Near-Miss Modeling:
A Segment-Based Approach to Speech Recognition

by

Jane W. Chang

B.S., Stanford University (1992)
S.M., Massachusetts Institute of Technology (1995)

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1998

Author ....
Department of Electrical Engineering and Computer Science
May 20, 1998

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

Accepted by ....
Arthur Smith
Chairman, Departmental Committee on Graduate Students

Near-Miss Modeling:
A Segment-Based Approach to Speech Recognition

by

Jane W. Chang

Submitted to the Department of Electrical Engineering
and Computer Science
on May 20, 1998, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Currently, most approaches to speech recognition are frame-based in that they represent speech as a temporal sequence of feature vectors. Although these approaches have been successful, they cannot easily incorporate complex modeling strategies that may further improve speech recognition performance. In contrast, segment-based approaches represent speech as a temporal graph of feature vectors and facilitate the incorporation of a wide range of modeling strategies. However, difficulties in segment-based recognition have impeded the realization of potential advantages in modeling.

This thesis describes an approach called near-miss modeling that addresses the major difficulties in segment-based recognition. Probabilistically, each path should account for the entire graph including the segments that are off the path as well as the segments that are on the path. Near-miss modeling is based on the idea that an off-path segment can be modeled as a "near-miss" of an on-path segment. Each segment is associated with a near-miss subset of segments that contains the on-path segment as well as zero or more off-path segments such that the near-miss subsets that are associated with any path account for the entire graph. Computationally, the graph should contain only a small number of segments without introducing a large number of segmentation errors. Near-miss modeling runs a recognizer and produces a graph that contains only the segments on paths that score within a threshold of the best scoring path.

A near-miss recognizer using context-independent segment-based acoustic models, diphone context-dependent frame-based models, and a phone bigram language model achieves a 25.5% error rate on the TIMIT core test set over 39 classes. This is a 16% reduction in error rate from our best previously reported result and, to our knowledge, is the lowest error rate that has been reported under comparable conditions. Additional experiments using the ATIS corpus verify that these improvements generalize to word recognition.

Thesis Supervisor: James R. Glass
Title: Principal Research Scientist

Acknowledgments

First, I thank Jim for the guidance and encouragement he has given me over the past four years and especially in the last half year of writing. I also thank Victor and Paul for their insightful suggestions that have strengthened this thesis. In addition, I thank Chao, Drew, Joe, Lee, Michelle, Mike, Stephanie, and TJ for all of their helpful comments.

I am grateful to Victor and all of the members of SLS who have provided the inspiring research environment in which I have spent these past six years. I am sure that I will miss this environment whenever it is that I finally leave.

I am also grateful to Rich and AT&T Bell Laboratories for supporting me through my undergraduate career at Stanford and my graduate career here at MIT. I couldn't have come this far without their help.

I dedicate this thesis to Mom and Dad, who instilled in me the goal of the PhD and nurtured the abilities to attain my goals. I also thank Kay and Fay who have been my companions through all of these years.

In addition, I thank Victor, Stephanie, Greg, Soma, Tim, and Cory for all of their support. I am also grateful to Bill, Helen, and Melanie.

Finally, I thank Mike for making these past four years much more fun than I ever thought that they could be.

This research was supported by AT&T Bell Laboratories under a Graduate Research Program for Women Fellowship, by DARPA under Contract N66001-94-C-6040 monitored through Naval Command, Control and Ocean Surveillance Center, and by the National Science Foundation under Grant No. IRI-9618731.

To Mom and Dad

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Currently, most approaches to speech recognition represent the speech signal using a temporal sequence of feature vectors called frames and therefore are described as frame-based approaches. Typically, frames contain short-time spectral and energy information and are used to distinguish speech segments called phones, which in turn are used to distinguish words. In particular, most approaches use a finite state model called a hidden Markov model (HMM) to model how phones can be realized as frames [2, 53].

Although HMM approaches have been relatively successful, humans remain superior to state-of-the-art recognizers, and researchers continue to explore methods to improve speech recognition performance [34]. One of the most commonly targeted weaknesses of an HMM is its assumption of independence between frames. In an HMM, the frames within a phone are modeled independently even though they demonstrate a high degree of correlation. To overcome this weakness, researchers have developed methods to jointly model the frames within a phone [1, 19, 44]. Although these methods have been described as segment-based, they still use a sequence-based representation and therefore are still described as frame-based in this thesis.

A more fundamental limitation of the HMM approach is its inability to extract feature vectors across an entire phone. In an HMM, each frame typically does not span a phone and therefore cannot capture characteristics across the entire phone. To overcome this limitation, some researchers have pursued an alternative approach to speech recognition that represents the speech signal using a temporal graph of feature vectors, where each hypothesized phone is modeled using its own feature vector [5, 69]. In this thesis, the description of segment-based is reserved for such approaches that use a graph-based representation. Although segment-based approaches can potentially provide improvements in modeling, they have been relatively unsuccessful due to difficulties in recognition.

This thesis describes a new segment-based approach called near-miss modeling after its ability to model one segment as a "near-miss" of another. Near-miss modeling is a combination of two methods that overcome the major difficulties in developing a segment-based approach. First, near-miss search provides a method for enforcing constraints across a graph-based representation. Second, near-miss segmentation provides a method for producing a useful graph-based representation. Empirically, near-miss modeling is shown to achieve state-of-the-art performance in phonetic recognition. Furthermore, near-miss modeling enables the exploration of modeling strategies that may further improve speech recognition performance. This chapter introduces the motivation and difficulties in developing a segment-based approach and provides an overview and an outline of the thesis.

1.1 Motivation

The motivation for pursuing a segment-based approach to speech recognition is to enable improvements in modeling. To illustrate this motivation, Figure 1-1 shows an example speech utterance consisting of a waveform and spectrogram, frame- and segment-based representations, frame- and segment-based paths, and phone and word transcriptions, all aligned with time on the x-axis. The spectrogram shows the magnitude (dB) of the short-time Fourier transform of the utterance, with frequency on the y-axis and energy coded in gray level. In the frame- and segment-based representations, each rectangular region corresponds to a feature vector. The frame-based representation is a temporal sequence of feature vectors, where one feature vector is extracted every 10 ms. The segment-based representation is a temporal graph

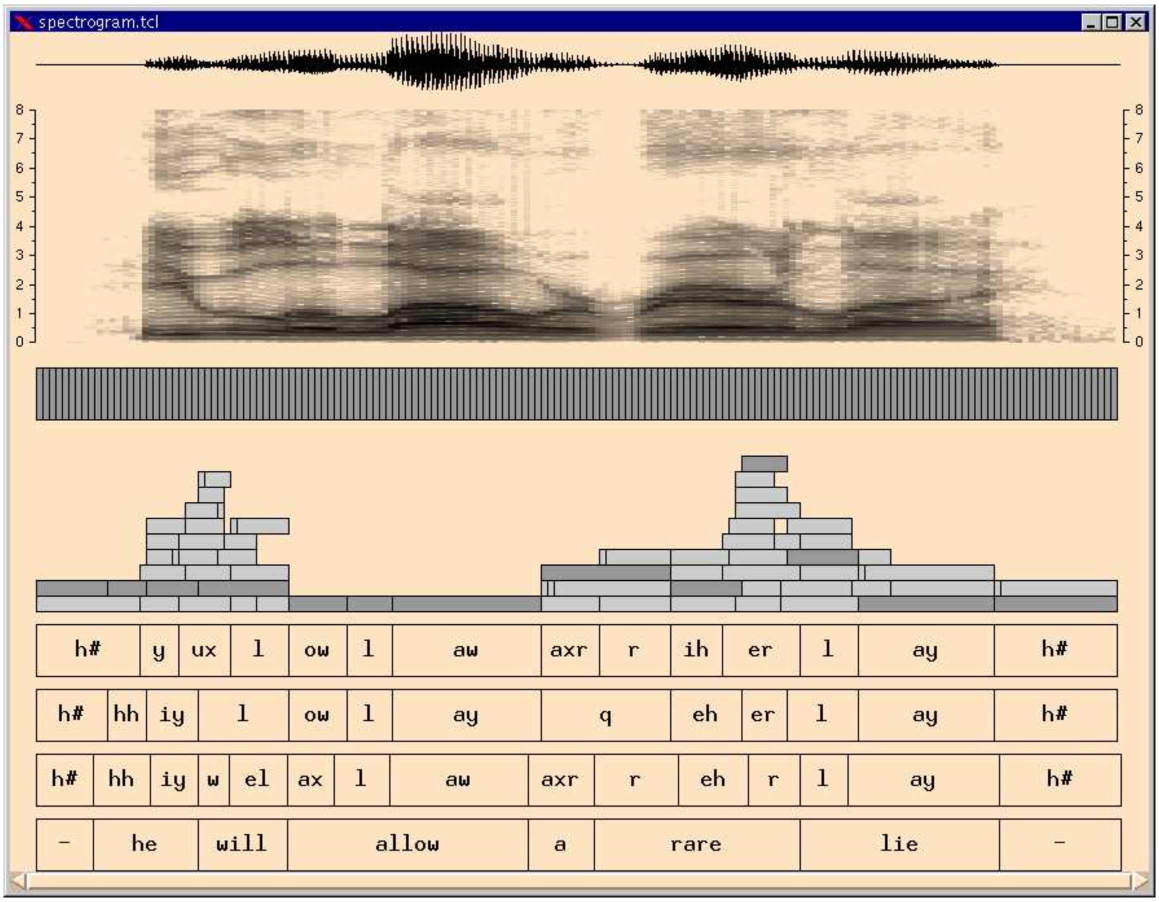


Figure 1-1: An example speech utterance consisting of a waveform and spectrogram, frame- and segment-based representations, frame- and segment-based paths, and phone and word transcriptions, all aligned with time on the x-axis. The spectrogram shows the magnitude (dB) of the short-time Fourier transform of the utterance, with frequency on the y-axis and energy coded in gray level. In the frame- and segment-based representations, each rectangular region corresponds to a feature vector. The frame-based representation is a sequence of feature vectors, where one feature vector is extracted every 10 ms. The segment-based representation is a graph of feature vectors, where each feature vector corresponds to a hypothesized phone. The goal in recognition is to find the best path through the representation. The best frame- and segment-based paths are shaded through their respective representations and also shown underneath. The frame-based path uses the entire sequence-based representation, while the segment-based path uses only a sequence of feature vectors through the graph-based representation. The utterance is extracted from the TIMIT corpus and labeled using the TIMIT phone labels [16, 20, 30].

of feature vectors, where each feature vector corresponds to a hypothesized phone. The goal in recognition is to find the best path through the representation. The best frame- and segment-based paths computed by phonetic recognition are shaded through their respective representations and also shown underneath. The frame-based path uses the entire sequence-based representation, while the segment-based path uses only a sequence of feature vectors through the graph-based representation. The utterance is extracted from the TIMIT corpus and labeled using the TIMIT phone labels [16, 20, 30].

Figure 1-1 suggests that phones have time, frequency and energy characteristics that are useful for recognition [13, 68, 67]. In particular, some of these characteristics may be better modeled at the segment level rather than the frame level. The most common example of such a characteristic is duration [9, 64]. Studies have shown that duration can help make fine distinctions between similar phones, such as between tense and lax vowels and between voiced and unvoiced consonants. A segment, unlike a frame, spans a phone and can capture its duration. In addition to duration, other time, frequency and energy characteristics may also be better modeled at the segment level. For example, studies have shown that the transitions between phones contain important information, and a segment can focus on these transition characteristics at its boundaries [37, 49]. It may also be useful to capture the timing of events within a segment, such as when and at what frequency an energy band peaks. Overall, a segment-based approach provides a richer framework for the exploration of improved modeling strategies. In addition, a segment-based approach is more general and offers the flexibility to explore both frame- and segment-based approaches.

1.2 Difficulties

Despite their potential advantages, segment-based approaches have been relatively unsuccessful largely due to two major difficulties in recognition. The first difficulty concerns the search process of finding the best path through a graph of segments. The second difficulty concerns the segmentation process of constraining the graph of segments for search. The following two sections elaborate on these difficulties.

1.2.1 Search

Most current approaches to speech recognition, whether they are frame- or segment-based, use a similar probabilistic framework [2, 53]. To directly compare paths, the probabilistic framework requires that all paths account for the entire set of feature vectors that is used to represent the speech signal. In a frame-based approach, each path accounts for the entire sequence-based representation, and therefore the search strategy is straightforward. However, in a segment-based approach, each feature vector corresponds to a segment, and each path only accounts for a sequence of segments through the graph-based representation. To maintain the probabilistic framework, a segment-based approach requires a more complex search strategy that accounts for the *entire* graph of segments, including both the segments that are on a path as well as the segments that are off a path.

Recently, we have recognized the necessity of accounting for the entire graph of segments. To this end, we have developed a segment-based framework called anti-phone modeling based on the idea that an off-path segment is not a phone and therefore can be modeled as an anti-phone [22]. Anti-phone modeling maintains the probabilistic framework by normalizing all paths to implicitly account for all segments. However, anti-phone modeling requires all off-path segments to be modeled by a single anti-phone model even though off-path segments can vary greatly with context. For example, off-path segments through vocalic regions have different characteristics than off-path segments through consonantal regions. The inability to enforce contextual constraints across all segments limits the modeling strategies than can be explored and limits the recognition performance, thereby impeding the development of segment-based approaches.

1.2.2 Segmentation

Most current approaches to speech recognition, whether they are frame- or segment-based, also use a similar dynamic programming strategy [2, 53]. To efficiently compare paths, dynamic programming takes advantage of shared structure between paths. In a frame-based approach, each path shares the entire sequence-based representation, and therefore a frame-based approach can efficiently search all possible connections of frames into segments. However, in a segment-based approach, different paths may account for different segments through the graph-based representation. To reduce computation, a segment-based approach requires a more complex segmentation strategy that constrains the graph of segments that is searched.

Currently, the SUMMIT framework developed in our group uses an acoustic segmentation algorithm [69]. The acoustic segmentation algorithm hypothesizes boundaries at times of large spectral change and connects the boundaries into segments. However, although this algorithm is efficient, segmentation often depends on contextual effects that cannot be captured by spectral change alone. For example, transitions between similar phones, such as between vowels, tend to be gradual and may not be delimited by large spectral change. The introduction of errors in segmentation causes errors in recognition and undermines potential gains in modeling, thereby further impeding the development of segment-based approaches.

1.3 Overview

The objective of this thesis is to develop an approach to speech recognition that can overcome the difficulties that currently impede progress in segment-based recognition. The approach is called near-miss modeling based on the idea that an off-path segment can be modeled as a "near-miss" of an on-path segment. The near-miss search associates each segment with a near-miss subset of segments that contains the on-path segment as well as zero or more off-path segments such that the near-miss subsets that are associated with any path account for the entire graph. As a result, the near-miss search can maintain the probabilistic framework without sacrificing the ability to enforce contextual constraints across all segments. In addition, near-miss segmentation runs a recognizer and hypothesizes only the segments on paths that score within a threshold of the best scoring path. As a result, near-miss segmentation can generate a small number of segments without introducing large numbers of segmentation errors.

Near-miss modeling is primarily evaluated on the task of phonetic recognition. A near-miss recognizer using diphone context-dependent acoustic models and a phone bigram language model achieves a 25.5% error rate on the TIMIT core test set over 39 classes [16, 20, 30]. This is a 16% reduction in error rate from our best previously reported result and, to our knowledge, is the lowest error rate that has been reported under comparable conditions. Additional experiments using the ATIS corpus verify that these improvements generalize to word recognition [47, 48]. Furthermore, near-miss modeling enables the exploration of modeling strategies that promise even greater improvements in the future.

1.4 Outline

The remainder of the thesis is organized in six chapters:

- Chapter 2 sets the background for the thesis. It describes the framework, modeling strategies, and search algorithms that are used in most speech recognition systems. It also specializes this background to the frame- and segment-based approaches.

- Chapter 3 details the framework for the experiments in phonetic recognition that serve as the primary evaluation of the thesis. This chapter describes the corpus and recognizers that are used in phonetic recognition.

- Chapter 4 describes the search framework that is used in near-miss modeling. It begins with an example of the near-miss modeling problem. It then describes the near-miss assignment algorithm and the resulting near-miss framework for speech recognition. It then describes issues in assigning near-miss subsets and modeling the near-miss units. Finally, it evaluates near-miss modeling in phonetic recognition.

- Chapter 5 describes the segmentation algorithm that is used in near-miss modeling. It describes a general framework for segmentation. It evaluates segmentation on phonetic recognition and compares near-miss modeling against other approaches to speech recognition.

- Chapter 6 describes the experiments in word recognition. The chapter parallels Chapters 3, 4, and 5 by describing the experimental framework, evaluating near-miss modeling in search and segmentation, and a comparing near-miss modeling to other approaches, all on the task of word recognition.

- Chapter 7 concludes the thesis. It summarizes the contributions of this thesis and suggests directions for future research.

Chapter 2

Background

This chapter sets the background for the thesis. The first three sections describe the framework, modeling strategies, and search algorithms that are used in most approaches to speech recognition. The remaining two sections focus on the background to the frame- and segment-based approaches.

2.1 Probabilistic Framework

Currently, most approaches to speech recognition are based on a similar probabilistic framework that provides a method for combining the diverse sources of constraints that are used in speech recognition [2, 53]. The framework can be described using two terms: W is a sequence of words, and A is a set of acoustic feature vectors. Using these terms, the goal of speech recognition is to find W^*, the word sequence that maximizes $P(W|A)$, the posterior probability of a word sequence given a set of feature vectors:

$$W^* = \arg\max_{W} P(W|A)$$

This framework is simplified by decomposing a word sequence into two simpler sequences: a sequence of linguistic units and a sequence of temporal segments. The following two sections describe these units and segments, and the next section describes how they are used in speech recognition.

2.1.1 Units

Except for tasks that are constrained to a small vocabulary, there are too many words to allow the creation of robust whole word models [54]. Linguistically, each word, and therefore each word sequence, can be decomposed into a sequence of subword units called a pronunciation. As a result, all words can share a relatively small set of subword units which can be directly modeled. Currently, the most commonly used subword units are based on the fundamental linguistic units that distinguish words, called phonemes, or their acoustic realizations, called phones [13]. English has approximately 40 phonemes.

2.1.2 Segments

Temporally, each unit, such as a phone, is assumed to occupy a finite amount of time called a segment. Furthermore, the sequence of segments that is associated with a unit sequence must span the duration of the utterance in a contiguous and non-overlapping manner. Note that a segment sequence has also been called a segmentation. However, in this thesis, the term segmentation is reserved for the process of producing a graph of segments, which typically contains multiple segment sequences.

2.1.3 Summation

The probabilistic framework can be extended to include two more terms: U is a unit sequence, and S is a segment sequence. Each word sequence can be decomposed into one or more unit sequences, which in turn can be associated with one or more segment sequences. The probability of a word sequence is computed by summing the probabilities of all possible unit and segment sequences that are associated with that word sequence:

$$P(W|A) = \sum_{US} P(WUS|A)$$

To reduce computation, the summation over unit and segment sequences is approximated with a maximization [2, 53]. For simplicity, a combination of word, unit, and segment sequences is called a path. As a result, the goal of speech recognition is to find the best path that maximizes the posterior probability of a path given a set of feature vectors:

$$W^*U^*S^* = \arg\max_{WUS} P(WUS|A)$$

2.2 Models

The framework can be further simplified by separating the sources of constraint used in speech recognition [2, 54]. To do this, $P(WUS|A)$ is expanded by successive applications of Bayes' Rule:

$$P(WUS|A) = \frac{P(A|WUS)P(S|UW)P(U|W)P(W)}{P(A)}$$

Since $P(A)$ is always constant relative to the maximization, it can be dropped from the formulation.

$$W^*U^*S^* = \arg\max_{WUS} P(A|WUS)P(S|UW)P(U|W)P(W)$$

As a result, there are four constraints: $P(A|WUS)$ is called an acoustic constraint, $P(S|UW)$ is called a duration constraint, $P(U|W)$ is called a pronunciation constraint, and $P(W)$ is called a language constraint. The following four sections discuss how each of these constraints is modeled.

