search, was to reduce computational costs by constraining the search space over which the context-dependent models are required to operate. Further incentive came from experimental results which showed that the correct answer, though not the recognizer's top choice, was in fact among the alternative hypotheses. In the case of Resource Management, for instance, the correct answer is within the top 100 hypotheses for 88.3 % of the test utterances, but occurs as the top choice in only 45.7% of them.

Several standard types of context-dependent models were discussed, including word-dependent phone models, left and right biphone models, and triphone models. Instead of combining poorly trained models together to obtain complete coverage, a simple count-threshold was used to select the contexts to be covered, ignoring those with insufficient training data. Deleted interpolation was used to combine the context-dependent models with the more robust context-independent models. This technique was shown, in general to improve the performance of the system.

Of the context-dependent models, the most successful were the word-dependent models and the right biphone models, both of which were able to reduce the word error rate of the context-independent system by over twenty percent. More surprising, however, was the performance of boundary models, which in the resort pass were able to reduce the word error rate by more than forty percent. When included directly in the Viterbi search, boundary models were able to achieve a reduction in error rate over the baseline system of more than 60 percent, although at a significant cost in computation.

8.2 Future Work

The most impressive results that the SUMMIT recognizer has been able to obtain on the Resource Management task come from the use of boundary models, along with context-independent models, in the Viterbi and A* searches. At present, the computational cost of doing so is not exhorbitant for most utterances, but on a small fraction of utterances the computation required to recognize the utterance is exhorbitant, for reasons that are not well understood. The computational cost for the majority of utterances, as well as the few unruly ones, can probably be reduced significantly by changes to the implementation, which at the moment is fairly memory intensive. In addition to simple data structure modifications, more intelligent algorithms that detect cases where the search space becomes intractable would be very helpful.

More sophisticated techniques for combining together several evaluations of a recognition hypothesis are needed. Fisher's linear discriminant provides a mechanism for determining the significance to attach to each evaluation, but more sophisticated classifiers might be more useful. Especially interesting would be methods that examine only regions of speech where two hypotheses *differ* in order to determine which of the two is superior, rather than evaluating each hypothesis independently from all the others.

Finally, the right approach to context-dependent modeling probably has more to do with offset models than with traditional techniques. If phonetic theory is to be accepted, whereby speech is a concatenation of basic units which, unfortunately, are distorted by their own phonetic context, then the obvious approach is to somehow un-do the effects of this distortion and go back to the use of context-independent models. The offset model is a first attempt to do something of this sort, but the assumption that the distortion takes the form of a linear offset really hampers its potential for success (though surprisingly it actually performs quite well). Furthermore, trying to predict the distortion caused by phonetic context from only the *identity* of the surrounding phones is bound to be extremely difficult, especially in light of the fact that the phonetic units being used are extremely broad. The best approach, I think, would be to try to train a function that takes the *measurement vectors* of the surrounding phones as input and produces as output the measurement vector that would have been pronounced were there no co-articulatory effects present.

Appendix A

Segment Measurements

The measurement vectors calculated for every segment consist of 40 components, 39 of which are averages or derivatives of MFCC values over portions of the segment. The last component is the log of the duration of the segment. These measurements are the forty dimensional set described by Manish Muzumdar in his Master's thesis [20]. These features were found to provide a compact vector with good phonetic classification performance. The following table details the exact form of the measurements:

Components	**Operation**	**Portion of Segment**	**MFCC Values**
1–8	average	first 30%	0–7
9–22	average	middle 60%	0–13
23–28	average	last 30%	0–5
29–31	derivative	first frame	0–2
32–39	derivative	last frame	0–7
40	log duration	entire segment	X

Table A-1: Definition of the 40 measurements taken for each segment in the experiments described in this thesis.

The first eight components of the measurement vector are the averages, taken over the frames in the first 30% of the segment, of the values of the first eight MFCC's (coefficients zero through seven). Similarly for the next twenty components. Components 29 through 39 are derivatives calculated from a 40 msec. window centered either at the beginning or the end of the segment. Thus, the measurements for a given segment actually extend to frames in adjacent segments. Finally, the 40th component is simply the log of the duration of the segment.

Appendix B

Language Modeling in the Resource Management Task

B.1 Introduction

The language model defined by the Resource Management task is a word pair grammar (WPG) which specifies, for each word in the vocabulary, the set of words that can immediately follow it. Unlike an n-gram model, a word pair grammar is not probabilistic; rather, it simply accepts or rejects sentences according to whether or not they obey the word pair constraints. This lack of probabilistic rigor is problematic for speech recognition systems that are grounded in a statistical framework. This chapter explores some of the issues concerning the word pair grammar of the Resource Management task and its interpretation as a probabilistic model.