2.2.1 Acoustic Model

The acoustic constraint, $P(A|WUS)$, is estimated by an acoustic model [2, 54]. The goal in acoustic modeling is to score how well a set of feature vectors represents a hypothesized path. The following four sections describe the procedures that are involved in acoustic modeling.

Representation

The speech signal is typically transformed into a cepstral representation by using short-time Fourier analysis or linear predictive analysis [42, 55]. During this procedure, other signal processing techniques may be applied. For example, auditory processing can be used to capture auditory constraints or normalization techniques can be used to account for environmental effects [6, 12, 35, 38].

Feature Extraction

The initial signal representation is used to extract a set of feature vectors. Typical feature vectors are cepstral vectors or averages and derivatives of cepstral vectors. The feature vectors can also include more knowledge-based feature vectors such as formant frequencies [60]. In addition, feature vectors can be determined empirically by running automated search procedures on training data [40, 51].

Classification

The feature vectors are modeled by a pattern classifier [15]. Currently, the most commonly used classifier is based on a mixture of Gaussian distributions [4, 54, 66, 70]. To improve efficiency and robustness, many systems assume that the Gaussian covariance matrices are diagonal. In addition, it is useful to diagonalize the feature vectors and reduce their dimensionality by principal components analysis [15, 21]. The mixtures can be trained in an unsupervised manner, such as k-means or Expectation-Maximization (EM) [15, 21].

Unit Selection

Although word-dependent units are sometimes used, typically the units are based on phones, and the acoustic model estimates $P(A|US)$. The phones that are modeled may or may not depend on context. For example, when the phones are context-independent, each phone is assumed to be independent of its phonetic context. However, there can be a large degree of variation within the same phone depending on its context. As a result, most state-of-the-art speech recognition systems use context-dependent phones, where a phone is modeled depending on its neighboring phones [31, 36, 61]. In addition to phonetic context, systems can also enforce other types of context. For example, gender-dependent units allow the acoustic models to focus on male or female speech [26]. In general, context-dependent modeling increases computation and training requirements, and the number of context-dependent units that can be modeled is limited. As the number of units increases, computation typically increases. In addition, the amount of training data per unit decreases, and the resulting models become less robust.

2.2.2 Duration Model

The duration constraint, $P(S|WU)$, is estimated by a duration model [2, 54]. The goal in duration modeling is to score how well a sequence of segment and word times temporally matches a hypothesized unit or word sequence. Most systems use a simple duration model based on a segment or word transition weight whereby a weight is multiplied in for each segment or word transition in the hypothesized path [44, 69]. This weight trades off deletions and insertions, with a larger weight favoring sequences with more transitions, and a smaller weight favoring sequences with fewer transitions. Duration models can also be based on Gaussian and other distributions.

2.2.3 Pronunciation Model

The pronunciation constraint, $P(U|W)$, is estimated by a pronunciation model [2, 54]. The goal in pronunciation modeling is to score how well a unit sequence represents a pronunciation of a word sequence. The simplest pronunciation model admits only a single pronunciation per word. However, some words, such as "data", have multiple pronunciations. In addition, words can be pronounced in different ways depending on context. There are various approaches for modeling alternate pronunciations. One technique is to categorically allow the deletion or insertion of a phone with some penalty. However, much of the phonological variation is systematic and therefore can be modeled more directly [43, 50, 56]. Many of the variations can be captured in general phonological rules. For example, the last phone in the word "from" and the first phone in the word "Miami" are both /m/. When the two words are pronounced in sequence, as in "from Miami", they can share the same /m/. The effect, called gemination, occurs systematically across word boundaries, so a rule can be used to allow the reduction of identical phones across word boundaries. Phonological rules can also be derived automatically [56].

2.2.4 Language Model

The language constraint, $P(W)$, is estimated by a language model [2, 54]. The goal of language modeling is to score how well a word sequence represents a valid sentence in a language. The simplest language model is a uniform distribution, where every word is equally likely to follow any given word. However, there are many constraints in language, such as syntax, semantics, discourse, and dialogue, which can be used to provide more predictive power for the language model. The following sections introduce a common statistical language model, called an n-gram, which uses local word order to provide constraint, and a common evaluation metric, called perplexity.

n-gram

For a word sequence with N words, $P(W)$ may be expanded causally by the chain rule [2, 54]:

$$P(W) = \prod_{i=1}^{N} P(w_i|w_{i-1}...w_1)$$

The n-gram language model assumes that the probability of a word depends only on the previous $n-1$ words:

$$P(W) = \prod_{i=1}^{N} P(w_i|w_{i-1}...w_{i-(n-1)})$$

To reduce computation, many systems use a bigram language model in which $n = 2$, and the probability of a word depends only on the immediately preceding word:

$$P(W) = \prod_{i=1}^{N} P(w_i|w_{i-1})$$

The n-gram probabilities are estimated by gathering statistics from a training set. The performance of the n-gram is dependent upon many factors. To improve coverage, words can be added to the vocabulary. This includes the addition of compound words that can be used to model common sequences of words, such as "San Francisco," as a single word. To derive more robust estimates, higher order n-grams can be smoothed with lower order n-grams [70]. In addition, a class n-gram language model can be used to model classes of words that share relatively similar probability distributions, thus making more effective use of limited training data [4, 54, 66, 70]. For example, both the words, "Boston" and "San_Francisco" can be classed in the "city" class.

Perplexity

The complexity of a task and the power of a language model is often evaluated by perplexity [2, 54]. The perplexity of a word sequence with N words under a language model is:

$$Perplexity = 2^{-\frac{\log P(W)}{N}}$$

Perplexity is always measured on a test set which was not used to train the language model parameters. Perplexity measures the predictive power of a language model and can be loosely interpreted as the average number of words which can follow any word under the language model. The higher the perplexity, the more confusable the task and the less effective the language model.

2.3 Search

The goal of search is to combine the diverse sources of constraints according to the probabilistic framework and to find the best scoring path based on all model scores. The following sections describe some of the search space and the search algorithms that are used in speech recognition.

2.3.1 Search Space

This thesis focuses on the subword level, where the search space can be visualized as a segment graph. During recognition, this graph is intersected with a graph representing the pronunciation and language models. A graph that fully interconnects n times has $\frac{n(n-1)}{2}$ segments. Figure 2-1 shows an example segment graph that is fully interconnected over four times, t_i, with six segments, s_i. Note that throughout this thesis, the subscripts in the text are not subscripted in the figures.

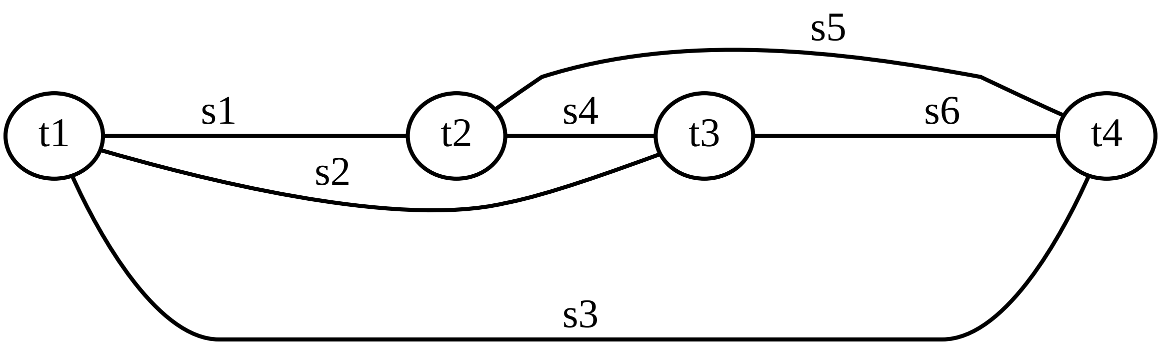


Figure 2-1: Example segment graph that fully interconnects four times, t_i, using six segments, s_i.

The segment graph is a compact representation for the space of all possible segment sequences. In a graph that is fully interconnected over n times, there are 2^{n-2} segment sequences. Table 2.1 shows the four segment sequences in Figure 2-1, where the sequence, S_{ijk}, contains the segments, s_i, s_j, and s_k.

Sequence	Segments
S_{146}	$s_1 s_4 s_6$
S_{15}	$s_1 s_5$
S_{26}	$s_2 s_6$
S_3	s_3

Table 2.1: The four segment sequences in Figure 2-1, where the sequence, S_{ijk} contains the segments, s_i, s_j, and s_k.

2.3.2 Search Algorithms

To search all possible paths, many speech recognition systems use dynamic programming strategies [11]. Dynamic programming applies to problems that can be configured to have optimal substructure and overlapping subproblems. A problem exhibits optimal substructure if the optimal solution to the problem contains optimal solutions to subproblems of the problem. A subproblem is overlapping if it has to be solved over and over again. For problems that can be configured to have such structure, dynamic programming can be used to efficiently find a solution by storing the solutions to the subproblems and re-using them. The following sections describe two search algorithms that are commonly used in speech recognition.

Viterbi Algorithm

In speech recognition, the dynamic programming algorithm called the Viterbi algorithm is often used to find the best path through a search space [2, 17, 54, 63]. The Viterbi algorithm is a time-synchronous search that explores the entire search space by completely processing all paths ending at one time point before extending them to the next time point. The Viterbi search can be extended to offer an efficient and effective method of pruning, called beam pruning, by retaining only the best scoring paths at each time point. With pruning, although the result is not guaranteed to be optimal, in practice there can be little degradation in performance with significantly less computation. The Viterbi search is an efficient way of applying local constraints and finding the best path through a graph. However, it cannot easily apply longer distance constraints or find alternate paths.

A^* Algorithm

To apply more complex constraints or to find the n-best paths, many speech recognition systems use an A^* search [7, 11, 27, 41, 45]. In the A^* search strategy, partial paths are maintained in sorted order in a queue. During search, the best partial path is extracted from the queue, and all of its possible extensions are subsequently inserted. The score of the extensions is based on the score of the partial path plus an upper bound estimate of the remaining score. If this estimate is guaranteed to be greater than or equal to the actual remaining score, then the A^* search is guaranteed to find the best path. Since each node has a unique history, the A^* search facilitates the application of long distance constraints and can compute the n best paths.

One common recognition strategy is to compute n-best lists using the A^* search and to subsequently resort the n-best lists rather than searching the entire search space [7, 45]. This n-best paradigm is often used to incorporate expensive modeling strategies. However, n-best paths typically have significant overlap between them and may be an inefficient representation, especially for a large search space with many overlapping paths. To address this problem, researchers have modified the A^* search into a word graph search that collapses the n-best paths into a word graph [27, 41]. The modification involves merging paths that arrive at the same point in the A^* search. A word graph search provides both computational and representational efficiency. The path merging during the search results in computational savings, while the graph output is representationally more compact than a list of paths.

2.4 Frame-Based Approach

The probabilistic framework described so far can be specialized to particular approaches to speech recognition. The following two sections describe the basic frame-based representation and search strategy. The next two sections describe the dominant frame-based approach based on the hidden Markov model (HMM) and some of its extensions.

2.4.1 Frame-Based Representation

A frame-based approach represents speech as a temporal sequence of feature vectors. For example, Figure 2-2 shows an example frame-based representation that spans four times, t_i, with three frame-based feature vectors, a_i.

In a typical frame-based approach, the feature vectors are computed at a fixed

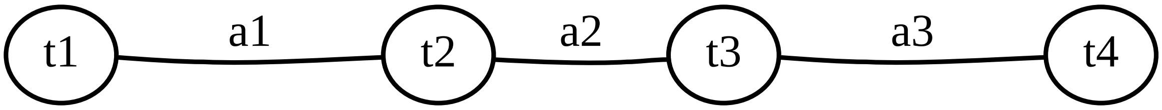


Figure 2-2: Example frame-based representation that spans four times, t_i, with three frame-based feature vectors, a_i.

rate, such as every 10 ms [2, 54]. However, in general, frame-based feature vectors can be computed at a variable rate [37, 44, 52]. To differentiate these variable frame rate approaches, they are also described as landmark-based.

2.4.2 Frame-Based Search

Figure 2-3 shows how the frame-based feature vectors in Figure 2-2 map to the segment graph in Figure 2-1.

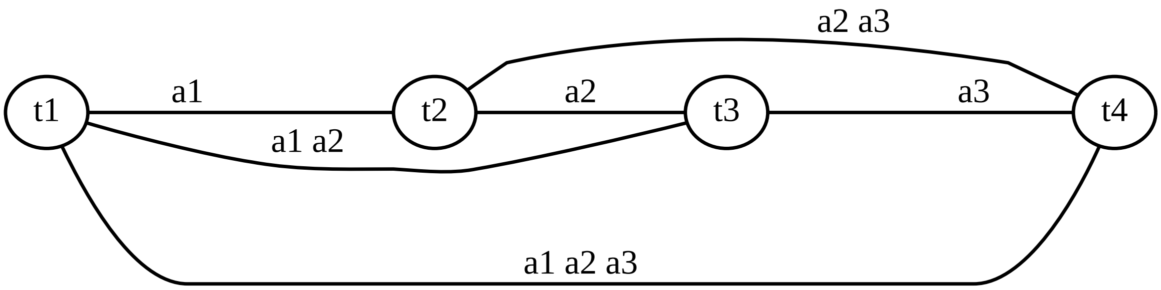


Figure 2-3: The frame-based feature vectors in Figure 2-2 mapped to the segment graph in Figure 2-1. In a frame-based search, each segment is represented by the sequence of the feature vectors which it spans, and each segment sequence accounts for all feature vectors.

In a frame-based approach, a segment is typically represented by a variable number of feature vectors. For example, when the feature vectors are computed at a fixed rate, the number of feature vectors that represent a segment is directly proportional to the duration of the segment. Each segment is represented by the sequence of the feature vectors which it spans. As shown, each segment sequence naturally accounts for all feature vectors.

Mathematically, each segment, s_i, spans a sequence of frame-based feature vectors, A_i, such that each segment sequence, S, accounts for all feature vectors:

$$A = \bigcup_{s_i \in S} A_i$$

In practice, most speech recognition systems assume independence at the segment level. As a result, a frame-based search can compute a total path score as the product of the scores of the feature vectors that are associated with the segments on the path:

$$P(A|US) = \prod_{s_i \in S} P(A_i|US)$$

For example, the four segment sequences in Figure 2-1 require the following computations:

$$\begin{aligned}
P(A|US_{146}) &= P(a_1|US_{146})P(a_2|US_{146})P(a_3a_4|US_{146}) \\
P(A|US_{15}) &= P(a_1|US_{15})P(a_2a_3a_4|US_{15}) \\
P(A|US_{26}) &= P(a_1a_2|US_{26})P(a_3a_4|US_{26}) \\
P(A|US_3) &= P(a_1a_2a_3a_4|US_3)
\end{aligned}$$

As shown, in a frame-based approach, the feature vectors are easily associated with each segment, so that any segment sequence accounts for all feature vectors. The following section describes a particularly efficient and effective method for estimating the probability of the feature vectors that are associated with a segment, $P(A_i|US)$.

2.4.3 HMM

The HMM can be described as a generative model [2, 31, 54]. An HMM models speech as a collection of states that are connected by transitions. Each state associates an output observation with a corresponding output probability, and each transition is associated with a transition probability that reflects the likelihood of transition. The model is Markov in the sense that the probability of a state at a given time point depends only on the state at the previous time point. The states are hidden in the sense that they are observable only through the sequence of output observation.

In the HMM approach to speech recognition, each speech unit is modeled using an HMM [2, 31, 54]. The feature vectors are the output observations, and the output probabilities model the acoustic constraint. The transition probability between HMM states models the duration constraint. In speech recognition, HMMs have a small number of states due to limitations in training data [54]. In addition, HMM states have self-loops to allow each segment to be realized as a variable number of frames. Figure 2-4 shows an example of a simple transition diagram that consists of three states, q_i, and five transitions, b_{ij}, from state q_i to state q_j.

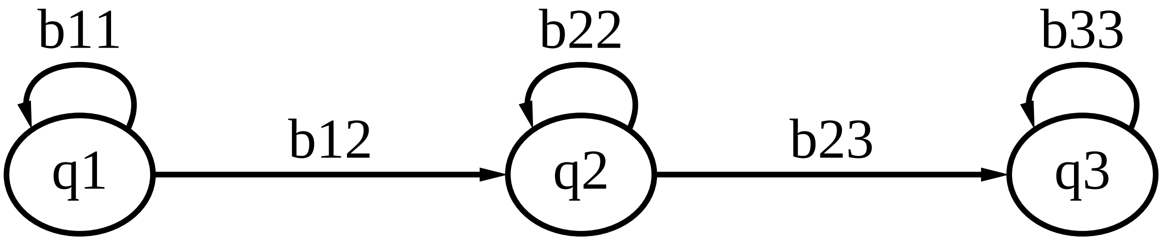


Figure 2-4: Example state transition diagram with three states, q_i, and five transitions, b_{ij}, from state q_i to state q_j.

During recognition, the goal is to find the sequence of states which best predicts the observed feature vectors. To do this, the typical HMM that is used in speech recognition assumes that the output probability depends only on the current state [2, 31, 54]. Although the frame-based feature vectors across a segment are certainly dependent upon each other, this assumption enables HMMs to take advantage of efficient frame-based search algorithms. HMM approaches typically use a Viterbi search to consider all paths in the segment graph. The application of the Viterbi search to an HMM has particularly effective pruning characteristics, since each path shares the same feature vectors, and furthermore, all feature vectors are independent. In training, some HMM approaches use the forward-backward, or Baum-Welch, algorithm to effectively estimate model parameters [2, 54]. The forward-backward algorithm is an instance of the Expectation-Maximization (EM) algorithm and is thus guaranteed to improve training set probability with each iteration.

Mathematically, HMMs model the feature vectors, A_i, that correspond to a segment, s_i, by assuming conditional independence between the feature vectors, a_j, that are associated with the segment:

$$P(A_i|US) = \prod_{a_j \in A_i} P(a_j|US)$$

As a result, HMMs can independently score the feature vectors across a path:

$$P(A|US) = \prod_{A_i \in A} \prod_{a_j \in A_i} P(a_j|US)$$

Overall, the HMM approach offers an efficient method of modeling and searching all possible paths. For example, the four segment sequences in Figure 2-1 can be computed, one feature vector at a time:

$$\begin{aligned}
P(A|US_{146}) &= P(a_1|US_{146})P(a_2|US_{146})P(a_3|Us_{146})P(a_4|US_{146}) \\
P(A|US_{15}) &= P(a_1|US_{15})P(a_2|US_{15})P(a_3|US_{15})P(a_4|US_{15}) \\
P(A|US_{26}) &= P(a_1|US_{26})P(a_2|US_{26})P(a_3|US_{26})P(a_4|US_{26})
\end{aligned}$$

$$P(A|US_3) \quad = \quad P(a_1|US_3)P(a_2|US_3)P(a_3|US_3)P(a_4|US_3)$$

2.4.4 HMM Extensions

Due to its advantages in search, HMMs have become the dominant approach in speech recognition [4, 54, 66]. However, studies have shown that the basic assumption of conditionally independent observations given the underlying state sequence is inaccurate [14, 24]. To better model correlation across a segment, HMMs have been extended in various ways to relax this assumption. The simplest method is to use feature vectors that are less sensitive to or remove some of the dependence [18, 31]. These include dynamic feature vectors such as derivatives and feature vectors that span longer durations in an attempt to implicitly capture more segmental characteristics. Other efforts have focused on developing better segment models through strategies such as trajectory modeling or using neural networks [1, 14].