B.2 Perplexity

Perplexity is commonly used to measure the complexity of language models. In order to compare the results of different speech recognition systems, it is often helpful to know the perplexity of the task. In general, tasks with higher perplexity will result in lower recognition performance on any given system.

B.2.1 Test-Set Perplexity

A statistical language model defines a probability $Q(w_1 \ldots w_n)$ for all word sequences $w_1 \ldots w_n$. The *test-set perplexity* is defined for a given word sequence $w_1 \ldots w_n$ and a given language model Q as follows [25]:

$$L = 2^K,$$

where K, the average per word log probability, is given by

$$K = -\frac{1}{n} \lg\left[Q(w_1 \ldots w_n)\right].$$

In the case of a bigram language model, the equation for test-set perplexity can be re-written as

$$K = -\frac{1}{n}\sum_{i=1}^{n} lg[Q(w_i \mid w_{i-1})].$$

Note that the definition of test-set perplexity requires both a language model $Q(\cdot)$ and a test set $w_1 \ldots w_n$. To speak of the test-set perplexity of a language model alone is misleading. When presenting results, the test-set perplexity and the recognition performance should be evaluated on the same set of utterances.

B.2.2 Language Model Perplexity

If the test set $w_1 \ldots w_n$ is obtained from a well behaved source with probability $P(w_1 \ldots w_n)$, then the average per word logprob converges to its expected value with large n:

$$\lim_{n\to\infty} -\frac{1}{n}\sum_{w_1 \ldots w_n} P(w_i \ldots w_n) lg[Q(w_i \ldots w_n)] = \lim_{n\to\infty} -\frac{1}{n} lg[Q(w_i \ldots w_n)].$$

If the language model truly characterizes the source (that is, $P = Q$), then the expected value of the test-set perplexity is given by 2^H, where H is the entropy of the source language itself [25]:

$$H = \lim_{n\to\infty} -\frac{1}{n}\sum_{w_1 \ldots w_n} P(w_i \ldots w_n) lg[P(w_i \ldots w_n)].$$

The value 2^H is called the *language model perplexity*, which differs from the test-set perplexity in that it is a characteristic of the language model itself, regardless of any given test-set. The two measures are related as follows: if the language model $Q(\cdot)$ were used to generate sentences, the test-set perplexity of those sentences would converge to the language model perplexity as the number of sentences increases.

Normally, the language model used by the recognizer does not match the source model generating the sentences (*i.e.* $Q \neq P$). In this case, the expected test-set perplexity will be larger than the language model perplexity 2^H for any n [25].

For a bigram language model, the language model perplexity can be calculated directly from the model as follows [31]:

$$H = -\sum_{w_1, w_2} Q(w_1, w_2) lg[Q(w_2 \mid w_2)].$$

B.3 The Word Pair Grammar

A word pair grammar can be implemented by a finite state automaton, where each state in the automaton corresponds to a word in the grammar, and where a transition is possible between two states if and only if the second state corresponds to one of the words that can follow (according to the grammar) the word corresponding to the first state. Note that word pair grammars are a *subset* of the class of regular languages (the set of languages that can be accepted by finite state automata).

In order to characterize the word pair grammar in terms of perplexity, it must first be converted to a probabilistic language model. Since the grammar itself provides constraints based only on the current word, it seems reasonable to make the Markov assumption that the conditional probability of a given word in a sequence depends only upon the previous word. Furthermore, in the absence of additional information, it is assumed that all words that can legally follow another word are equally likely to do so.

Making these assumptions transforms the finite state automaton described above into a first order Markov process, where for any given state with M transitions, the probability for each transition leaving that state is $\frac{1}{M}$. Instead of simply accepting or rejecting input sentences, the model now assigns a non-zero probability to accepted sentences. This probability is calculated as the product of the transition probabilities along the state sequence taken by the model in accepting that sentence.

These transition probabilities are the conditional probabilities of a bigram language model, and are sufficient to calculate the test-set perplexity of the model or to apply the model during recognition. To calculate the language model perplexity, however, we need the unigram probabilities of the words as well. Assuming that the Markov process is ergodic and contains no periodic states (almost certainly the case for a realistic grammar), then the unigram probability of each word is the limiting-state probability of the state corresponding to that word in the Markov model. The limiting state probabilities obey the following relationships [4]:

$$P_j = \sum_{i=1}^{N} (P_i \cdot P_{ij}) \qquad\qquad \sum_{i=1}^{N} P_i = 1.$$

where N is the number of states in the model, P_j is the limiting state probability for state j, and P_{ij} is the probability of a transition to state j given state i.

In words, the probability of being in state j is equal to the sum (over all possible states i) of the probability of being in state i and then transitioning to state j. The sum of all limiting-state probabilities P_i is one (by the definition of probability).

The limiting state probabilities of a Markov model can be computed by first initializing the values arbitrarily, and then repeatedly re-calculating the values according to the formula given above until they converge to a fixed-point. (After each step the values must be normalized so that they obey the constraint that they sum to one.) Figure B-1 is a more precise description of the algorithm:

B.4 Measuring Perplexity

Three different bigrams were created to simulate the word pair language model. Bigram-A assigns a "probability" $P(w_j \mid w_i) = 1$ for each word pair $< w_i, w_j >$

```
for i ← 1 to N
    P_i = 1/N
repeat
    for j ← 1 to N
        P'_j ← 0
        for i ← 1 to N
            P'_j ← P'_j + (P_i · P_ij)
    sum ← 0
    for j ← 1 to N
        sum ← sum + P'_j
    change ← 0
    for j ← 1 to N
        P'_j = P'_j/sum
        change ← change + (P'_j − P_j)^2
        P_j = P'_j
until change < ε
```

Figure B-1: The algorithm for computing the limiting state probabilities of a Markov model.

in the grammar and $P(w_j \mid w_i) = 0$ for each word pair $< w_i, w_j >$ that does not occur in the grammar. This "bigram" is not probabilistic, since the conditional probabilities are not normalized to sum to one, but it simulates the word pair grammar during recognition by assigning an equal score to all legal competing word sequences while still excluding illegal sequences.

Bigram-B assumes (falsely) that every word has equal unigram probability $P_j = \frac{1}{N}$ and that, given a word, all M words that can legally follow are equally likely with probability $P_{ij} = \frac{1}{M}$. Bigram-C is identical to Bigram-B except that the unigram probabilities are calculated by solving for the limiting state probabilities as described in section B.3.

For each of the three bigrams, both the language model perplexity and the test-set perplexity were calculated. (For Bigram-A and the word pair grammar these values are nonsensical since the models are not probabilistic.) In addition, the baseline recognition system was tested on the 1989 test set (*test89*) using each bigram. The results are presented in Table B-1.

Bigram-B and Bigram-C yield the same results because they differ only in the unigram probabilities, which do not factor into the calculation of test-set perplexity, nor into the language model score computed during recognition. The differences in performance between the word-pair grammar and Bigram-A are the results of differences in the way the recognizer chooses the best alternative in the case of a tie, which is more likely to happen in these cases due to the fact that these grammars

Language Model	Language Model Perplexity	Test-Set Perplexity	Word Error Rate
Word Pair	–	–	84.2 %
Bigram-A	–	–	84.3 %
Bigram-B	16.637	62.257	86.3 %
Bigram-C	41.548	62.257	86.3 %

Table B-1: A comparison of three interpretations of the word pair grammar.

assign an equal score to all sentences. The following is an example of such a case:

```
Reference Sentence:    Are there four submarines?
Recognizer Hypothesis: Are there for submarines?
```

B.5 Conclusion

Part of the motivation for exploring the issues regarding the perplexity of the word pair grammar arose from the fact that in the Resource Management literature the language model is consistently referred to as the "perplexity 60 word pair grammar," despite the contradiction inherent in speaking of the perplexity of a non-probabilistic model. Moreover, as the results displayed in Table B-1 indicate, the language model itself cannot be said to have perplexity 60 (in fact, 41.5 seems most reasonable), nor can test-set perplexity values be given without reference to specific data. Finally, it is not clear from published results whether recognizers were evaluated using a language model that enforces the word pair constraint directly or using a bigram (such as Bigram-C) that interprets the word pair grammar in a probabilistic framework. The performance of the two strategies differ, and for the 1989 DARPA evaluation, my baseline results suggest that the latter (using a bigram) provides better performance. For the sake of maintaining a probabilistic framework, Bigram-C has been used as the default language model in the experiments described in this thesis.

Appendix C

Nondeterminism in the SUMMIT Recognizer

C.1 Introduction

The SUMMIT recognizer is nondeterministic[1] due to the clustering procedure used to create the mixture Gaussian acoustic models. This chapter explains the basis for this nondeterminism, explores the theoretical consequences of it, and presents some experimental results that demonstrate the sort of variation in performance exhibited by the system. Finally, a solid probabilistic framework is developed for interpreting the results presented throughout the thesis.