More complex extensions focus on the HMM itself. To more explicitly capture correlation, segmental HMMs assume that each observation is dependent not only on the state but also on its mean over the segment of speech which it represents [19, 59]. For each state, the output probability is described by two distributions: one describing the segment mean and the other describing the observation given the mean. Stochastic segment modeling [44] is described as a generalization of hidden Markov modeling which relaxes the independence assumption between frames and allows the explicit modeling of correlation across frames within a segment. Segment models can be thought of as a higher dimensional version of an HMM, where a single Markov state may generate a sequence of vector observations rather than a single vector.

Overall, these methods have been shown to improve performance and provide further evidence for the potential advantages of segment-based modeling. However, an HMM cannot be extended to model segment-based feature vectors. Since a single frame-based feature vector typically does not span an entire unit of speech, a frame-based model cannot capture constraints associated with the entire unit. The most common example of such a constraint is duration. An HMM implicitly models duration through the transition probabilities, resulting in a geometric distribution. However, actual segment durations are poorly modeled by a geometric distribution [44]. The pursuit of such strategies requires a more general recognition framework, as provided by a segment-based approach.

2.5 Segment-Based Approach

A segment-based approach offers an alternative framework for recognition that can incorporate the same information as an HMM but also allow explorations of segment-based modeling strategies [5, 69]. The following section describes the segment-based representation. The next two sections describe two major difficulties in segment-based recognition concerning search and segmentation. The final section discusses some of the potential advantages of a segment-based approach.

2.5.1 Segment-Based Representation

In contrast to the frame-based approach, a segment-based approach represents speech as a temporal graph of segments, where each feature vector is associated with a segment of speech such as a phone [5, 69]. Figure 2-5 shows an example segment-based representation that fully interconnects four times, t_i, with six segment-based feature vectors, a_i.

The segment-based approach is a generalization of the frame-based approach and allows the extraction of both frame- and segment-based feature vectors. For example, the frame-based configuration shown in Figure 2-3 is just one form of the general segment-based configuration shown in Figure 2-5. In this case, if an HMM is used to model each unit, the segment-based system would be functionally equivalent to an HMM system. However, the systems would differ computationally. The HMM approach has been optimized to use a simpler representation, and therefore can be more computationally efficient given such a representation. In contrast, segment-based approaches allow more general representations and therefore cannot take advantage of

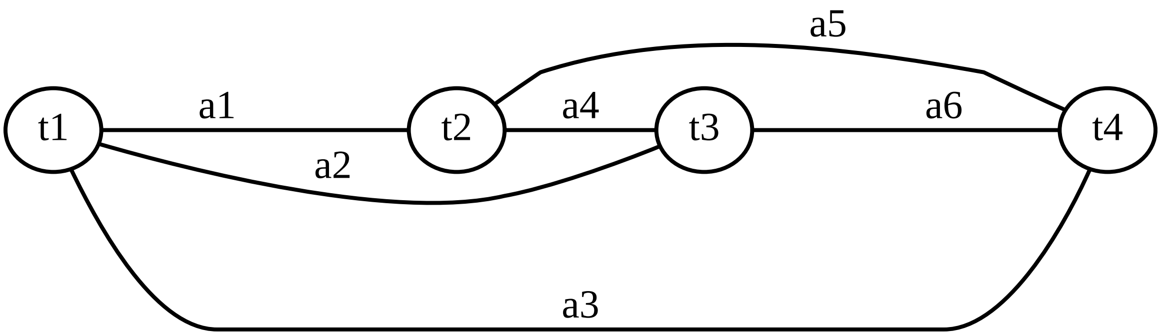


Figure 2-5: This example segment-based representation fully interconnects four times, t_i, with six segment-based feature vectors, a_i.

the same efficiencies.

2.5.2 Segment-Based Search

In a segment-based approach, each segment, s_i, is associated with its own feature vector, a_i. Probabilistically, a path must account for the entire set of feature vectors, A. However, each segment sequence, S, that is associated with the path does not in general account for all of the segments or feature vectors in the graph:

$$A \neq \bigcup_{s_i \in S} a_i$$

Heuristic Approaches

In the past, many segment-based approaches have used a heuristic framework to incorporate segment-based feature vectors [31, 67]. In scoring a path, these approaches only accounted for the segments in the path. They then used heuristic methods to normalize the different paths and allow for direct comparison [67]. As a result, many of these segment-based approaches performed poorly and were not pursued [31].

Anti-Phone Modeling

Recently, we have realized the need to model the entire set of feature vectors during segment-based search and have developed a strategy called anti-phone modeling to do so [21]. Anti-phone modeling is based on the idea that off-path segments are not phones and therefore should be modeled using an anti-phone unit. Each segment sequence, S, divides the entire set of feature vectors, A, into two subsets: A_S is the subset of feature vectors that are associated with the segment sequence being explored by the search, while $\bar{A}_S$ is the subset of feature vectors that is not associated with the segment sequence:

$$A = A_S \bigcup \bar{A}_S$$

As a result, a segment-based search should account for both subsets of feature vectors:

$$P(A|US) = P(A_S \bar{A}_S|US)$$

Anti-phone modeling uses a single non-lexical model, called the anti-phone or $\bar{\alpha}$, to model the feature vectors that are not associated with the segment sequence:

$$P(\bar{A}_S|US) = P(\bar{A}_S|\bar{\alpha})$$

Anti-phone modeling then eliminates the dependence on the off-path segments by estimating a likelihood ratio. As described, most speech recognition systems assume independence at the segment level:

$$
\begin{aligned}
L(A|US) &= \frac{P(A|US)}{P(A|\bar{\alpha})} \\
&= \frac{P(A_S\bar{A}_S|US)}{P(A_S\bar{A}_S|\bar{\alpha})} \\
&= \frac{P(A_S|US)P(\bar{A}_S|\bar{\alpha})}{P(A_S|\bar{\alpha})P(\bar{A}_S|\bar{\alpha})} \\
&= \frac{P(A_S|US)}{P(A_S|\bar{\alpha})} \\
&= \prod_{s_i \in S} \frac{P(a_i|US)}{P(a_i|\bar{\alpha})}
\end{aligned}
$$

As a result, anti-phone modeling can be interpreted as a normalization technique that allows the direct comparison of different segment sequences by normalizing the probability of each segment-based feature vector by its probability of not being a phone. The anti-phone model captures the general characteristics of segments that are not valid examples of phones and provides a means of normalizing scores for direct comparison.

For example, in Figure 2-5, the paths can be directly compared by scoring only the on-path segments:

$$
\begin{aligned}
L(A_{146}|US_{146}) &= \frac{P(a_1|US_{146})}{P(a_1|\bar{\alpha})}\frac{P(a_4|US_{146})}{P(a_4|\bar{\alpha})}\frac{P(a_6|US_{146})}{P(a_6|\bar{\alpha})} \\
L(A_{15}|US_{15}) &= \frac{P(a_1|US_{15})}{P(a_1|\bar{\alpha})}\frac{P(a_5|US_{15})}{P(a_5|\bar{\alpha})} \\
L(A_{26}|US_{26}) &= \frac{P(a_2|US_{26})}{P(a_2|\bar{\alpha})}\frac{P(a_6|US_{26})}{P(a_6|\bar{\alpha})} \\
L(A_3|US_3) &= \frac{P(a_3|US_3)}{P(a_3|\bar{\alpha})}
\end{aligned}
$$

Overall, anti-phone modeling uses a probabilistic framework that provides significant improvements in performance over our previous heuristic framework [21]. However, anti-phone modeling requires that all off-path segments be modeled by a *single* anti-phone class and does not allow these segments to be modeled in a context-dependent manner. This sacrifices the ability of a segment-based search to enforce constraints across the entire graph of segments.

2.5.3 Segmentation

Segmentation refers to the process of generating a graph of segments for search. Theoretically, a segment-based approach can search through a fully interconnected graph of segments. However, since the number of segments grows as the square of the number of times, it is computationally expensive to perform such an exhaustive search. As a result, many segment-based approaches constrain the search space by pruning the segment graph prior to search.

For example, while Figure 2-1 shows a fully interconnected segment graph, Figure 2-6 shows a segment graph that is only partially connected over 4 times, t_i, using 5 segments, s_i. The pruned segment graph does not contain the third segment, s_3, that starts at t_1 and ends at t_4, and therefore does not contain the segment sequence, S_3, that includes only the segment s_3.

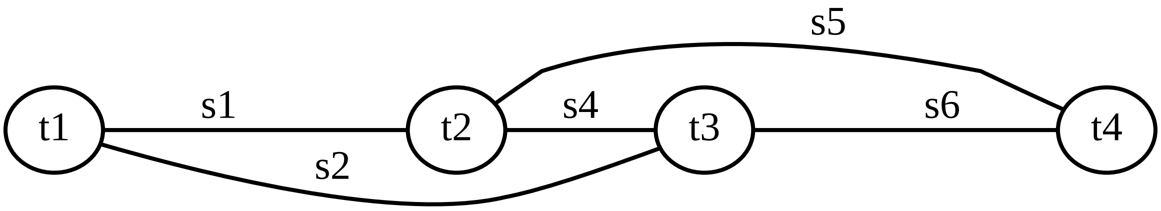


Figure 2-6: This segment graph is only partially connected over 4 times, t_i, using 5 segments, s_i. The pruned segment graph does not contain the third segment, s_3, that starts at t_1 and ends at t_4, and therefore does not contain the segment sequence, S_3, that includes only the segment s_3.

Segmentation is a difficult problem that often results in poor alignments, deletions and insertions of important phonetic events. These errors cannot be easily corrected by subsequent stages of recognition and typically create errors. In Figure 2-6, for example, if the correct path consisted of only one phone, the segment graph would not include a segment sequence that could align to one phone.

In general, segmentation results in a tradeoff between performance and computation. On the one hand, a segmentation algorithm can include a large number of segments in the segment graphs to avoid errors but sacrifices the efficiency of the subsequent search. On the other hand, a segmentation algorithm can generate a small segment graph but sacrifice the performance of the subsequent search.

Acoustic Segmentation

Historically, the SUMMIT system has used a variety of different acoustic segmentation algorithms [21]. Currently, we use an acoustic segmentation algorithm which detects landmarks and produces fully interconnected blocks [21]. First, major segment boundaries are hypothesized when a measure of spectral change exceeds a pre-specified global threshold. Then, minor segment boundaries are hypothesized between the major segment boundaries when the spectral change exceeds a local threshold that is computed between the major segment boundaries. Finally, all segment boundaries between major segment boundaries are fully interconnected to form a graph of segments. Over- and under-generation are compromised by varying the threshold.

Unfortunately, segmentation depends on many constraints that cannot be captured by a simple local measure of spectral change. For example, transitions between vowels and consonants may correspond to large acoustic discontinuities, and thus be reliably detected. However, transitions between vowels and semivowels may instead be gradual. As a result, some segments may not be detected by such a simple acoustic algorithm.

Furthermore, an acoustic segmentation algorithm limits the type of subword units that can be segmented and therefore the modeling strategies that may be explored. Because the acoustic segmentation algorithm hypothesizes segment boundaries at points of spectral change, the units must also correspond to spectral discontinuities.

For example, the transition between stop closures and releases are often defined by sharp discontinuities and therefore a segment boundary is typically placed between them. However, studies have shown that there is a high correlation between the closure and burst regions of stops. Therefore, stops could be better modeled as a single unit [25].

n-Best Resorting

Another approach that is used to explore alternative segment sequences is n-best resorting [1, 45]. In this paradigm, a less complex system can be used to generate an n-best list for rescoring by a more complex system. Just as a segmentation algorithm produces a pruned segment graph, an n-best search can generate a pruned segment graph for the subsequent search. However, an n-best list is an inefficient representation, since there is typically a significant degree of overlap between different paths.

Furthermore, although the n-best paradigm may be an effective formalism for integrating diverse recognition strategies, the n-best paradigm does not by itself provide a framework for segment-based recognition [45]. In order to rescore an n-best path using segment-based feature vectors, it is still necessary to account for all segments in all of the n-best paths.

2.5.4 Segment-Based Modeling

Although difficult, the development of a segment-based approach has significant rewards. First, a segment-based approach offers the flexibility to explore the relative advantages of both frame- and segment-based approaches. In fact, it is likely that the best approach will be a combination of the relative advantages of these two approaches [31, 67]. The frame-based approach can provide greater efficiency, while the segment-based approach provides more powerful modeling.

In addition, in a segment-based approach, a unit can be represented by a single feature vector, so that a segment-based model can capture constraints that are extracted with respect to the entire segment [5, 69]. These constraints may be as simple as duration, but can also include more complex modeling strategies where feature vectors are extracted based on their relationship with the entire segment [25]. In fact, it is likely that speech recognition can be improved by combining our knowledge of speech and the use of automatic training techniques [25, 67].

2.6 Summary

Both frame- and segment-based approaches to speech recognition use the same probabilistic framework, modeling strategies and search algorithms. However, the choice of an approach trades off advantages and disadvantages in modeling and search. The dominant HMM frame-based approach has capitalized on the advantages of an efficient search but may be limited by its inability to model segment-based feature vectors. In contrast, a segment-based approach has the potential to improve modeling but faces difficulties in search and segmentation.

Chapter 3

Experimental Framework

The evaluation in this thesis focuses on the acoustic-phonetic level where near-miss modeling has the greatest impact. This chapter describes the framework for the experiments in phonetic recognition that are reported in the Chapters 4 and 5. Additional experiments in word recognition are described in Chapter 6. The following two sections describe the corpus and the recognizers used in phonetic recognition.

3.1 TIMIT Corpus

To facilitate comparison with other approaches, experiments in phonetic recognition are performed on the commonly used TIMIT corpus [16, 20, 30]. TIMIT is a corpus of read, continuous speech that has been phonetically and orthographically time-aligned and transcribed. The following three sections describe the sets used in training and testing, the phones used in transcription, and the classes used in reporting results.

3.1.1 TIMIT Sets

TIMIT contains 6300 utterances read by 630 speakers [20]. The speakers are 70% male and 30% female and are grouped into 8 major dialect regions of American English. Each speaker read 10 utterances, including 2 "sa" dialect utterances designed to demonstrate dialectical differences, 5 "sx" phonemically compact utterances designed to cover all phoneme pairs, and 3 "si" phonetically diverse utterances designed to add phonetic contexts. There are a total of 2342 sentences, including 2 "sa" sentences each read by all 630 speakers, 450 "sx" sentences each read by 7 speakers, and 1890 "si" sentences each read by only 1 speaker.

NIST has divided the "sx" and "si" data into independent training and test sets that do not overlap either by speaker or by sentence [16, 20, 30]. The core test set contains 192 "sx" and "si" utterances read by 24 speakers, including 2 male and 1 female from each dialect region. The complete test set contains a total of 1344 "sx" and "si" utterances read by the 168 speakers who read any sentence in the core test set. The training set contains the remaining 3696 "sx" and "si" utterances read by the remaining 462 speakers.

All of TIMIT results in this thesis are reported on the NIST core test set, and all of the TIMIT models are trained on the NIST training set. To avoid biasing the results to the core test set, all intermediate experiments are run on a development set containing 400 utterances read by 50 speakers drawn from the complete test set minus the core test set. Table 3.1 shows the number of speakers, utterances, and phones in the core test, training, and development sets:

Set	# Speaker	# Utterance	# Phone
Core Test	24	192	7,333
Train	462	3,696	142,910
Development	50	400	15,334

Table 3.1: The number of speakers, utterances, and phones in the test, training, and development sets in TIMIT.

3.1.2 TIMIT Phones

TIMIT was phonetically transcribed using a set of 61 phones [30]. Table 3.2 shows these phones along with their corresponding IPA symbols and example sounds as indicated by the italicized letters in the example words.

TIMIT	IPA	Example	TIMIT	IPA	Example
aa	ɑ	b*o*ttle	ix	ɨ	deb*i*t
ae	æ	b*a*t	iy	i	b*ee*t
ah	ʌ	b*u*t	jh	ǰ	*j*oke
ao	ɔ	b*ou*ght	k	k	*k*ey
aw	ɑʷ	ab*ou*t	kcl	k□	k closure
ax	ə	*a*bout	l	l	*l*ay
ax-h	əʰ	s*u*spect	m	m	*m*o*m*
axr	ɚ	butt*er*	n	n	*n*oo*n*
ay	ɑʸ	b*i*te	ng	ŋ	si*ng*
b	b	*b*ee	nx	ɾ̃	wi*nn*er
bcl	b□	b closure	ow	o	b*oa*t
ch	č	*ch*oke	oy	ɔʸ	b*oy*
d	d	*d*ay	p	p	*p*ea
dcl	d□	d closure	pau	□	pause
dh	ð	*th*en	pcl	p□	p closure
dx	ɾ	bu*tt*er	q	ʔ	co*tt*on
eh	ɛ	b*e*t	r	r	*r*ay
el	l̩	bott*le*	s	s	*s*ea
em	m̩	bott*om*	sh	š	*sh*e
en	n̩	butt*on*	t	t	*t*ea
eng	ŋ̍	Wash*ing*ton	tcl	t□	t closure
epi	□	epenthetic silence	th	θ	*th*in
er	ɝ	b*ir*d	uh	ʊ	b*oo*k
ey	e	b*ai*t	uw	u	b*oo*t
f	f	*f*in	ux	ü	t*oo*t
g	g	*g*ay	v	v	*v*an
gcl	g□	g closure	w	w	*w*ay
hh	h	*h*ay	y	y	*y*acht
hv	ɦ	a*h*ead	z	z	*z*one
ih	ɪ	b*i*t	zh	ž	a*z*ure
h#	-	utterance initial and final silence			

Table 3.2: The set of 61 phones used in transcribing TIMIT along with their corresponding IPA symbols and example sounds as indicated by the italicized letters in the example words.

3.1.3 TIMIT Classes

To facilitate comparison with other approaches, this thesis reports all results in phonetic recognition on TIMIT over the set of 39 classes that are commonly used for such evaluation [32]. Table 3.3 shows these classes:

Class	Class
aa ao	k
ae	l el
ah ax ax-h	m em
aw	n en nx
ay	ng eng
b	ow
ch	oy
d	p
dh	r
dx	s
eh	sh zh
er axr	t
ey	th
f	uh
g	uw ux
hh hv	v
ih ix	w
iy	y
jh	z
bcl dcl gcl kcl pcl tcl epi q pau h#	

Table 3.3: The set of 39 classes that are used for reporting results in phonetic recognition on TIMIT.

All phonetic recognition error rates are computed using the NIST alignment program [16]. This program finds the minimum cost alignment, where the cost of a substitution is 1.0, and the cost of a deletion or an insertion is 0.75. The total recognition error rate is the sum of the substitution, deletion, and insertion rates.

3.2 Phonetic Recognizers

To perform experiments in TIMIT, phonetic recognizers are built using the SAPPHIRE speech analysis and recognition toolkit [28]. The goal of this thesis is to improve the use of the models within a recognition framework rather than the models themselves. As a result, the recognizers use commonly used models that are comparable to those used in other systems. This section describes the phonetic recognizer components.

3.2.1 Acoustic Model

The following four sections describe the acoustic model in detail.