C.2 Mixture Gaussian Models

The acoustic models are continuous probability density functions, one per phonetic unit, that approximate the probability of an observed feature vector, given the identity of the phonetic unit. Although in theory the density functions may be of any form, those used in this thesis work are mixtures of several component densities, each of which is a diagonal Gaussian probability density function. For a given phonetic unit α, a mixture of M components approximates the probability of a feature vector $\mathbf{x}$ by

$$p(\mathbf{x} \mid \alpha) = \sum_{i=1}^{M} w_i p_i(\mathbf{x} \mid \alpha),$$

The training procedure first clusters the training examples of α into M clusters, uses each cluster to estimate the parameters of each component, and then uses maximum likelihood estimation to successively refine the initial estimates of the parameters. In the case of a mixture of diagonal Gaussian models, for example, the

[1] That is, the system *appears* to be nondeterministic to the user, who observes varying results despite running identically configured experiments on the same data. In reality, the system is deterministic, but its behavior is dependent on inputs that the user can not control nor predict, thus providing the illusion of randomness.

parameter vector for a given component is simply the mean and the variance of the measurement vectors belonging to the cluster associated with that component. The mixture weight ω_i for a given component is estimated as the fraction of examples in the entire model that belong to the i^{th} cluster.

C.3 Clustering

The space of possible clustering procedures is overwhelming [5]. Choosing one requires making a series of difficult decisions, from the distance metric to be used to the criterion function that decides what constitutes a cluster. The system described in this thesis uses a Euclidean distance metric and an agglomerative (bottom-up) clustering procedure known as *k-means clustering*. The algorithm divides the data into M clusters, $C_1, \ldots, C_M$, as follows [31]:

1. Select M data samples to serve as the initial cluster means.
2. Assign each data sample to the cluster with the closest mean.
3. Recompute the mean for each of the M clusters.
4. Repeat steps 2 and 3 until the change in distortion (average distance of the data samples to their associated cluster means) falls below some threshold.

The nondeterminism in the system arises from Step 1, where the initial M cluster means are chosen randomly from the set of available data samples. A difference in initial values leads to a difference in the final assignment of data samples to clusters, which in turn leads to a difference in the shape of the resulting acoustic model and a variation in the performance of the system during testing.

C.4 Handling Variability

As explained above, different sets of models can differ in performance, even when the models are configured identically and trained on the same data. This variation is not such a problem in and of itself; more problematic is the difficulty that arises when comparing the performance of models trained on the same data but configured differently. For example, suppose I change the feature set and want to measure the effect on performance. I train a new set of models using the new feature set and measure its performance, only to find that the resulting word accuracy is half a percent worse than before. How can I know whether this difference is due to the change in the feature set or to the random variation that I can expect whenever I train a set of models?

Test Set	No. Utterances	Mean	Variance	Range
dev	1200	87.31 %	0.12	1.6 %
test89	300	85.86 %	0.31	2.6 %

Table C-1: Statistics of the variation encountered when 50 model sets, each trained under the same conditions on the same data, are tested on two different test sets.

C.4.1 Experimental Observations

To answer the above question, the first step is to find out what kind of variability we can expect from the system under identical conditions. An experiment was performed whereby 50 different sets of models were created, each trained on the same data under identical conditions, and then evaluated on two different test sets. The results are summarized in Table C-1. Most striking is the range of performance possible, more than two and a half percent over a set of 300 utterances. A difference of 1.5 percent corresponds to a ten percent reduction in the error rate, which normally would be considered quite a significant result. Clearly, when comparing different systems, performance differences of less than a few percent will have to be evaluated more carefully.

A second notable result is the difference in range between the evaluations on the dev set and those on test89. The dev set, which contains four times as many utterances as test89, has only 40 percent the variance. Similarly, we might expect that a larger training set would also narrow the range in performance.

C.4.2 Error Estimation

Looking at Table C-1, we can be reasonably confident that test89 is "harder" than the dev set, at least for this configuration of the recognizer. Our confidence stems from the fact that we ran many experiments, and we expect that if we ran many more, we would still find that on average our models perform worse on test89 than on the dev set. But how confident are we? In general, how many experiments do we have to run to be able to state with confidence that we believe the results?

The answer derives from basic probability theory, though taking advantage of the theory requires making some assumptions. The basic idea is to consider the set of experimental results $y_1, \ldots, y_n$ to be a set of independent, identically distributed random variables. The sample mean M_n of these random variables is related to the distribution for y as follows [4]:

$$M_n = \frac{y_1 + y_2 + \ldots + y_n}{n}$$

$$E(M_n) = \frac{nE(y)}{n} = E(y)$$

$$\sigma^2_{M_n} = \frac{n\sigma^2_y}{n^2} = \frac{\sigma^2_y}{n}$$

Applying the Chebyshev inequality to M_n yields the following weak upper bound on the probability that a measured value of M_n will differ from the true expected value of y by more than some offset ϵ [4]:

$$Prob[\mid M_n - E(M_n) \mid \geq \epsilon] \leq (\frac{\sigma^2_y}{n\epsilon^2})^2$$