Representation

All of the phonetic recognizers initially transform the speech signal to the same cepstral representation. The speech signal is sampled at 16 kHz, analyzed at a 10 ms analysis rate with a 20 ms Hamming window, and transformed into the frequency domain using a 256 point Fourier transform. The frequency samples are then compressed into 40 Mel-Frequency Spectral Coefficients (MFSCs) using a bank of triangular filters spaced on a Mel-frequency scale that approximates an auditory scale [12, 38]. The first 12 Mel-Frequency Cepstral Coefficients (MFCCs) are computed from the MFSCs using a cosine transform. For each utterance, the MFCCs are normalized by subtracting the mean of the MFCC vectors across the utterance.

Feature Extraction

From the cepstral representation, the phonetic recognizers extract different features depending on whether they are segment- or frame-based:

- The segment-based features consist of three averages over non-overlapping time spans across the segment, two averages over time spans before and after the segment, and the logarithm of the duration of the segment, for a total of 61 dimensions [8, 25, 40]. The time spans of the intra-segmental averages are a function of the segment duration, and are computed with a 3-4-3 ratio across the segment, while the time span of the extra-segmental averages is 30 ms independent of the segment duration.

- The frame-based features consist of six average cepstral vectors computed over non-overlapping time spans before and after the frame for a total of 72 dimensions [37, 33]. The time spans are symmetric about the frame, widening from 10 ms to 20 ms to 40ms.

Classification

All of the features, whether they are segment- or frame-based, are modeled using mixture of diagonal covariance Gaussian distributions. Each feature is first diagonalized by a principal components rotation computed over the training set [15, 21]. For each model, a maximum of 100 mixtures with a minimum of 10 tokens is computed using k-means clustering followed by EM training [15, 2, 54].

Unit Selection

The units that are modeled vary between the segment- and frame-based models:

- The segment-based units are context-independent. They include all 61 TIMIT phones plus some additional units that will be described in the following chapter.

- The frame-based units are diphone context-dependent. They include all 1505 TIMIT diphones that have at least 10 tokens in the NIST training data plus one unit to cover all remaining diphones for a total of 1506 units.

3.2.2 Other Models

Phonetic recognition does not involve a pronunciation model. All of the phonetic recognizers use the same duration and language models:

- The duration model is a phonetic transition weight that is multiplied in at each phonetic transition [67]. The weight is set by minimizing recognition error on the development set.

- The language model is a bigram [2, 54]. The vocabulary contains all 61 TIMIT phones. The bigram is trained on the NIST training set, which covers all 61 phones so there are no out-of-vocabulary (OOV) phones. The bigram is smoothed with a unigram, and the smoothing parameters are set by minimizing perplexity on the development set. The resulting bigram model has a perplexity of 15.8 on the core test set, including utterance initial and final silences. In testing, the bigram model is exponentially weighted, with the weight being set by minimizing recognition error on the development set.

3.3 Summary

The evaluation in this thesis focuses on experiments in phonetic recognition using the TIMIT corpus. This chapter has described the framework for the experiments in phonetic recognition that will be reported in the following two chapters. All results in phonetic recognition on TIMIT are reported on the core test set over 39 classes. All of the phonetic recognizers use mixture of diagonal Gaussian acoustic models, a phonetic transition weight, and a phone bigram language model.

Chapter 4

Near-Miss Search

This chapter describes a new segment-based approach that generalizes anti-phone modeling to allow for more complex modeling of off-path segments. The approach is called near-miss modeling, since it is based on the idea that a segment that overlaps with another segment can be thought of as a "near-miss" of that segment. For any segment graph, near-miss modeling associates each segment with a near-miss subset of feature vectors such that the near-miss subsets that are associated with any path account for all feature vectors. As a result, near-miss modeling can score the entire near-miss subset of feature vectors that are associated to the on-path segments, and provides a more general framework for segment-based recognition that allows the off-path segments to be modeled with a wide range of contextual units. In the case when no context is used, near-miss modeling is reduced to anti-phone modeling. However, near-miss modeling can also be extended to model the context of the off-path segments. The next section provides a detailed example of near-miss modeling. The following two sections describe the near-miss assignment algorithm for assigning the near-miss subsets and the near-miss framework for speech recognition. The next two sections explore some methods for determining near-miss subsets and modeling near-miss units. Finally, the chapter concludes with an evaluation of near-miss modeling for the task of phonetic recognition.

4.1 Near-Miss Example

This section describes how near-miss modeling applies to the example in Figure 4-1, which is the same as Figure 2-6.

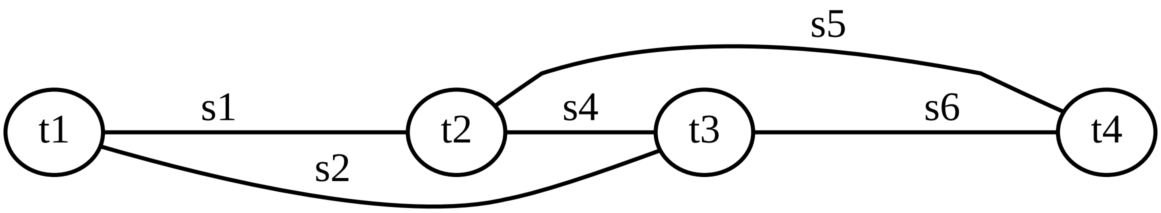


Figure 4-1: Example graph.

Each segment, s_i, is associated with a feature vector, a_i. In near-miss modeling, the goal is to extend this association so that each segment is associated with a near-miss *subset* of feature vectors, A_i, which includes not only a_i, but also zero or more other feature vectors. Overall, this association must satisfy the constraint that for every possible segment sequence through the graph, the union of the near-miss subsets of each segment in the segment sequence is exactly equal to the entire set of feature vectors.

4.1.1 Features to Subsets

To achieve this goal, each feature vector, a_i, is assigned to its own near-miss subset, A_i, and zero or more other near-miss subsets. For each segment sequence, S, the feature vector, a_i, must be assigned to one segment in the sequence. For segment sequences that contain the segment, the feature vector is already assigned to a near-miss subset in the sequence and cannot be assigned to any other near-miss subset in the sequence. For segment sequences that do not contain the segment, the feature vector must be assigned to one and only one near-miss subset in the sequence. These constraints can be used to reason about the assignments in Figure 4-1:

- Since s_1 is in S_{146} and S_{15}, a_1 must not be assigned to A_4, A_5, or A_6. Since s_1 is not in S_{26}, a_1 must be assigned to either A_2 or A_6. As a result, a_1 must be assigned to A_1 and A_2.

- Since s_2 is in S_{26}, a_2 must not be assigned to A_6. Since s_2 is not in S_{146} or S_{15}, a_2 must be assigned to either A_1 or A_4 and either A_1 or A_5. As a result, a_2 must be assigned to A_2 and either A_1 or both A_4 and A_5.

- Since s_4 is in S_{146}, a_4 must not be assigned to A_1 or A_6. Since s_4 is not in S_{15} or S_{26}, a_4 must be assigned to A_5 and A_2. As a result, a_4 must be assigned to A_4, A_2 and A_5.

- Since s_5 is in S_{15}, a_5 must not be assigned to A_1. Since s_5 is not in S_{146} or S_{26}, a_5 must be assigned to either A_4 or A_6 and either A_2 or A_6. As a result, a_5 is assigned to A_5 and either A_6 or A_2 and A_4.

- Since s_6 is in S_{146} and S_{26}, a_6 must not be assigned to A_1, A_2, or A_4. Since s_6 is not in S_{15}, a_6 must be assigned to A_5. As a result, a_6 must be assigned to A_6 and A_5.

Table 4.1 summarizes the solutions to the constraints. In assigning feature vectors to near-miss subsets, each feature vector, a_i, is assigned to its own near-miss subset, A_i, and zero or more additional near-miss subsets. In this case, a_1, a_4, and a_6 each have only one option for assignment, while a_2 and a_5 each have two options for assignment.

Option	a_1	a_2	a_4	a_5	a_6
1	A_1 A_2	A_2 A_1	A_4 A_2 A_5	A_5 A_6	A_6 A_5
2	-	A_2 A_4 A_5	-	A_5 A_2 A_4	-

Table 4.1: Assignment of feature vectors, a_i, to near-miss subsets, A_i, for Figure 4-1. a_1, a_4, and a_6 each have only one option for assignment, while a_2 and a_5 each have two options for assignment.

4.1.2 Subsets to Features

The assignment of feature vectors to near-miss subsets can be inverted to determine the possible near-miss subsets and their feature vectors. Since two of the feature vectors have two options, there are four different ways of drawing near-miss subsets as shown in Table 4.2. Each near-miss subset, A_i, contains its associated feature vector, a_i, and zero or more additional feature vectors.

Option	A_1	A_2	A_4	A_5	A_6
1	a_1 a_2	a_2 a_1 a_4	a_4	a_5 a_4 a_6	a_6 a_5
2	a_1 a_2	a_2 a_1 a_4 a_5	a_4 a_5	a_5 a_4 a_6	a_6
3	a_1	a_2 a_1 a_4	a_4 a_2	a_5 a_2 a_4 a_6	a_6 a_5
4	a_1	a_2 a_1 a_4 a_5	a_4 a_2 a_5	a_5 a_2 a_4 a_6	a_6

Table 4.2: Possible near-miss subsets, A_i, and their feature vectors, a_i, from Table 4.1.

The following three sections introduce useful ways of visualizing near-miss modeling.

By Graph

One way of visualizing the solutions in Table 4.2 is by labeling each arc in the graph with its near-miss subset. Table 4.3 shows the four options by graph. These graphs are useful in that they clearly show that each sequence of segments through the graph accounts for all segments in the graph once and only once.

By Subset

Another way of visualizing the solutions in Table 4.2 is by drawing each near-miss subset. Table 4.4 shows the four possible solutions by near-miss subset. To allow future differentiation of near-miss subsets, the segments within each near-miss subset are drawn using a consistent line style. In particular, the first and second near-miss subsets are drawn with solid lines, the fourth and fifth near-miss subsets are drawn with dashed lines, and the sixth near-miss subset is drawn with dotted lines. For each near-miss subset, A_i, the arc labeled a_i corresponds to the on-path feature

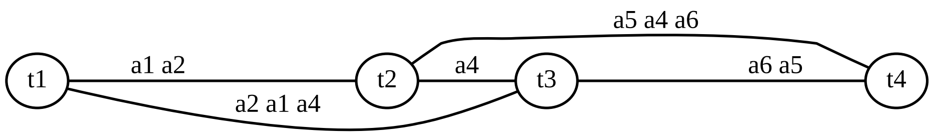
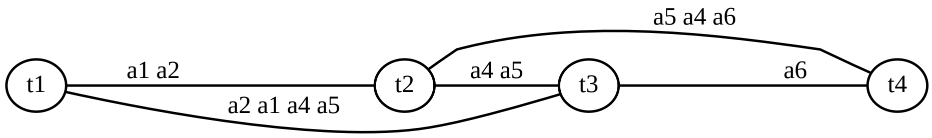
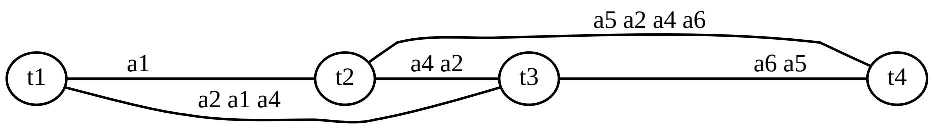
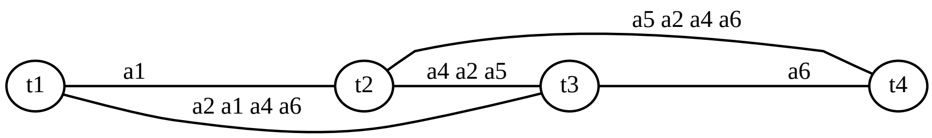
<table>
<tr><th>Option</th><th>Graph</th></tr>
<tr><td>1</td><td>
</td></tr>
<tr><td>2</td><td>
</td></tr>
<tr><td>3</td><td>
</td></tr>
<tr><td>4</td><td>
</td></tr>
</table>

Table 4.3: Options for near-miss subsets in Table 4.2 shown by graph.

vector, and the remaining feature vectors are not on the path. These graphs are useful for showing how each segment is associated with a near-miss subgraph of the entire segment graph.

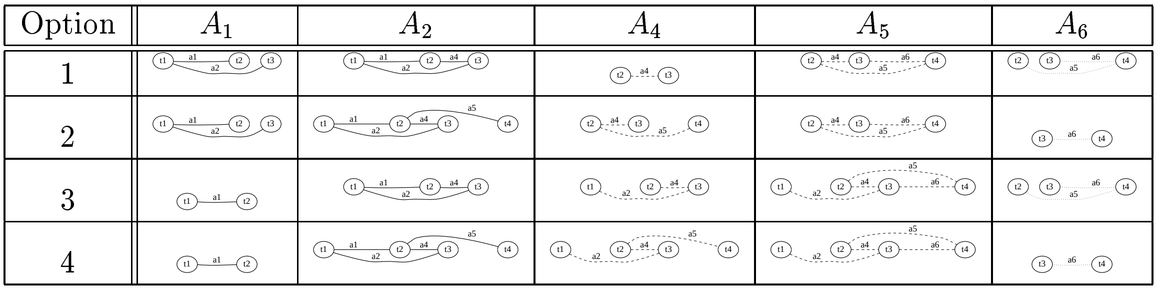


Table 4.4: Near-miss subsets in Table 4.2. Each near-miss subset is drawn using a consistent line style to allow future differentiation of near-miss subsets.

By Sequence

A third way of visualizing Table 4.2 is to concatenate the near-miss subsets in Table 4.4 by segment sequence. Table 4.5 shows the four possible solutions by sequence. For each sequence, S_{ijk}, the near-miss subsets A_i, A_j, and A_k are concatenated, and the feature vectors, a_i, a_j, and a_k correspond to the on-path feature vectors. As shown, the different line styles allow the visual differentiation of near-miss subsets. These graphs are useful for showing how each segment sequence accounts for all segments in the graph.

4.2 Near-Miss Assignment

To be useful for speech recognition, near-miss modeling should be applicable to any possible segment graph. For any segment graph, it must be possible to compute an assignment of segments to near-miss subsets such that every segment sequence accounts for all of the feature vectors in the graph. This section provides an existence proof using an algorithm that can compute near-miss subsets for any graph. A segment, s_i, is defined to span the interval, $[b_i, e_i)$, from and including its begin time,

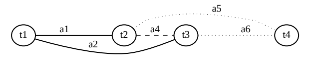
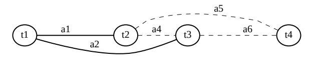
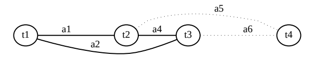
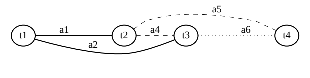
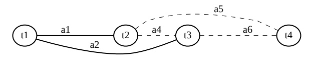
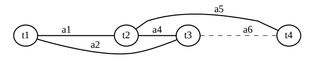
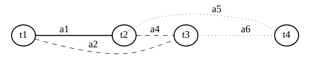
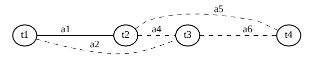
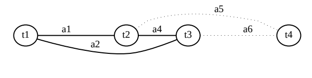
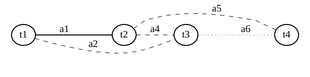
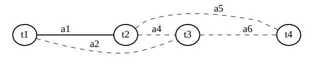
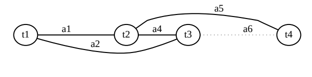
Option	S_{146}	S_{15}	S_{26}
1			
2			
3			
4			

Table 4.5: Near-miss subsets in Table 4.4 concatenated by segment sequence.

b_i, to and excluding its end time, e_i. For each segment, s_i, select any time, t_i, in the span of segment, $[b_i, e_i)$. Then, for each segment, s_j, if the span $[b_j, e_j)$, includes the selected time, t_i, add the feature vector, a_i, to the near-miss subset, A_j. Note that a feature vector, a_i, that is associated with a segment, s_i, is always assigned to its own near-miss subset, A_i. Figure 4-2 is pseudo-code for the near-miss assignment algorithm.

for s_i in $\{s_i\}$:
  $A_i = \{\}$
for s_i in $\{s_i\}$:
  choose t_i in $[b_i, e_i)$
  for s_j in $\{s_j\}$:
    if t_i in $[b_j, e_j)$ **then**
      $A_j = a_i \bigcup A_j$

Figure 4-2: A segment, s_i, is defined to span the interval, $[b_i, e_i)$, from and including its start time, b_i, to and excluding its end time, e_i. The near-miss assignment algorithm assigns each feature vector, a_i, to near-miss subsets by selecting a time, t_i, that is spanned by a_i, and assigning a_i to each near-miss subset, A_j, that also spans the selected time, t_i.

The proof that a near-miss assignment exists for all segment graphs is based on the fact that any segment sequence accounts for all times. Since any segment sequence spans any time exactly once, the near-miss assignment algorithm will necessarily assign each segment that is not in the segment sequence to one and exactly one of the segments that is in the segment sequence. Therefore, the near-miss assignment algorithm can compute a suitable assignment for any segment graph and can be used to develop a general search strategy. Note that the proof does not require the time that is chosen for each segment to fall within the segment. However, if the time that is chosen for each segment does not fall within its span, the segment will not be assigned to its own near-miss subset. For the application to speech recognition in this thesis, it seems more reasonable to account for each segment as it is traversed during the search. In other applications of near-miss modeling, it may be more appropriate to take advantage of this generalization.

Note also that the near-miss assignment algorithm can easily accommodate frame-based feature vectors. Each frame-based feature vector can be associated with a single time and assigned to near-miss subsets based on that time. This results in each segment being associated with the frame-based feature vectors that it spans.

4.3 Near-Miss Framework

This section describes how the near-miss assignment algorithm can extend the general probabilistic framework for speech recognition to a segment-based approach. In graph-based representation, each segment is associated with a feature vector, a_i. The near-miss assignment algorithm extends the association of each segment to a near-miss subset, A_i, that, in addition to a_i, can also contain zero or more other feature vectors such that the union of the near-miss subsets that are associated with any segment sequence is the entire set of feature vectors:

$$A = \bigcup_{s_i \in S} A_i$$

As a result, near-miss modeling can compute the score for a path by accounting for all of the near-miss subsets on the path. Assuming segmental independence, each segment-based feature vector, a_j, in the near-miss subset, A_i, is independent of the other feature vectors in the near-miss subset:

$$\begin{aligned} P(A|US) &= \prod_{s_i \in S} P(A_i|US) \\ &= \prod_{s_i \in S} \prod_{a_j \in A_i} P(a_j|US) \end{aligned}$$

Note that segments in speech are certainly not independent of each other. For example, different segments of speech spoken by the same speaker are correlated. However, the assumption of segmental independence is widely used in speech recognition to allow a more efficient search. In near-miss modeling, it is not necessary to assume independence between all of the segments within a near-miss subset. For example, it is possible to identify feature vectors that always appear together in the near-miss subsets and model them jointly. However, such strategies are not explored in this thesis.

The near-miss modeling framework can be implemented in a time-synchronous Viterbi search. When updating each segment, s_i, the near-miss search scores all of the feature vectors in its near-miss subset, A_i. Since the scoring directly accounts for all segments, all segments can be modeled using any number of units. For Option 1 in Table 4.5, the three segment sequences can be scored, one near-miss subset at a time:

$$\begin{aligned} P(A|US_{146}) &= [P(a_1|US_{146})P(a_2|US_{146})] * [P(a_4|US_{146})] \\ &\quad *[P(a_5|US_{146})P(a_6|US_{146})] \\ P(A|US_{15}) &= [P(a_1|US_{15})P(a_2|US_{15})] * [P(a_4|US_{15})P(a_5|US_{15})P(a_6|US_{15})] \\ P(A|US_{26}) &= [P(a_1|US_{26})P(a_2|US_{26})P(a_4|US_{26})] * [P(a_5|US_{26})P(a_5|US_{26})] \end{aligned}$$

In this example, the first and second feature vectors, a_1 and a_2, and the fifth and sixth feature vectors, a_5 and a_6, always appear together and could be modeled jointly. The following two sections focus on particular strategies for assigning of the near-miss subsets and the modeling of the units.

4.4 Near-Miss Subsets

The general near-miss assignment algorithm allows each segment to be assigned to near-miss subsets based on any time within the segment. Different choices of times may result in different near-miss subsets. Each segment must be assigned to the near-miss subset of any segments by which it is completely overlapped. For example, each segment is always assigned to its own near-miss subset. However, each segment has some degree of freedom over whether to be assigned to the near-miss subsets of a segment by which it is only partially overlapped. This degree of freedom is expressed as the selection of a single time within the segment. For example, the options in Table 4.2 can be quickly derived by looking at the overlapping segments in Figure 4-1: a_1, a_4, and a_6 must be assigned to the near-miss subsets by which they are completely overlapped. However, a_2 and a_5 are each partially overlapped by two segments and have two options.

For ease of implementation, this thesis explores the space of strategies in which all segments are assigned based on the same relative time within the segment, ranging from its begin time to its end time. Varying the relative segment time produces a wide range of near-miss assignments. In Figure 4.3, three of the four options are contained within the space of assigning all segments based on the same relative time:

- Option 1 can be achieved by assigning based on the midpoint of each segment.
- Option 2 can be achieved by assigning based on the begin time of each segment.
- Option 3 can be achieved by assigning based on the end time of each segment.
- Option 4 cannot be achieved by using the same relative time across segments.

4.5 Near-Miss Units

The motivation for developing near-miss modeling is to enable the modeling of context across all segments in the graph. In general, near-miss modeling can use any number of additional near-miss units that depend on any contextual information extracted based on segment times or unit labels. In choosing a modeling strategy, this thesis limits its exploration to strategies that model the on-path segment as a lexical unit and model an off-path segment as a near-miss unit. Such strategies are based on the idea that on-path segments are positive examples of units while off-path segments are negative examples of units. By modeling both positive and negative examples, near-miss modeling may be better able to distinguish between alternatives in a segment graph.

Note that it is not necessary to assume a difference between on- and off-path segments. In fact, near-miss modeling suggests a more sophisticated representation of the speech signal as a graph rather than a flat sequence of units. However, for practical reasons, this thesis restricts itself to a limited space of strategies and reserves these remaining ideas for future work.

The following sections describe three particular types of near-miss units: a single 0-state unit that does not depend on context, a set of 1-state units that depend on the phonetic context of the on-path segment, and various sets of multi-state units that depend on the temporal alignment of the on-path segment in addition to its phonetic context. The selection of these additional near-miss units shares issues with the selection of context-dependent phones. Both modeling strategies tend to increase computation and divide the training data. To improve robustness, more specific near-miss models can be smoothed with more general near-miss models.

4.5.1 0-State Unit

The simplest near-miss unit is a 0-state unit, denoted as $\bar{u}$. In the 0-state strategy, all off-path segments are modeled by the 0-state unit. Table 4.6 shows the use of the 0-state unit by near-miss subset for Option 1 in Table 4.4. For each segment, s_i, the feature vector that is associated with the segment is modeled using the lexical unit, u_i, while the remaining off-path feature vectors in its near-miss subset are modeled using the same 0-state unit, $\bar{u}$.

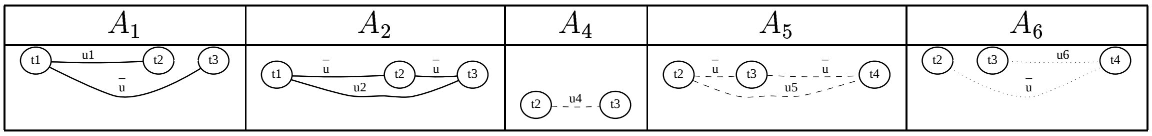


Table 4.6: For Option 1 in Table 4.4, the 0-state strategy models the segment using the lexical unit, u_i, and models the remaining off-paths segments in the near-miss subset using a single 0-state unit, $\bar{u}$.

Table 4.7 shows the use of the 0-state unit by segment sequence for Option 1 in Table 4.5. The feature vectors on the path are scored against the lexical units, while the remaining off-path feature vectors are scored against the 0-state unit.

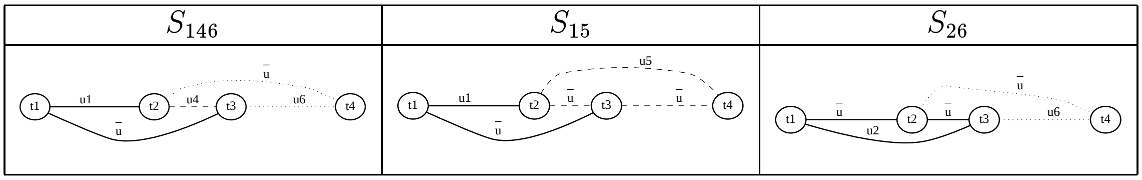


Table 4.7: For Option 1 in Table 4.5, S, the 0-state strategy models each segment on the segment sequence using a lexical unit, u_i, and models all remaining off-path segments using a single 0-state unit, $\bar{u}$.

Function

The use of the 0-state unit in near-miss modeling is functionally equivalent to the use of the anti-phone unit in anti-phone modeling [22]. Since near-miss modeling and anti-phone modeling maintain the same probabilistic framework and use the same modeling strategy, they will find the same best path, as long as they search the same space. Further, the two strategies find the same best path, even with pruning, as long as they use the same pruning threshold. This is due to the fact that near-miss modeling, anti-phone modeling, and pruning are all implemented in a time-synchronous manner. At a corresponding time point in the near-miss and anti-phone searches, the same paths have the same scores, offset by a value that is equal to the product of the 0-state scores of the segments that have been accounted for in the near-miss search up to that time point.

Computation

Since they search the same paths, 0-state near-miss modeling and anti-phone modeling score the same on-path segments against the same lexical units. However, the two strategies can differ in which off-path segments they score against the 0-state or anti-phone units. For each segment sequence, near-miss modeling scores all of the off-path segments against the near-miss unit, while anti-phone modeling scores only the on-path segments against the anti-phone. As a result, near-miss modeling scores every segment that is a near-miss segment of at least one segment other than itself, while anti-phone modeling scores every segment that is explored. The following three cases compare the two approaches:

- Near-miss modeling scores fewer segments then anti-phone modeling. This occurs when a segment is on every segment sequence and therefore is not in the near-miss subset of any segment except itself. In general, near-miss modeling scores one less segment against the 0-state unit for each segment that is on every segment sequence. In the limit, when the segment graph is a single sequence of segments, near-miss modeling does not score any segment against the 0-state unit, while anti-phone modeling scores all segments against the anti-phone unit.

- Near-miss modeling scores more segments than anti-phone modeling. This occurs when a segment is not explored. Typically, all segments are explored, even with pruning, since pruning allows at least one path to survive at any time point. However, segments may not be explored in the rare cases when prior to the end of the utterance, the only surviving paths end in final nodes that cannot be extended. For example, in Figure 4-1, if both paths S_{146} and S_{26} cannot be continued after the third time, t_3, then the sixth segment, s_6, is not explored. In this case, near-miss modeling scores s_6 in order to score S_{15}, but anti-phone modeling does not. In general, near-miss modeling scores one more segment against the 0-state unit for each segment that is not explored.

- If neither of the above cases occur, near-miss modeling and anti-phone modeling score the same number of segments. For example, to score all paths in Figure 4-1, near-miss modeling computes:

$$\begin{aligned}
P(A|US_{146}) &= [P(a_1|u_1)P(a_2|\bar{u})] * [P(a_4|u_4)] * [P(a_5|\bar{u})P(a_6|u_6)] \\
P(A|US_{15}) &= [P(a_1|u_1)P(a_2|\bar{u})] * [P(a_4|\bar{u})P(a_5|u_5)P(a_6|\bar{u})] \\
P(A|US_{26}) &= [P(a_1|\bar{u})P(a_2|u_2)P(a_4|\bar{u})] * [P(a_5|\bar{u})P(a_6|u_6)]
\end{aligned}$$

  In contrast, anti-phone modeling computes:

$$\begin{aligned}
P(A|US_{146}) &= \frac{P(a_1|u_1)}{P(a_1|\bar{u})}\frac{P(a_4|u_4)}{P(a_4|\bar{u})}\frac{P(a_6|u_6)}{P(a_6|\bar{u})} \\
P(A|US_{15}) &= \frac{P(a_1|u_1)}{P(a_1|\bar{u})}\frac{P(a_5|u_5)}{P(a_5|\bar{u})} \\
P(A|US_{26}) &= \frac{P(a_2|u_2)}{P(a_2|\bar{u})}\frac{P(a_6|u_6)}{P(a_6|\bar{u})}
\end{aligned}$$

  Although the ordering may be different, both approaches eventually score all segments against the 0-state or anti-phone model.

Although these differences are algorithmically interesting, they are practically negligible, since the scoring of the single near-miss or anti-phone unit is far outweighed by the scoring of the multiple lexical units. Therefore, near-miss modeling and anti-phone modeling are effectively equivalent both at functional and computational levels. The important point is that near-miss modeling allows more complex modeling strategies without sacrificing computational efficiency when using simpler models.

4.5.2 1-State Unit

This section introduces a 1-state unit that depends on the context of the on-path segment. Each lexical unit, u_i, has a corresponding 1-state near-miss unit, $\bar{u}_i$. When a segment, s_i, is modeled as u_i, the remaining off-path segments in its near-miss subset are modeled as $\bar{u}_i$. Table 4.8 shows the use of 1-state units by near-miss subset for Option 1 in Table 4.4. For each segment, s_i, the feature vector that is associated with the segment is modeled using the lexical unit, u_i, while the remaining off-path feature vectors in the near-miss subset are modeled using the corresponding 1-state unit, $\bar{u}_i$.

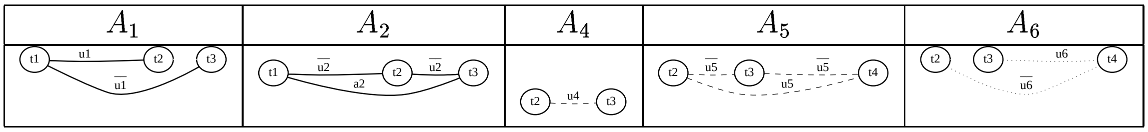


Table 4.8: For Option 1 in Table 4.4, the 1-state strategy models the on-path segment using the lexical unit, u_i, and models the remaining off-path segments in the near-miss subset using the corresponding 1-state unit, $\bar{u}_i$.

Table 4.9 shows the use of the 1-state units by segment sequence for Option 1 in Table 4.5. Each feature vector on the path is scored against a lexical unit, while the remaining off-path feature vectors in its near-miss subset are scored against the corresponding 1-state unit.

In comparison to the 0-state unit, the 1-state unit can capture contextual dependencies between a near-miss segment and the segment for which it is a near-miss. Furthermore, since a near-miss segment always overlaps with the segment for which it is a near-miss, these segments should share similar 1-state characteristics in frequency and energy. Computationally, 1-state units directly increase computation over the 0-state unit. Rather than classifying each segment against a single 0-state unit, the 1-state strategy classifies each segment against multiple 1-state units.

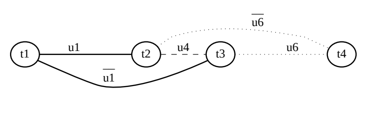
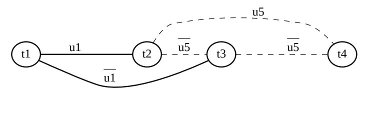
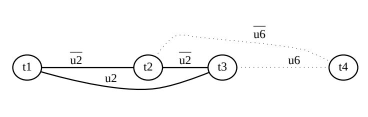
S_{146}	S_{15}	S_{26}
		

Table 4.9: For Option 1 in Table 4.5, the 1-state strategy models each segment on the segment sequence using a lexical unit, u_i, and models the remaining off-path segments in its near-miss subset using the corresponding 1-state unit, $\bar{u}_i$.

4.5.3 Multi-State Unit

This section introduces a third type of multi-state unit that depends on the temporal alignment of the on-path segment in addition to its context.

2-State Unit

In the 2-state strategy, each lexical unit, u_i, has two corresponding near-miss units, $\bar{u}_{i1}$ and $\bar{u}_{i2}$. When a segment, s_i, is modeled as u_i, each remaining off-path segment in its near-miss subset is modeled as either $\bar{u}_{i1}$ or $\bar{u}_{i2}$ depending on whether the midpoint of the off-path segment falls in the first or second half of s_i. Table 4.10 shows the use of the 2-state units by near-miss subset for Option 1 in Table 4.4. For each segment, s_i, the feature vector that is associated with the segment is modeled using the lexical unit, u_i, while each remaining off-path feature vector in its near-miss subset is modeled using $\bar{u}_{i1}$ if the midpoint of the off-path segment falls in the first half of s_i or $\bar{u}_{i2}$ otherwise.

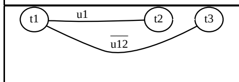
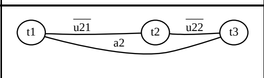
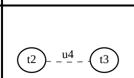
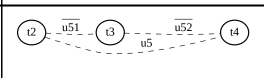
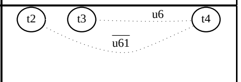
A_1	A_2	A_4	A_5	A_6
				

Table 4.10: For Option 1 in Table 4.4, the 2-state strategy models the segment using the lexical unit, u_i, and models each remaining off-path segment in the near-miss subset using $\bar{u}_{i1}$ if the midpoint of the off-path segment falls in the first half of s_i and $\bar{u}_{i2}$ if the midpoint of the off-path segment falls in the second half of s_i.

Table 4.11 shows the use of the 2-state units by segment sequence for Option 1 in Table 4.5. Each feature vector on a path is scored against the lexical units, while each remaining off-path feature vector in its near-miss subset is scored against $\bar{u}_{i1}$ or $\bar{u}_{i2}$ depending on whether the midpoint of the off-path feature vector falls in the first or second half of the segment.

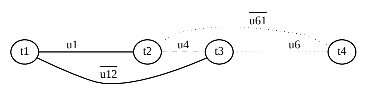
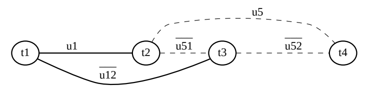
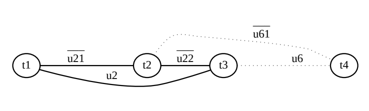
S_{146}	S_{15}	S_{26}
		

Table 4.11: For Option 1 in Table 4.5, the 2-state strategy models each segment on the segment sequence using a lexical unit, u_i, and models each remaining off-path segments in its near-miss subset using $\bar{u}_{i1}$ or $\bar{u}_{i2}$, depending on whether the midpoint of the off-path segment falls in the first or second half of the segment.

In comparison to the 1-state units, the 2-state units can model the phonetic and temporal characteristics of a segment and its near-misses. For example, the near-miss subset of segment s_2 in Table 4.11 suggests that the 2-state units $\bar{u}_{21}$ and $\bar{u}_{22}$ can be used to capture "states" in u_2. If u_2 were the diphthong /ɑ^{y}/, $\bar{u}_{21}$ would capture its first half and may be more /ɑ/-like, while $\bar{u}_{22}$ would capture its second half and may be more /i/-like.

Computationally, multi-state units may not directly increase computation over the 1-state units. Since the midpoint of a segment may fall only in the first halves of other segments, it may not have to be scored against all temporal units. For example, in Table 4.11, the segment s_2, from t_1 to t_3, is scored against $\bar{u}_{12}$ but is not scored against $\bar{u}_{11}$. Similarly, s_1 from t_1 to t_2, s_5 from t_2 to t_3, and s_6 from t_3 to t_4 do not have to be scored against both temporal units. Only the segment s_4, from t_2 to t_3 must be scored against both temporal units, depending on whether the segment sequence is S_{15} or S_{26}. Therefore, expanding context temporally may be an effective strategy for enforcing context without increasing computation.

3-State Unit

In this section, temporal division is extended to 3-state units. Each lexical unit, u_i, has three corresponding near-miss units, $\bar{u}_{i1}$, $\bar{u}_{i2}$, and $\bar{u}_{i3}$. When a segment, s_i, is modeled as u_i, each remaining off-path segment in its near-miss subset is modeled as either $\bar{u}_{i1}$, $\bar{u}_{i2}$ or $\bar{u}_{i3}$ depending on whether the midpoint of the off-path segment falls in the first, second, or last third of s_i. Table 4.12 shows the use of the 3-state units by near-miss subset for Option 1 in Table 4.4.

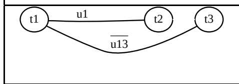
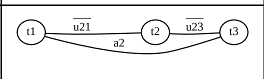
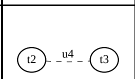
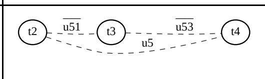
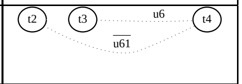
A_1	A_2	A_4	A_5	A_6
				

Table 4.12: For Option 1 in Table 4.4, the 3-state strategy models the segment using the lexical unit, u_i, and models each remaining off-path segments in the near-miss subset using $\bar{u}_{i1}$ if the midpoint of the off-path segment falls in the first third of s_i, $\bar{u}_{i2}$ if the midpoint of the off-path segment falls in the second third of s_i, and $\bar{u}_{i3}$ if the midpoint of the off-path segment falls in the last third of s_i..

Table 4.13 shows the use of the 3-state units by segment sequence for Option 1 in Table 4.5. Each feature vector on a path is scored against the lexical units, while each remaining off-path feature vector in its near-miss subset is scored against $\bar{u}_{i1}$, $\bar{u}_{i2}$, or $\bar{u}_{i3}$ depending on whether the midpoint of the off-path feature vector falls in the first, second, or last third of the segment.

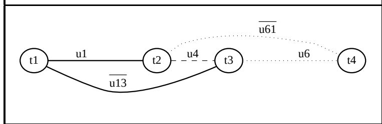
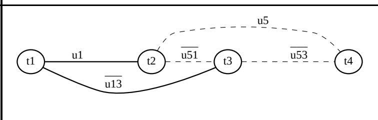
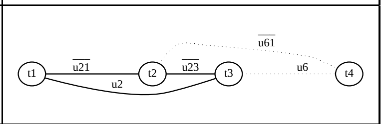
S_{146}	S_{15}	S_{26}
		

Table 4.13: For Option 1 in Table 4.5, S, the 3-state strategy models each segment on the segment sequence using a lexical unit, u_i, and models each remaining off-path segments in its near-miss subset using $\bar{u}_{i1}$, $\bar{u}_{i2}$, or $\bar{u}_{i3}$, depending on whether the midpoint of the off-path segment falls in the first, second, or last third of the segment.

In this case, the use of 3-state units does not increase computation over the 2-state units, since it does not introduce any new temporal ambiguity. For example, in

Table 4.13, s_1 from t_1 to t_2, s_2 from t_1 to t_3, s_5 from t_2 to t_3, and s_6 from t_3 to t_4 are still only scored against one of the temporal divisions, and segment s_4, from t_2 to t_3 is still scored against only two of the temporal divisions. Depending on the topology of the graph, it is possible for higher order units to remove temporal ambiguity and result in less computation.