Unfortunately, this bound is rather weak, but it can be tightened significantly by assuming that the random variables obey a normal distribution[2]. If so, then knowing the form of the PDF for M_n, we can easily calculate the probability of error using the following formula:

$$Prob[\mid M_n - E(M_n) \mid \geq \epsilon] = 1 - 2\Phi(\frac{-\epsilon\sqrt{n}}{\sigma_y}),$$

where $\Phi()$ is the unit normal PDF, which is evaluated by consulting a table. The above formula can be rearranged to solve for any one of the many parameters, assuming that the others are fixed. For example, for the 50 experiments run on test89, if we fix our level of confidence at 0.95, using the value of σ_y calculated from the data, we get a value of $\epsilon = 0.15$. In other words, the probability that the expected value of the accuracy of a set of models on test89 is within 0.15% of the sample mean 85.86% is 0.95. That is, we can be 95% sure that the true mean (*i.e.*, the value we would find if we ran an infinite number of experiments) is between 85.71% and 86.01%.

C.5 Cross Correlations

One might be tempted, from the above results, to train many sets of models, evaluate them all on the dev set, and then use only the best set of models for testing on the test set. Indeed, at this point we don't know whether this strategy is likely to work or not. The answer depends on whether the variation in the models is a reflection on their fundamental "goodness" or simply a measure of their coincidental alignment with a given test set. This question can be resolved by measuring the *correlation coefficient* (or normalized covariance) between the values recorded for each of the two test sets. The correlation coefficient for two random variables x and y is defined as follows:

[2]This assumption is clearly false in this case, since the number of choices for randomly seeding the clustering algorithm is finite. However, according to the central limit theorem, as $n \rightarrow \infty$ the PDF for the sample mean will approach the Gaussian PDF, regardless of the form of the PDF of the random variables, so the assumption may not be so bad after all (see Figure C-1).

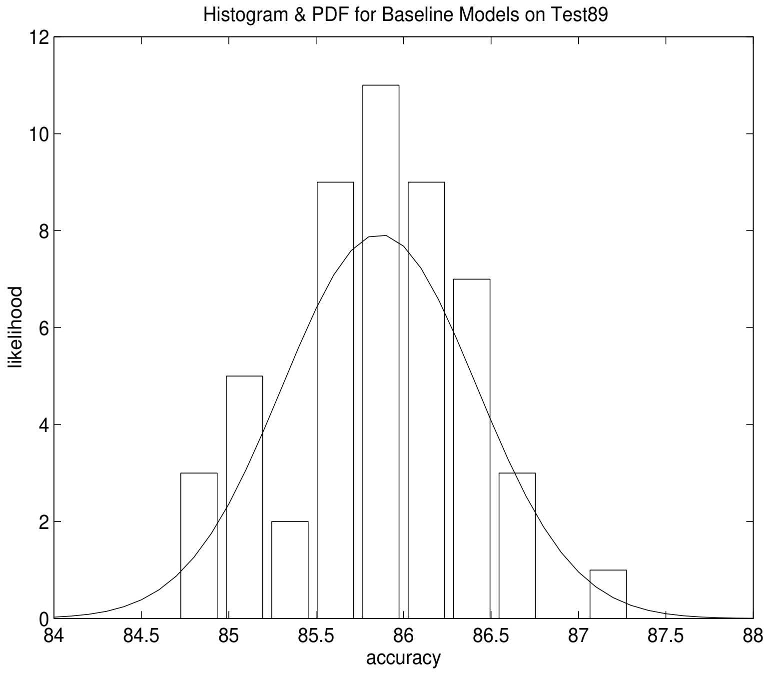

Figure C-1: A histogram of the results of 50 different sets of models evaluated on test89, as described in Table C-1. Overlayed is a Gaussian distribution with the sample mean and sample variance as its parameters.

$$\rho_{xy} = E[(\frac{x - E(x)}{\sigma_x})(\frac{y - E(y)}{\sigma_y})]$$

A correlation coefficient of zero implies that the two random variables are completely independent, whereas a coefficient of one implies that they are completely dependent. Evaluating the results of the above experiment yields a correlation coefficient of 0.217. This result is somewhat ambiguous, showing that there is some correlation, but not much. In general, it can not be assumed that unusually good performance (due to probabilistic fluctuations) on one set of data will lead to particularly good performance on another set.

C.6 Conclusions

This nondeterminism in the recognizer seems to be a real hassle: why not design the system to exclude the possibility? This could surely be done, either by choosing a different clustering algorithm, or even by simply modifying the existing one to always pick a predefined set of starting points. Unfortunately, this sort of solution would not solve the problem, because although the results would be deterministic, there would be no guarantee that they are particularly good results.