4.5.4 Frame-Based Unit

In addition to segment-based feature vectors, the near-miss subsets can also contain frame-based feature vectors. These frame-based feature vectors are modeled using frame-based units. In general, the frame-based units can also depend on any contextual information extracted based on segment times or unit labels. Using only frame-based units, near-miss modeling is functionally equivalent to a frame-based approach. The next chapter will explore the use of frame-based units alone. This chapter uses frame-based units in conjunction with the segment-based units already described. In particular, the frame-based units are diphone context-dependent internal or transition models [21, 49]. Each frame-based feature vector is considered to be either internal to a phone or a transition between two phones.

4.6 Near-Miss Evaluation

This section evaluates near-miss modeling on the task of phonetic recognition using the TIMIT corpus. The experimental framework has been described in detail in Chapter 3. In addition, all of the experiments use the same set of segment graphs, which will be described in the following chapter. The experiments are divided into four sections. The first two sections explore strategies for determining near-miss subsets and near-miss units. The following two sections explore additional issues in computation and the use of frame-based units.

4.6.1 Near-Miss Subsets

This section explores near-miss assignment strategies. The goal in near-miss assignment depends on its application. In this thesis, the goal is to maximize the amount of temporal overlap between each off-path segment and the on-path segment to which it is associated. This allows near-miss modeling to focus on the differences between a segment and its closest near-miss competitors. The temporal overlap of a segment is measured with respect to a reference on-path segment and is defined as the percentage of the segment that is overlapped by the reference segment. In order to avoid bias towards a particular segment-based modeling strategy, the reference segment sequence for all of the experiments on near-miss subsets is the best path through the graph found by a frame-based recognizer. The following three sections show examples, temporal overlap statistics, and recognition error rates for different near-miss assignment strategies.

Examples

Before showing statistics, this section gives examples of different near-miss strategies on the same segment graph. Figure 4-3 shows a waveform, spectrogram, segment graphs with near-miss subsets computed using the begin time and midpoint of each segment, the frame-based phone path used to compute near-miss subsets, and the phone and word transcriptions. The segments belonging to the same near-miss subset are drawn using a consistent intensity or gray-level. Three intensities are alternated to allow the visual differentiation of near-miss subsets. The figure shows examples of how near-miss subsets can vary depending on the assignment strategy. Of the two strategies compared, the midpoint strategy in the lower graph seems to provide a larger degree of temporal overlap across segments. The next section quantifies these observations.

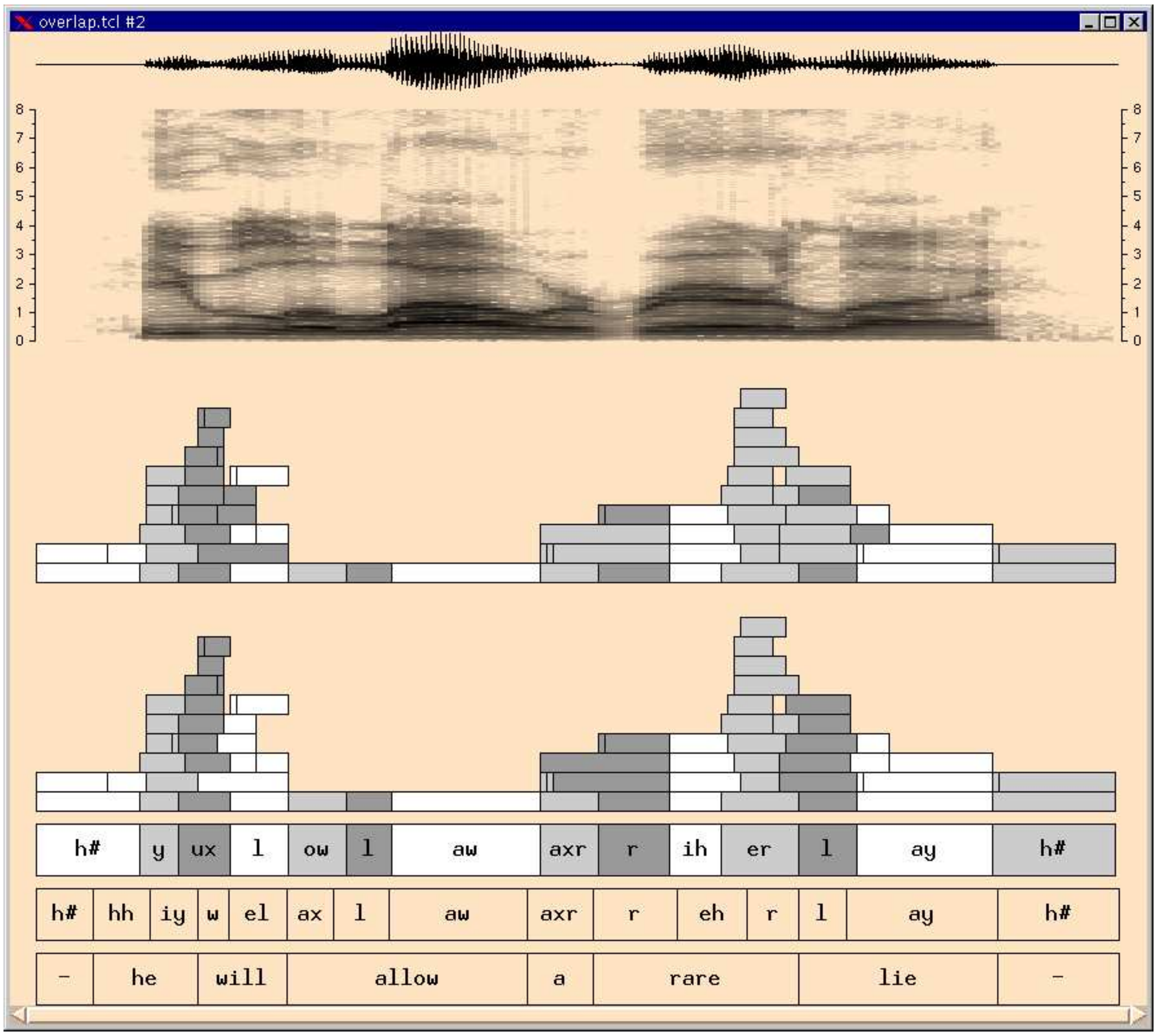


Figure 4-3: This example shows a waveform, spectrogram, segment graphs with near-miss subsets computed using the begin time and midpoint of each segment, the frame-based phone path used to compute near-miss subsets, and the phone and word transcriptions. The segments belonging to the same near-miss subset are drawn using a consistent intensity or gray-level. Three intensities are alternated to allow the visual differentiation of near-miss subsets.

Overlap

This experiment varies the relative segment time that is used in near-miss assignment, from the begin time of each segment to the end time of each segment, and measures the average temporal overlap per segment across the core test set. The temporal overlap of a segment is measured with respect to a reference on-path segment and is defined as the percentage of the segment that is overlapped by the reference segment.

Figure 4-4 shows the results of this experiment. The x-axis shows the relative segment time that is chosen for each segment. For example, 0 is the begin time, 0.5 is the midpoint, and 1 is the end time. The y-axis shows the average temporal overlap over all segments in the core test set. The upper dotted line gives an upper bound on the temporal overlap computed by choosing the best assignment for each segment, and the lower dotted line gives a lower bound on the temporal overlap computed by choosing the worst assignment for each segment.

This experiment explores the space of near-miss assignment strategies in which all segments are assigned based on the same relative time within the segment. In this space, the best strategy is to assign each segment based on its midpoint. In fact, the midpoint strategy is shown to provide close to the best performance that can be achieved, even if each segment is allowed to freely choose the time that maximizes overlap with respect to a reference segment sequence.

Recognition

The goal of maximizing temporal overlap is based on the hypothesis that a larger degree of temporal overlap leads to improved modeling and recognition. To verify this hypothesis, the midpoint and begin time strategies are compared in recognition using 3-state units on the core test set over 39 classes:

- Midpoint assignment results in a 30.5% phonetic recognition error rate.
- Begin time assignment results in a 33.2% phonetic recognition error rate.

As suggested by temporal overlap, assigning near-miss subsets based on the midpoint of each segment results in a lower recognition error rate. All of the experiments

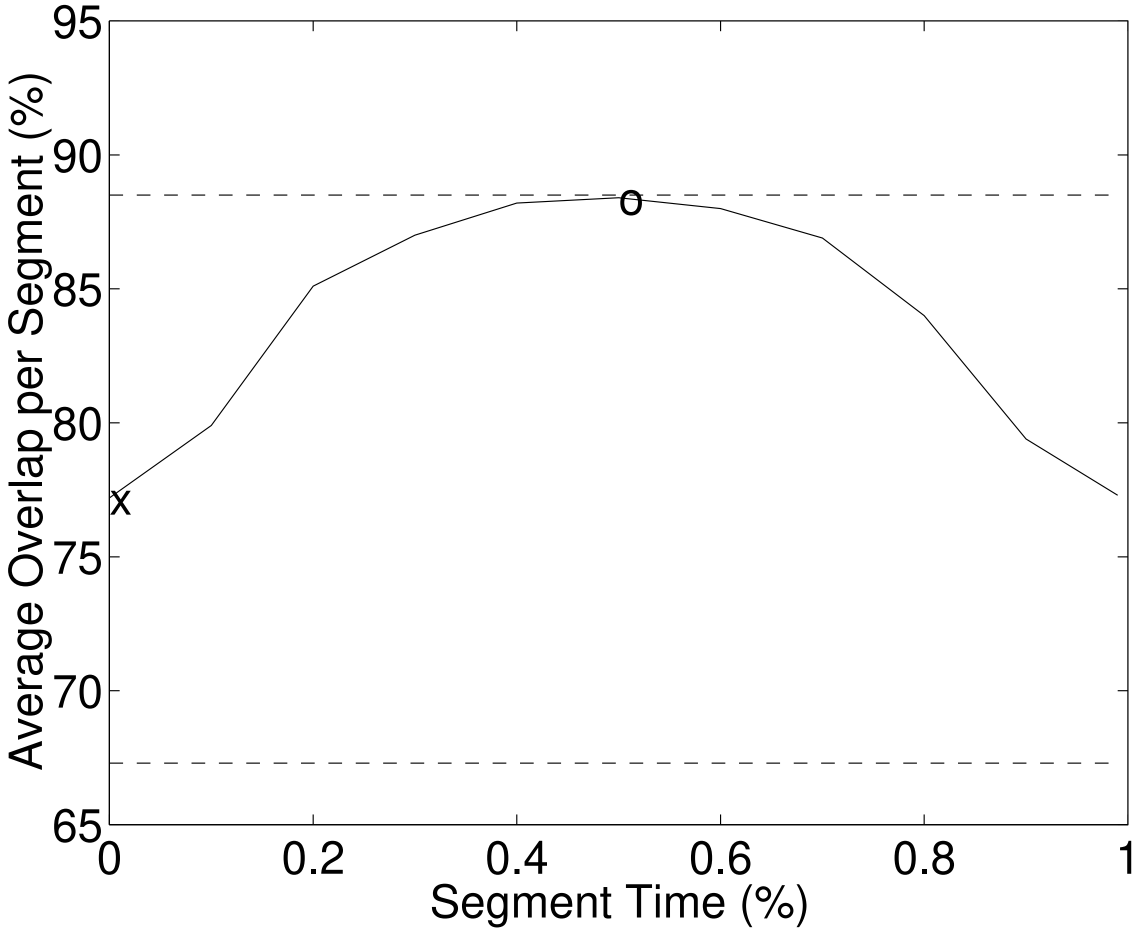


Figure 4-4: This figure shows the average temporal overlap per segment as a function of the relative segment time that is used in near-miss assignment. The x-axis shows the relative time that is chosen for each segment. For example, 0 is the begin time, 0.5 is the midpoint, and 1 is the end time. The y-axis shows the average temporal overlap over all segments in the core test set. The upper dotted line gives an upper bound on the temporal overlap computed by choosing the best possible assignment for each segment, and the lower dotted line gives a lower bound on the temporal overlap computed by choosing the worst possible assignment for each segment.

in the remainder of this chapter use the midpoint strategy.

4.6.2 Near-Miss Units

This section explores near-miss modeling strategies. All of the experiments assign near-miss subsets based on the midpoint of each segment. In addition, these experiments use a recognizer with context-independent segment-based models, a phonetic transition weight, and a bigram language model. The following two sections describe examples and recognition experiments using different near-miss units.

Example

The following two figures contrast the use of multiple contextual near-miss units with the use of a single non-contextual near-miss unit on the same utterance. Figure 4-5 shows a waveform, spectrogram, segment graph, best paths computed using 0-state and 3-state units, and phone and word transcriptions. The unshaded best path is computed using a single non-contextual near-miss unit. The shaded best path is computed by a recognizer using multiple contextual near-miss units, in this case 3-state units. The near-miss subsets are computed and shaded with respect to this best path. The best scoring phones for the segment labeled “ey” in the 0-state path and label “ay” in the 3-state path are listed on the left. In the log domain, the total score for a phone is the sum of the score of the best path segment against the phone and the scores of the five remaining off-path segments in the near-miss subset against the corresponding near-miss units. As shown, the single near-miss unit adds the same score of -78.3 to all phones and cannot result in re-ranking of the phones. In contrast, the use of contextual near-miss units can re-rank scores, such that the correct unit, “ay”, can have the best total score (-88.4) although it does not have the best segment score (-23.4). Overall, this example suggests that the ability to model the context of all segments may lead to improved recognition.

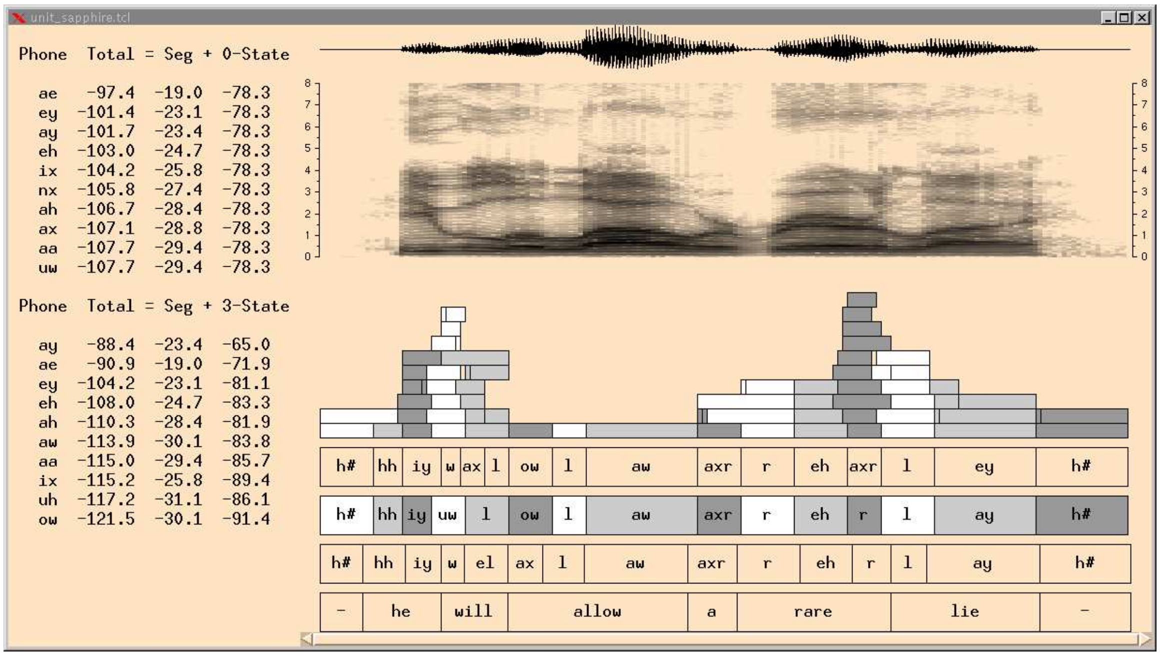


Figure 4-5: This example shows a waveform, spectrogram, segment graph, best path computed using 0-state units, best path computed using 3-state units, and phone and word transcriptions. The near-miss subsets are computed with respect to the best path computed by 3-state units and are drawn with consistent intensities. The best units for the segment labeled "ay" in this best path are listed on the left for both the 0-state and 3-state units. In the log domain, the total score for a unit is the sum of the score of the best path segment against the unit and the scores of the five remaining off-path segments in the near-miss subset against the corresponding 3-state unit.

Recognition

Table 4.14 shows recognition error rates on the core test set over 39 classes for 0-state, 1-state, and multi-state units. There are 61 lexical units and therefore 1 0-state unit, 61 1-state units, 122 2-state units, and 183 3-state units. However, because the 0-state unit is also scored for the contextual units to provide backoff, there are effectively 1 0-state unit, 62 1-state units, 123 2-state units, and 184 3-state units. The smoothing parameter for backoff is set by minimizing recognition error rate over a development set. For all experiments, the smoothing weights the higher order models by 0.4 and the 0-state model by 0.6.

State	Error (%)	Δ (%)
0	33.9	-
1	31.7	6.5
2	30.9	8.9
3	30.5	10.0

Table 4.14: This table shows recognition error rates over the core test set for 0-state, 1-state, and multi-state units. As the number of units increases, the error rate consistently decreases. There is a substantial improvement over the 0-state model, which is functionally equivalent to the anti-phone model.

In comparison to using the 0-state unit, the use of 1-state units reduces recognition error rate by 6.5%, and the use of additional temporal constraints further reduces error to a total of 10.0%.

4.6.3 Computation

The following sections compare the computational costs of different near-miss units and the anti-phone unit. Computation is measured by the number of mixtures that are scored per millisecond of speech. The maximum number of mixtures in any model is 100. All experiments use the same pruning threshold, which is set relatively high in order not to sacrifice performance.

0-State vs. Anti-Phone

This section compares the computational requirements of near-miss modeling using the 0-state unit and anti-phone modeling. These two strategies are identical, functionally, but represent different implementations. The average number of mixtures that are scored per millisecond of speech during recognition over the core test set for 0-state near-miss modeling and anti-phone modeling:

- 0-state near-miss modeling scores 188.0 mixtures per millisecond of speech.
- Anti-phone modeling scores 188.3 mixtures per millisecond of speech.

Overall, 0-state near-miss modeling and anti-phone modeling require about the same amount of computation. The slight difference can be examined in more detail at the utterance level. Figure 4-6 shows a scatter plot of the relative number of mixtures scored by 0-state near-miss modeling and anti-phone modeling as a function of segment density for each utterance in the core test set. The x-axis shows the number of segments per second in the segment graph for each utterance. The y-axis shows the ratio of the number of mixtures scored by 0-state near-miss modeling and anti-phone modeling.

As shown, near-miss modeling tends to offer more computational savings when the segment graphs have fewer segments per second. This is because in smaller graphs, it is more likely that a segment may be on all segment sequences and therefore does not have to be scored against the 0-state unit. In addition, because the pruning is not aggressive, all segments are scored against the anti-phone unit, and near-miss modeling can only save computation.

Near-Miss Units

Table 4.15 shows the average number of mixtures that are scored per millisecond of speech during recognition over the core test set for the 0-state, 1-state, and multi-state units. In comparison to using the 0-state unit, the use of 1-state units more than doubles the required computation. However, the addition of temporal units

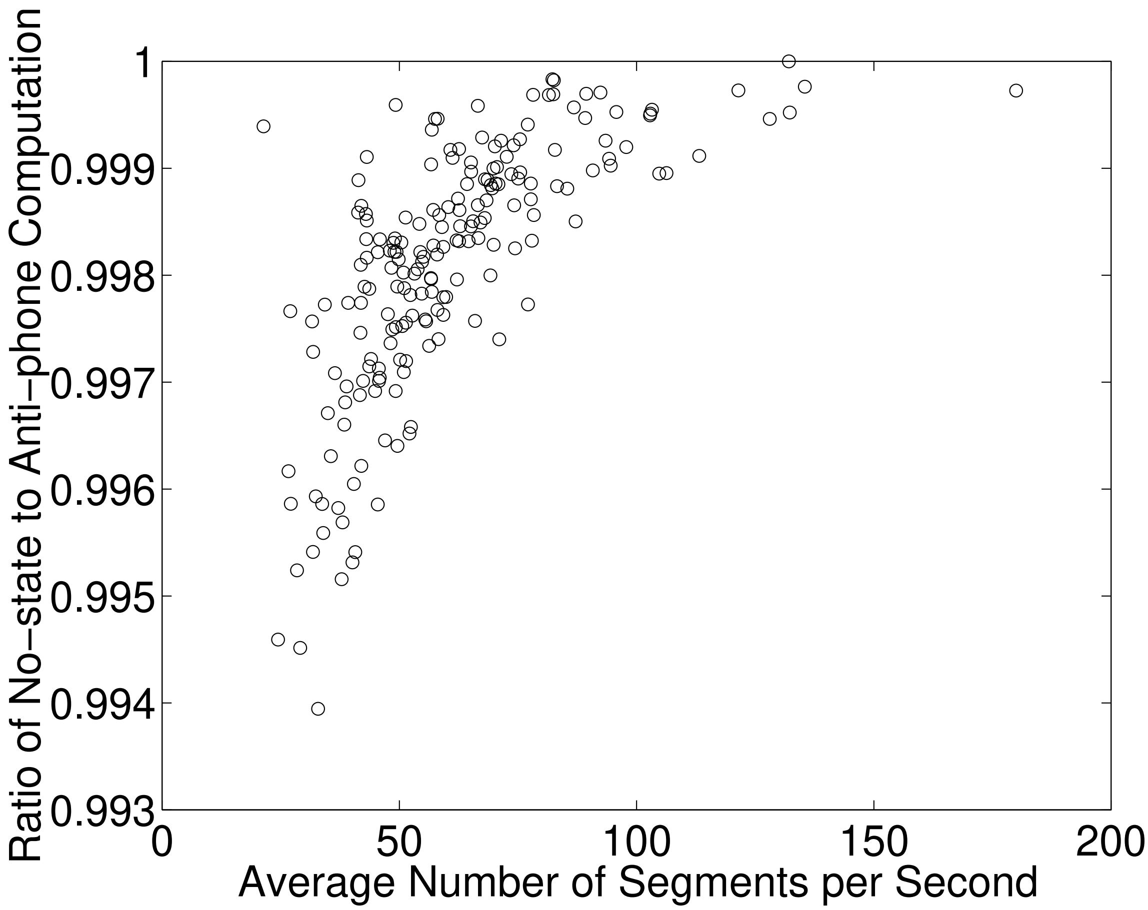


Figure 4-6: Ratio of the amount of computation performed, per utterance, between the 0-state near-miss model and the anti-phone model, for TIMIT. Each dot represents a single utterance from the TIMIT core test set, and the x-axis is the average number of segments per second for the utterance. The near-miss model is always equal in computation or slightly less. Utterances with fewer segments per second are more likely to contain segments which must be on all segment sequences, which the near-miss model will not score while the anti-phone model will.

results in a relatively small increase in computation. In fact, the 3-state units require less computation than the 2-state units. This suggests that the midpoints of many off-path segments may fall in the middle third of the on-path segments and result in less temporal ambiguity in assignment.

State	Mix/ms	Δ (%)
0	188.0	-
1	416.9	121.8
2	478.8	154.7
3	477.2	153.8

Table 4.15: This table shows the average number of mixtures that are scored per millisecond of speech during recognition over the core test set for the 0-state, 1-state, and multi-state units. The 0-state unit is also scored for the contextual units to provide backoff.

4.6.4 Context-Dependent Modeling

This section explores the use of diphone context-dependent frame-based units. The recognition error rates are computed on the core test set over 39 classes for a recognizer using context-independent segment-based units with and without diphone context-dependent frame-based units. The segment-based units consist of lexical units and 3-state near-miss units.

- Context-independent segment-based models alone achieve a recognition error rate of 30.5%.

- Context-dependent frame-based and context-independent segment-based models achieve a recognition error rate of 25.5%.

In comparison to using only context-independent segment-based units, the addition of diphone context-dependent frame-based units reduces recognition error rate by 16.0%. Note that the system using only segment-based models cannot be described as context-independent due to the manner in which the segment graphs are generated.

The purpose in describing it here is to show the generality of the near-miss modeling framework. The use of these diphone context-dependent units will be explained in more detail in the following chapter, as will a comparison with other results reported in the literature. This chapter has focused on near-miss modeling and concludes with a brief summary. The following chapter will report on the remaining aspects of near-miss modeling and then compare near-miss modeling with other approaches.

4.7 Summary

This chapter has described a novel probabilistic framework for segment-based speech recognition called near-miss modeling. Near-miss modeling is based on the ability to compute, for any graph, an association of near-miss subsets to segments such that any segment sequence accounts for all segments. The computation of this association is based on choosing any time within a segment and assigning it to all near-miss subsets that span the chosen time. Empirically, it is shown that an effective way of assigning segments is based on their midpoints.

In addition to maintaining the probabilistic framework, the near-miss search can be efficiently implemented using the Viterbi algorithm. During search, the score of each segment is effectively the product of the scores of all of the segments in its near-miss subset.

Near-miss modeling provides a general framework which can incorporate existing frame-based and anti-phone modeling strategies. In addition, the near-miss search can allow more sophisticated modeling strategies that enforce contextual constraints across all segments in a graph. Empirically, it is shown that near-miss units can capture phonetic and temporal constraints in the off-path segments, resulting in significantly improved performance on the task of phonetic recognition.

Chapter 5

Near-Miss Segmentation

This chapter describes a new segmentation algorithm that is well matched to the near-miss modeling framework. The near-miss segmentation algorithm applies multiple sources of constraints by running a recognizer and extracting the most probable segments. By taking more constraints into account, the algorithm produces segment graphs which have fewer segmentation errors and leads to improved recognition accuracy. The following sections describe the near-miss segmentation algorithm in more detail and continues the evaluation of near-miss modeling.

5.1 Segmentation Framework

The near-miss segmentation algorithm produces a segment graph by running a first pass recognizer with a backwards A^* search. This search is the same backwards A^* search that is used to generate a word graph, except that it is altered to work with segments instead of words [27]. The first pass recognizer runs a Viterbi search to find the best path and provides a lattice of scores. The backwards A^* search uses the Viterbi lattice as its future estimate and produces the segments that are in the paths that score within a threshold of the best path. When two paths arrive at the same point, they are merged such that only the higher scoring path continues. The A^* search guarantees that only those segments that are used in a recognition path whose score is within a global pruning threshold of the score of the best path will be selected and retained for the output segment graph. By varying the threshold, the recognizer will produce larger or smaller segment graphs. If the threshold is zero, the output includes exactly one sequence of segments, the best path from the Viterbi search. With a higher threshold, more segments will be included. As a potentially useful byproduct of the search, each retained segment also has an associated set of hypothesized labels and scores. This information can either be discarded or may be retained and used by the subsequent segment-based recognizer.

5.1.1 Characteristics

The near-miss segmentation algorithm has many attractive properties for segment-based speech recognition:

- In comparison to an acoustic segmentation algorithm, the near-miss segmentation algorithm uses all of the sources of constraint that are typically used in recognition to decide which segments should be included in the output segment graph. These constraints may include any acoustic, segmentation, pronunciation, and language models. As a result, the near-miss segmentation algorithm has the potential to be more accurate and result in better alignments and fewer insertions and deletions. This ability to produce a more accurate segment graph is an important step towards the development of segment-based approaches.

- As the near-miss segmentation algorithm hypothesizes only the most probable segments, it is adaptive to all sources of variation, whether from the segment, word, utterance, speaker, or environment. In regions where the recognizer is confident of its choices, the resulting segment graph is singular, containing a single sequence of segments. In regions of uncertainty, it will instead hypothesize many possible alternative segments. This can result in substantially different segment graphs for varying utterances. Some portions of utterances will have a high segment density, measured as segments per second, while others will have a low segment density. This adaptive property allows subsequent segment-based acoustic modeling to focus on those areas of utterances where it is most needed.

The previous chapter has already shown how near-miss modeling allows the acoustic model to focus on discriminating between confusable segments. The near-miss segmentation algorithm is well matched to focus the subsequent near-miss search on only the most confusable segments.

- The near-miss segmentation algorithm can use any recognition strategy to hypothesize segments. Computationally, the strategy can be as inexpensive as an acoustic segmentation algorithm or as expensive as running a complete HMM word recognizer. In general, the more constraint is incorporated into the first pass recognizer, the more accurate the segment graph will be.

- The near-miss segmentation algorithm can hypothesize any type of unit, not just acoustic-phonetic ones, by configuring the recognizer to use the new units. For example, the transition between stop closures and releases are often defined by sharp discontinuities and therefore a segment boundary is typically placed between them. However, studies have shown that there is a high correlation between the closure and burst regions of stops. Therefore, stops could be better modeled as a single unit [25].

- The combination of near-miss segmentation and near-miss search is a powerful framework for combining the relative advantages of frame- and segment-based approaches. First, an efficient frame-based recognizer can be used to prune the segment graph that is searched by the second stage segment-based recognizer. Then, the second stage segment-based recognizer can use segment-based modeling strategies to further constrain the output. As a result, the frame- and segment-based approaches can be combined in a mutually beneficial way. Furthermore, it is efficient in the sense that some of the computation towards frame-based classification, performed by the frame-based recognizer can subsequently be re-used by the segment-based search.

- As a byproduct of first pass recognition, each output segment is augmented with information from classification and search. The information about segments, phones, and scores can be used to improve and focus the second pass segment-based recognizer. For example, this model could support a framework for a hierarchical approach to speech recognition [8, 25].

5.1.2 Frame-Based Recognizer

The near-miss segmentation algorithm can employ any recognition strategy to hypothesize segments. The goal of this thesis is to use a powerful recognizer to generate segment graphs. The experiments in this chapter use a frame-based recognizer that applies all of the constraints of the recognizer used in the previous chapter except segment-based modeling itself. This includes the phonetic transition weight and the bigram language model. Acoustically, the frame-based recognizer hypothesizes frames every 10 ms and uses diphone-context dependent units as described in Chapter 3. For reference, this recognizer is able to achieve a phonetic recognition error rate of 26.5% on the TIMIT core test set.

Note that in this chapter, the near-miss segmentation algorithm uses diphone context-dependent acoustic models and a phone bigram language model that requires a substantial amount of computation and training. In contrast, the acoustic segmentation algorithm uses much less computation and requires little training. However, if the constraints of computational efficiency and domain independence are important, the near-miss segmentation algorithm can use an appropriate recognizer. In this thesis, the goal is to show that the near-miss segmentation algorithm can produce accurate segment graphs than can be used to achieve competitive performance in recognition. As a result, the issues of computational efficiency and domain independence are not directly addressed.

5.2 Evaluation

This section continues the evaluation of near-miss modeling on phonetic recognition using the experimental framework detailed in Chapter 3. The remaining evaluation has three parts. The first part evaluates the near-miss segmentation algorithm, the second part evaluates the combined near-miss recognizer, and the last part compares the combined near-miss modeling approach with other approaches to speech recognition.

5.2.1 Segmentation

This section characterizes the near-miss segmentation algorithm and compares it to our current acoustic segmentation algorithm [69]. To measure the accuracy of a segment graph, a temporal alignment tool has been developed and integrated into SAPPHIRE. The temporal alignment tool evaluates the accuracy of a segment graph by aligning a reference segment sequence to the segment graph. Given the reference segment sequence, the tool performs a search to find the best temporally matched sequence of segments through a graph by minimizing the total match, insertion, and deletion errors. The cost of a match is the relative percent of each reference segment that is not temporally overlapped by its matching segment in the graph. The cost of a deletion is 1.0, which is equivalent to the cost of a match with no temporal overlap. Similarly, the cost of an insertion is 1.0. The search is implemented using the Viterbi algorithm.

Example

This section compares example segment graphs produced for the same utterance by the acoustic and near-miss segmentation algorithms. Figure 5-1 shows a waveform and spectrogram, segment graphs generated by acoustic and near-miss segmentation, best alignments of the acoustic and near-miss segment graphs to the phone transcription, and phone and word transcriptions. In comparison to the acoustic segment graph, the near-miss segment graph is smaller and more adaptive. In this example, the acoustic segmentation algorithm generates the highest density of segments in the region in which the near-miss segmentation algorithm is most confident and produces three singular segments, which align closely to the phone transcription.

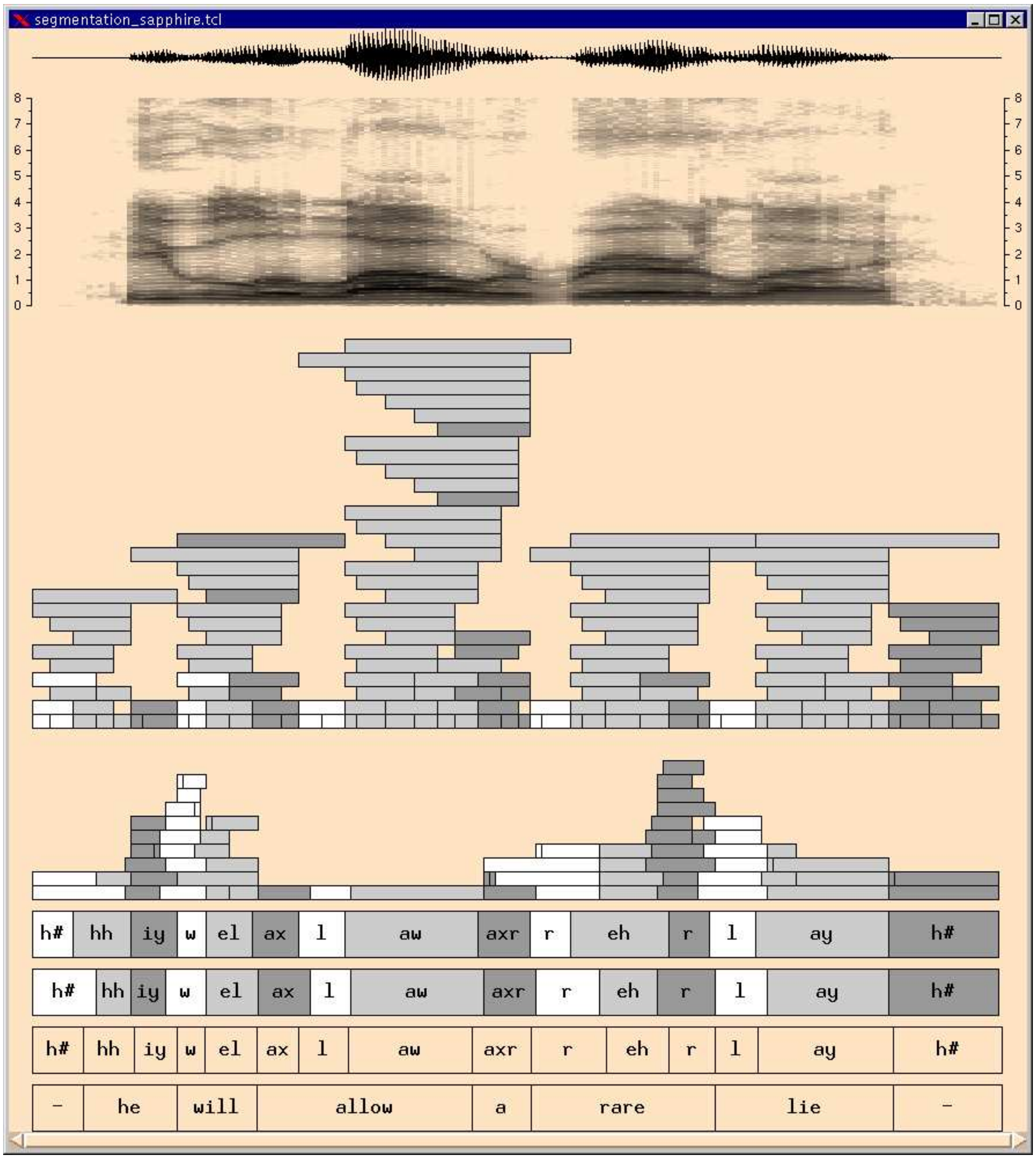


Figure 5-1: This figure shows a waveform and spectrogram, segment graphs generated by the acoustic and near-miss segmentation algorithms, best alignments of the acoustic and near-miss segment graphs to the phone transcription, and phone and word transcriptions.

Adaptability

Adaptability refers to the ability of a segmentation algorithm to hypothesize a variable, rather than a uniform, number of segments across an utterance. Adaptability can be measured by counting the number of segments per near-miss subset across the core test set. The near-miss subsets are computed with respect to the best path through the segment graph as computed by a frame-based recognizer.

Figure 5-2 shows a histogram of the number of near-miss segments in each near-miss subset generated by the acoustic segmentation algorithm. In this case, the mean near-miss subset size is 5.7 segments. Figure 5-3 shows the same histogram for the near-miss segmentation algorithm, in which case the mean near-miss subset size is 5.0 segments. In comparison to the acoustic segmentation algorithm, the near-miss segmentation algorithm generates fewer segments. In addition, the distribution of these segments across near-miss subsets is less uniform. In fact, most of the near-miss subsets contain only the single feature vector that is directly associated with each segment.

Phonetic Analysis

Another way to characterize the near-miss segmentation algorithm is to examine the size of the near-miss subsets as a function of the hypothesized phone. As in previous experiments, the near-miss subsets are computed with respect to the best path through the segment graph that is computed by the frame-based recognizer. Table 5.1 shows the mean number of segments in the near-miss subsets of each phone over the core test set.

The mean size of near-miss subsets varies substantially by phone and may reflect both duration and difficulty of classification. For example, the phones with smaller subsets tend to be stop closures or releases, which are relatively short in duration and acoustically well-defined. In contrast, the phones with larger subsets tend to be diphthongs and semivowels which are relatively long in duration and consist of gradual transitions.

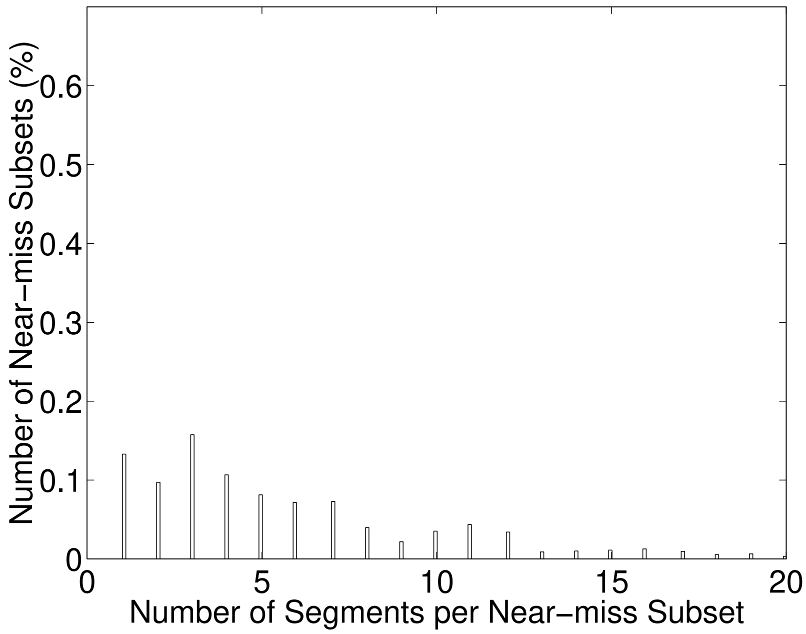


Figure 5-2: Histogram of the number of segments in each near-miss subset generated by the acoustic segmentation algorithm.