The real problem lies not in the randomized k-means seeding that introduces nondeterminism, but rather the inherent randomness in the choice of training and test data. The kind of variability observed by training and testing multiple models under identical conditions is the same kind one would observe if one trained and tested identical sets of models on different subsets of the data. On average, "better" models will perform better, regardless of the division between training and test data. Unfortunately, any one particular division might lead to results where one set of models performs better than another, despite being in fact a "worse" set of models.

But this problem is also a blessing. By forcing us to recognize the inherent nondeterminism in the speech recognition system, we are forced into running multiple experiments in order to minimize the uncertainty in the measurements. If we did not do so, we might, for example, reject an incremental improvement because it happened, on one particular trial, to generate worse results. Or, more likely, we might incorrectly view a large (1-2 percent) improvement in results as a sure-fire indication that the models have improved, when in fact they might not have changed at all. By requiring us to explicitly measure the uncertainty in the results, the nondeterministic nature of the recognizer has in fact strengthened the scientific legitimacy of our work.

Appendix D

Sapphire Configuration Files

The following is an example of a Sapphire configuration file used in this thesis. It demonstrates the power of the Sapphire framework by using control-flow statements in Tcl to configure the recognizer according to the availability of input arguments. For example, if no boundary models are included in the arguments, then they are not used by the Viterbi search; otherwise, they are. Thus, the recognizer can configure itself appropriately by infering the intentions of the user by the arguments provided.

```
# Recognized variables are:
#
#   -bpcs        (boundary pcs file)
#   -spcs        (segment pcs file)
#   -bmodels     (boundary models file)
#   -smodels     (segment models file)
#   -sweights    (segment weights file)
#   -blabels     (boundary labels file)
#   -slabels     (segment labels file)
#   -cd_smodels  (CD segment models file)
#   -cd_slabels  (CD segment labels file)
#   -lex         (pronunciation network)
#   -bigram      (viterbi bigram file)
#   -as_ngram    (astar ngram file)
#   -vprune      (viterbi pruning threshold)
#   -aprune      (A* pruning threshold)
#   -nbest       (N for the N-best list)
#   -type        (type of CD modeling to perform)

# set defaults

set_if_unset bpcs        None
set_if_unset spcs        None
set_if_unset bmodels     None
```

```
set_if_unset smodels     None
set_if_unset blabels     None
set_if_unset slabels     None
set_if_unset sweights    None
set_if_unset cd_smodels  None
set_if_unset cd_slabels  None
set_if_unset lex         None
set_if_unset bigram      None
set_if_unset as_ngram    None
set_if_unset nbest       None
set_if_unset vprune      None
set_if_unset aprune      None
set_if_unset type        None

set_if_unset server_client_order 0

if {[info command w] == ""} {
  s_waveform w
}

s_transform_waveform tw \
    -waveform w \
    -normalize yes \
    -removedc yes \
    -preemphasis 0.97

s_stft stft \
    -waveform tw \
    -framedurationms 5.0 \
    -preemphasis 0.0 \
    -dft 256

s_mfsc_from_spectrum mfsc \
    -spectrum stft

s_rotate_spectrum mfcc \
    -spectrum mfsc \
    -type cepstrum \
    -numout 14

s_segment_from_mfcc segs \
    -mfcc mfcc

s_spectrum_mean_norm mfcc \
    -spectrum unnorm_mfcc \
```

```
    -steptime 0.5

# Segment measurements
source "40.srec"

# Diagonalized measurements
s_pcs rot_meas \
    -parent $seg_meas \
    -pcsfile $spcs

# Segment Classifiers
s_classifier seg_scores \
    -type "mixture_diagonal_gaussian" \
    -parent rot_meas \
    -labelsfile $slabels \
    -modelsfile $smodels
    -random_seed $random_seed \
    -weightsfile $sweights

if {$cd_slabels != "None"} {

  puts "Computing CD measurements for $cd_smodels"

  s_classifier cd_seg_scores \
      -type "mixture_diagonal_gaussian" \
      -labelsfile $cd_slabels \
      -parent rot_meas \
      -modelsfile $cd_smodels \
}

# Boundary Classifiers
if {$blabels != "None"} {

  puts "Computing boundary measurements for $blabels"
  source "112.brec"

  s_pcs rot_bound_meas \
      -parent $bound_meas \
      -trpooled yes \
      -trcorrelation yes \
      -labelsfile $blabels \
      -pcsfile $bpcs \
      -numout 50

  s_classifier bound_scores \
```

```
      -type "mixture_diagonal_gaussian" \
      -labelsfile $blabels \
      -parent rot_bound_meas \
      -expscale 0.5 \
      -mindatapoints $bnumdims \
      -maxmix 30 \
      -modelsfile $bmodels
}

if {$vprune != "None"} {

  if {$bmodels != "None"} {

    puts "Running Viterbi with boundary models"

    s_viterbi viterbi \
        -bounds segs \
        -segs segs \
        -boundstype bounds \
        -segstype segs \
        -segmentscores seg_scores \
        -boundaryscores bound_scores \
-ngramdef 0.0 \
        -printpaths 1 \
        -backtrace yes \
        -disableins yes \
        -disabledel yes \
        -prune $vprune \
        -bcdfile $lex \
        -ngramfile $bigram

  } else {

    puts "Running Viterbi without boundary models"

    s_viterbi viterbi \
        -bounds segs \
        -segs segs \
        -boundstype bounds \
        -segstype segs \
        -segmentscores seg_scores \
        -ngramdef 0.02 \
        -printpaths 1 \
        -backtrace no \
        -sortnodes yes \
```

```
        -disableins yes \
        -disabledel yes \
        -prune $vprune \
        -bcdfile $lex \
        -ngramfile $bigram
  }

} else {

  puts "No vprune specified: assuming bare-bones Viterbi for training"

  if {$blabels != "None"} {

    puts "Training boundary models"

    s_viterbi viterbi \
        -boundaryscores bound_scores

  } else {

    puts "Training seg models"

    s_viterbi viterbi \
        -segmentscores seg_scores
  }
}

if {$nbest != "None"} {

  puts "Performing A* search"

  s_astar astar \
      -parent viterbi \
      -thresh $aprune \
      -ngramfile $as_ngram \
      -nbest $nbest
}

if {$type != "None"} {

  puts "Resorting using type $type"

  if {$bmodels != "None"} {

    puts "resorting with boundary models $bmodels"
```

```
    if {$cd_smodels != "None"} {

      puts "Resorting with CD models $cd_smodels"

      # resort with both CD models and boundary models
      s_resort resort \
          -astar_parent astar \
          -ci_parent seg_scores \
          -cd_parent cd_seg_scores \
          -bound_parent bound_scores \
          -type $type

    } else {

      # resort with only the boundary models
      s_resort resort \
          -astar_parent astar \
          -ci_parent seg_scores \
          -bound_parent bound_scores \
          -type $type
    }

  } else {       # $bmodels == None

    if {$cd_smodels != "None"} {

      puts "Resorting with CD models $cd_smodels"

      s_resort resort \
          -astar_parent astar \
          -ci_parent seg_scores \
          -cd_parent cd_seg_scores \
          -type $type

    } else {

      puts "Resort type $type specified, but no CD or bound models -- not resorting"

    }
  }
}
```

Bibliography

[1] L. R. Bahl, R. Bakis, P. S. Cohen, A.G. Cole, F. Jelinek, B.L. Lewis, and R.L. Mercer. Further results on the recognition of a continuously read natural corpus. In *Proceedings of the 1980 IEEE Conference on Acoustics, Speech, and Signal Processing*, April 1980.

[2] Y.L. Chow and R. Schwartz. The N-best algorithm: An efficient procedure for finding top N sentence hypotheses. In *Proceedings of the 1989 DARPA Speech and Natural Language Workshop*, pages 199–202, Cape Cod, MA, October 1989.

[3] Y.L. Chow, R. Schwartz, S. Roucos, O. Kimball, P. Price, F. Kubala, M. Dunham, M. Krasner, and J. Makhoul. The role of word-dependent coarticulatory effects in a phoneme-based speech recognition system. In *Proceedings of the 1986 International Conference on Acoustics, Speech, and Signal Processing*, pages 1593–1596, Tokyo, Japan, April 1986.

[4] A.W. Drake. *Fundamentals of Applied Probability Theory.* McGraw-Hill Publishing Company, 1967.

[5] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

[6] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, and N. Dahlgren. DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM, 1990.

[7] J. Glass, J. Chang, and M. McCandless. A probabilistic framework for feature-based speech recognition. In *Proceedings of the Fourth International Conference on Spoken Language Processing*, pages 2277–2280, Philadelphia, PA, October 1996.

[8] J. Glass, D. Goddeau, L. Hetherington, M. McCandless, C. Pao, M. Phillips, J. Polifroni, S. Seneff, and V. Zue. The MIT ATIS system: December 1994 progress report. In *Proceedings of the ARPA Spoken Language Systems Technology Workshop '95*, pages 252–256, Austin, TX, January 1995.

[9] L. Hetherington and M. McCandless. Sapphire: An extensible speech analysis and recognition tool based on tcl/tk. In *Proceedings of the Fourth International Conference on Spoken Language Processing*, pages 1942–1945, Philadelphia, PA, October 1996.

[10] L. Hetherington, M. Phillips, J. Glass, and V. Zue. A* word network search for continuous speech recognition. In *Proceedings of the Third European Conference on Speech Communication and Technology*, pages 1533–1536, Berlin, Germany, September 1993.

[11] X.D. Huang, M.Y. Hwang, L. Jiang, and M. Mahajan. Deleted interpolation and density sharing for continuous hidden Markov models. In *Proceedings of the 1996 International Conference on Acoustics, Speech, and Signal Processing*, pages 885–888, Atlanta, GA, May 1996.

[12] X.D. Huang, K.F. Lee, Hon H.W., and Hwang M.Y. Improved acoustic modeling with the SPHINX speech recognition system. In *Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing*, pages 345–348, Tokyo, Japan, May 1991.

[13] F. Jelinek and R.L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In E.S. Gelsema and L.N. Kanal, editors, *Pattern Recognition in Practice*, pages 381–397. North-Holland Publishing Company, 1980.

[14] R.L. L. R. Bahl, P.V. de Souza, P.S. Gopalakrishnan, D. Nahamoo, and M.A. Picheny. Decision trees for phonological rules in continuous speech. In *Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing*, pages 185–188, Tokyo, Japan, May 1991.

[15] C.H. Lee, L. Rabiner, R. Pieraccini, and J.G. Wilpon. Acoustic modeling of subword units for large vocabulary speaker independent speech recognition. In *Proceedings of the 1989 DARPA Speech and Natural Language Workshop*, pages 280–291, Tokyo, Japan, October 1989.

[16] C.H. Lee, L. Rabiner, R. Pieraccini, and J.G. Wilpon. Acoustic modeling for large vocabulary speech recognition. *Computer Speech and Language*, 4:127–165, 1990.

[17] K.F. Lee. *Automatic Speech Recognition: The Development of the SPHINX System*. Kluwer Academic Publishers, 1989.

[18] K.F. Lee. Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(4):599–609, April 1990.

[19] A. Ljolje. High accuracy phone recognition using context clustering and quasi-triphonic models. *Computer Speech and Language*, 8:129–151, 1994.

[20] M.D. Muzumdar. Automatic acoustic measurement optimization for segmental speech recognition. Master's thesis, Massachusetts Institute of Technology, 1996.

[21] M. Ostendorf and S. Roukos. A stochastic segment model for phoneme-based continuous speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-37(12):1857–1869, 1989.

[22] M. Phillips, J. Glass, and V. Zue. Modelling context dependency in acoustic-phonetic and lexical representations. In *Proceedings of the 1991 DARPA Speech and Natural Language Workshop*, pages 71–76, Genoa, Italy, February 1991.

[23] P. Price, W.M. Fisher, J. Bernstein, and D.S. Pallett. The DARPA 1000-word resource management database for continuous speech recognition. In *Proceedings of the 1988 International Conference on Acoustics, Speech, and Signal Processing*, pages 651–654, New York, NY, April 1988.

[24] L.R. Rabiner, J.G. Wilpon, and F.K. Soong. High performance connected digit recognition using hidden Markov models. In *Proceedings of the 1988 International Conference on Acoustics, Speech, and Signal Processing*, pages 119–122, New York, NY, April 1988.

[25] S. Roucos. Measuring perplexity of language models used in speech recognizers. Technical report, BBN Laboratories, 1987.

[26] R.M. Schwartz, Y.L. Chow, S. Roucos, M. Krasner, and J. Makhoul. Improved hidden Markov modeling of phonemes for continuous speech recognition. In *Proceedings of the 1984 International Conference on Acoustics, Speech, and Signal Processing*, pages 35.6.1–35.6.4, San Diego, CA, April 1984.

[27] F. K. Soong and E.-F. Huang. A tree-trellis based fast search for finding the n best sentence hypotheses in continuous speech recognition. In *Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing*, pages 705–708, Toronto, Canada, May 1991.

[28] P.H. Winston. *Artificial Intelligence.* Addison-Wesley Publishing Company, 1984.

[29] P. C. Woodland, C. J. Leggetter, J. J. Odell, V. Valtchev, and S.J. Young. The 1994 HTK large vocabulary speech recognition system. In *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing*, pages 73–76, Detroit, MI, May 1995.

[30] P.C. Woodland and S.J. Young. The HTK tied-state continuous speech recognizer. In *Proceedings of the Third European Conference on Speech Communication and Technology*, pages 2207–2210, Berlin, Germany, September 1993.

[31] V. Zue and J. Glass. Lecture Handouts for 6.345 ("Automatic Speech Recognition"), MIT, Spring 1995.