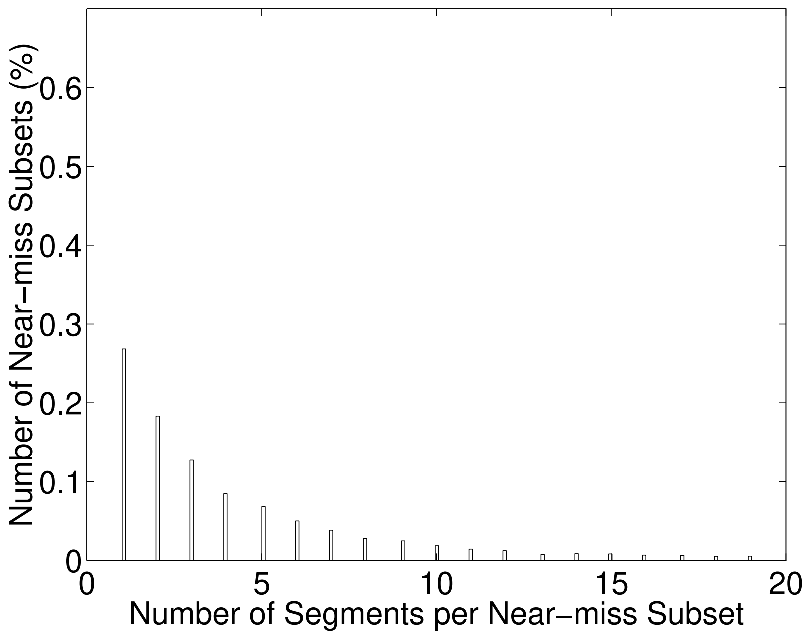


Figure 5-3: Histogram of the number of segments in each near-miss subset generated by the near-miss segmentation algorithm.

#	Class	#	Class
1.9	b	5.2	aa ix
2.0	t	5.4	ey ng w
2.1	epi k	5.5	ah ih n th
2.3	g	5.6	ae
2.6	d kcl p	5.7	f m
2.7	h#	6.0	aw
2.8	bcl hh	6.1	q
2.9	pcl	6.6	ao
3.1	tcl	7.0	axr
3.2	dh sh	7.2	l
3.3	ax-h s	7.3	pau
3.4	z	7.8	er
3.7	jh	8.1	hv ux
3.8	gcl nx	8.9	v
3.9	dcl	9.2	ow
4.4	ax iy y	9.8	el
4.6	ch	10.0	ay
4.7	eh r	10.8	uw
4.8	dx	16.6	oy
5.0	uh		

Table 5.1: The mean number of segments in the near-miss subsets generated by near-miss segmentation by phone across the core test set.

Alignment

This section presents a set of experiments that measure the alignment error against the phone transcription over the core test set using different pruning thresholds for near-miss segmentation to generate segment graphs of different sizes. The alignment error reflects the relative percent of each reference segment that is not temporally overlapped by its matching segment in the graph.

Figure 5-4 shows the alignment error of the segment graphs produced by the near-miss segmentation algorithm as a function of the size of the segment graphs. The x-axis shows the average number of segments per second, while the y-axis shows the average alignment error per reference segment in percent. For reference, the dashed vertical line shows 12.7 as the average number of segments per second in the phone transcriptions. The `x` above the curve shows the alignment error for the acoustic segment graph. At this point, there are 87.8 segments per second and 15.0% error per segment. The curve shows the alignment error for the near-miss segment graphs and is generated from six points. The `o` on the curve denotes the segment graphs that are used in all of the experiments that use a consistent segment graph size. At this point, there are 32.7 segments per second and 12.9% alignment error per segment.

The curve is generated by varying the pruning threshold for the A^* search. When the threshold is zero, the segment graphs contain only single segment sequences and have approximately as many segments as the phone transcriptions with an alignment error of 20.7%. As the threshold is increased, the number of segments increases, while the alignment error decreases. In comparison to the acoustic segmentation algorithm, marked by `x`, the near-miss segmentation algorithm offers better tradeoffs. Not only are the near-miss graphs smaller in size, but they also have lower alignment error.

Recognition

This section presents a set of experiments that measure the recognition error on the core test set over 39 classes using different pruning thresholds. For each threshold, a segment-based recognizer using 3-state near-miss units is retrained and tested.

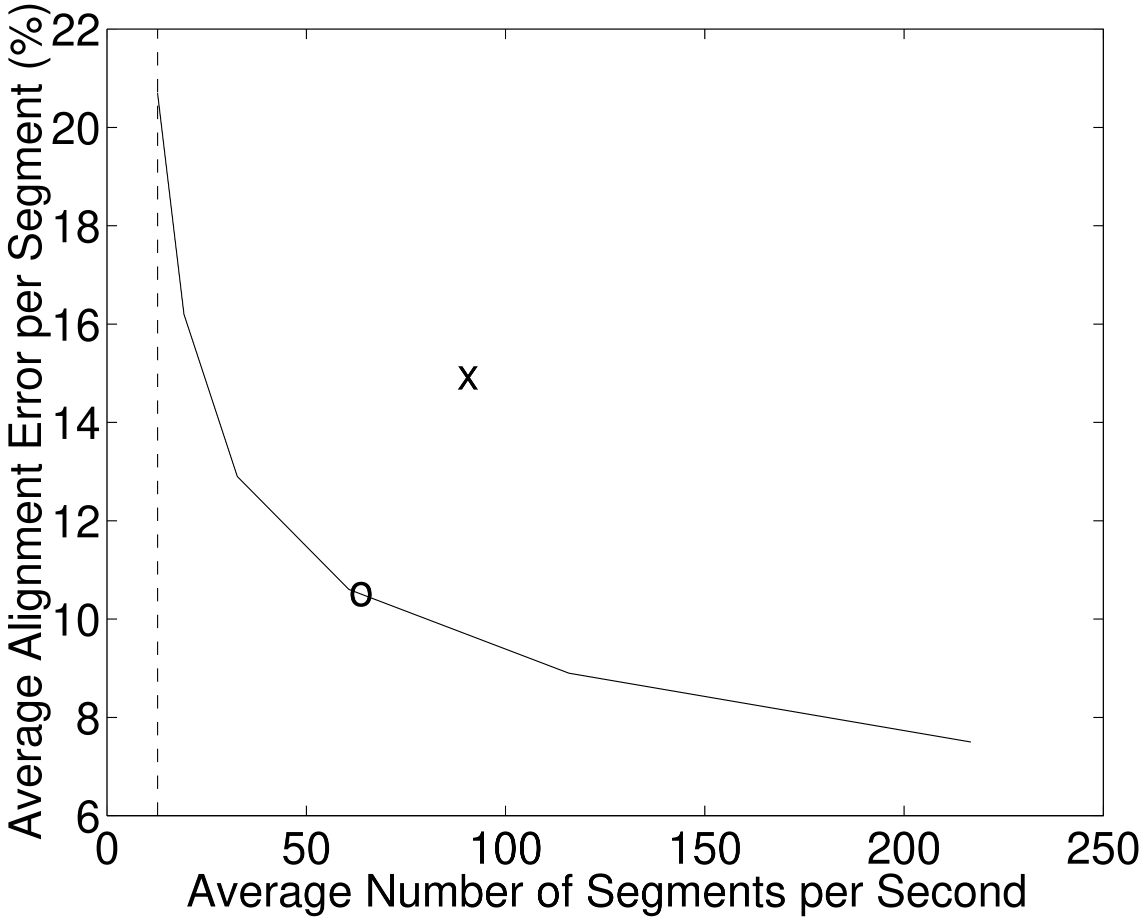


Figure 5-4: Alignment error as a function of the size of the segment graphs. The x-axis shows the average number of segments per second, while the y-axis shows the average alignment error per reference segment in percent. The curve shows the tradeoffs using near-miss segment graphs. The `o` on the curve denotes the segment graphs that are used in all of the experiments that use a consistent segment graph size. The `x` above the curve denotes the acoustic segment graph. For reference, the dashed vertical line shows the number of segments per second in the phone transcriptions.

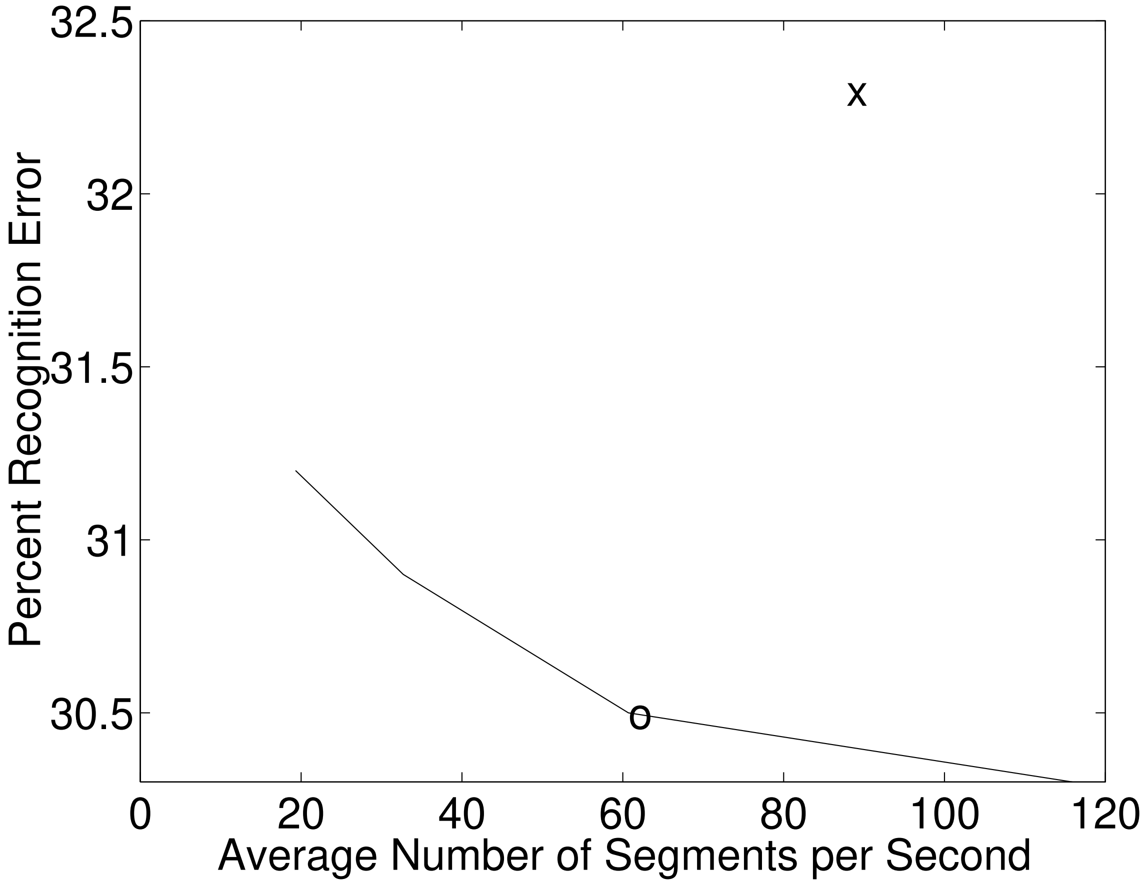


Figure 5-5: This figure shows segment-based recognition error rate as a function segment graph size. The x-axis shows the average number of segments per second. The y-axis shows the percent recognition error over the core test set. The `o` on the curve denotes the segment graph that is used in all of the experiments that use a consistent segment graph size. The `x` near the top of the graph shows the recognition error rate for the acoustic segment graph.

Figure 5-5 shows segment-based recognition error rate as a function the size of the segment graphs. The x-axis shows the average number of segments per second. The y-axis shows the percent recognition error over the core test set. The curve is drawn from four points. The `o` on the curve denotes the segment graph that is used in all of the experiments that use a consistent segment graph size. The `x` near the top of the graph shows the recognition error rate for the acoustic segment graph.

Figure 5-5 shows a tradeoff between performance and computation. The recognition error decreases as the segment graphs increase in size. In addition, the near-miss segment graphs are shown to provide better performance and computation than the acoustic segment graphs. However, this latter result may be biased by the fact that the two recognizers do not use the same degree of context. The acoustic segmentation algorithm is context-independent, while the near-miss segmentation algorithm makes use of diphone context-dependent constraints. The following section provides a less biased comparison of the two segmentation algorithms in recognition.

5.2.2 Combined Recognition

This section presents results using the combined recognizer running near-miss segmentation and near-miss search.

Segmentation

To compare the near-miss and acoustic segmentation algorithms using the same degree of context, both systems are run with diphone context-dependent frame-based models. Table 5.2 shows recognition error rates on the core test set over 39 classes using context-independent segment-based units and diphone context-dependent frame-based units for the near-miss and acoustic segment graphs. The segment-based units consist of lexical units and 3-state near-miss units.

The difference between these two systems is that the near-miss segmentation algorithm applies the diphone context-dependent constraints prior to segmentation, whereas the acoustic segmentation algorithm does not apply contextual constraints

Segmentation	Error (%)	Δ (%)
Acoustic	30.0	-
Near-Miss	25.5	15.0

Table 5.2: Recognition error rates over the core test set using context-independent segment-based units and diphone context-dependent frame-based units for the near-miss and acoustic segmentation algorithms.

until after segmentation. The earlier application of contextual constraints to the segmentation problem results in a 15% reduction in error.

Recognition

Table 5.3 compares the first and second pass recognition error rates on the core test set over 39 classes. The addition of context-independent segment-based models reduces error rate by 4%.

Models	TIMIT (%)	Δ (%)
CD Frame	26.5	-
CD Frame + CI Segment	25.5	4.0

Table 5.3: This table compares recognition error rates using diphone context-dependent frame-based models with and without context-independent segment-based models. The segment-based units consist of lexical units and 3-state near-miss units. The addition of segment-based modeling improves the error rate by 4%.

It is important to note that while the context-dependent frame-based recognizer is able to achieve low recognition error rate, the addition of context-independent segment-based models can still improve performance. Future explorations of more powerful segment-based modeling strategies is expected to further improve these results.

Error Analysis

This section presents an error analysis of the combined recognizer using context-independent segment-based models and context-dependent frame-based models on the core test set over 39 classes. Table 5.4 shows the breakdown of the recognizer error rate into substitution, deletion, and insertion rates.

Error (%)	Sub (%)	Del (%)	Ins (%)
25.5	15.5	7.0	3.0

Table 5.4: This table shows the breakdown of the recognition error rate of the combined recognizer into substitution, deletion, and insertion rates.

Table 5.5 shows the ten most frequent substitutions, deletions, and insertions, along with their frequency of occurrence.

Sub			#	Del	#	Ins	#
ih	→	ah	39	pau	109	pau	54
ah	→	ih	35	ih	53	ih	16
m	→	n	32	l	46	d	15
er	→	r	31	n	39	ah	14
z	→	s	30	ah	32	aa	12
r	→	er	27	r	30	n	12
eh	→	ih	24	dh	17	l	11
eh	→	ah	23	v	16	k	8
ih	→	iy	22	dx	15	dh	7
ih	→	eh	20	hh	14	hh	7

Table 5.5: This table shows the ten most frequent substitutions, deletions, and insertions for the context-dependent recognizer, along with their frequency of occurrence.

Most of the substitutions are between confusable phones within the same manner class. For example, the two most frequent substitutions are due to confusions between schwas. Most of the deletions and insertions are of temporally short or spectrally weak phones. For example, the most frequent deletion and insertion is of stop closures.

5.2.3 Comparison

TIMIT is a commonly used corpus for evaluation on the task of phonetic recognition. This thesis chooses representative state-of-the-art systems that have reported results on the NIST core test set over 39 phonetic classes. This section compares the near-miss modeling system with an anti-phone modeling system, HMM system, and recursive neural network (RNN) system. All of the systems use context-dependent acoustic models and a phone bigram language model. Only the HMM system uses gender-dependent acoustic models.

- The near-miss modeling system has two passes. The first pass uses diphone context-dependent frame-based acoustic models and a phone bigram language model within a frame-based framework to generate accurate segment graphs. The second pass re-uses both models from the first pass and also adds context-independent segment-based acoustic models within a near-miss modeling framework. Both frame- and segment-based acoustic models use mixture of diagonal Gaussian distributions.

- The anti-phone modeling system uses an acoustic segmentation algorithm to generate segment graphs [22]. It then uses context-independent segment-based models and diphone context-dependent landmark-based models and a phone bigram language model within an anti-phone modeling framework. Both frame- and segment-based acoustic models use mixture of diagonal Gaussian distributions. This system represents the state-of-the-art segment-based approach.

- The HMM system runs two gender-dependent recognizers [29]. Each recognizer uses triphone context-dependent acoustic models and a phone bigram language model within an HMM framework. The HMMs have three states and use mixture of diagonal Gaussian distributions. The higher scoring recognizer output is chosen for each utterance. This system represents the state-of-the-art HMM approach.

- The RNN system use context-dependent frame-based acoustic models and a phone bigram language model within an HMM framework [57]. The HMMs have one state. The acoustic models use recursive neural networks. This system represents the state-of-the-art neural network approach.

Table 5.6 summarizes reported results on the TIMIT core test set over 39 classes. These results show that near-miss modeling can achieve state-of-the-art performance in phonetic recognition. It represents a 16% reduction in error from our previously best reported result as described in the anti-phone modeling system. It also represents a 2% reduction in error from the best reported result as described in the RNN system.

Description	Error (%)
Near-Miss	25.5
Anti-phone [22]	30.5
HMM [29]	30.9
RNN [57]	26.1

Table 5.6: This table shows recognition error rates that have been reported on the TIMIT core test set over 39 classes.

Although this thesis does not focus on frame-based recognition, note that the first pass recognizer in the near-miss system by itself can achieve a recognition error rate of 26.5%. This result is significantly better than the result for the second anti-phone system, largely due to the fact that the anti-phone system constrains itself to a small set of landmarks and segments. This result is also significantly better than the result achieved using the HMM system that is more complex in its use of gender-dependent instead of gender-independent models and triphone instead of diphone context-dependent models. This is largely due to the fact that the near-miss system uses frame-based feature vectors that span a duration of 140 ms and allow the implicit capture of context-dependent and segment-based constraints. Furthermore, this suggests that HMM frame-based recognizers may significantly improve their performance by using feature vectors that span longer durations [25, 65].

5.3 Summary

This chapter has described a near-miss framework for segmentation that can utilize all sources of the constraint that are typically used in recognition towards the segmentation problem. This is done by running a recognizer and generating a segment graph that contains only those segments that correspond to paths that score within an input threshold of the best scoring path. Empirically, the near-miss segmentation algorithm is shown to significantly reduce both alignment error and phonetic recognition error rates. The near-miss modeling framework, which combines near-miss search with near-miss segmentation, has been shown to be an extremely competitive approach to phonetic recognition.

Chapter 6

Word Recognition

The previous chapters have shown that near-miss modeling can achieve significant improvements in phonetic recognition. Typically, improvements in phonetic recognition generalize to improvements in word recognition [29]. However, this chapter verifies that near-miss modeling can also achieve improvements in word recognition by performing experiments at the word level. The first section describes the framework for experiments at the word level. The next two sections describe the experiments in segmentation and search and compares near-miss modeling to other approaches in word recognition.

6.1 Experimental Framework

The following three sections describe the corpus, the lexicon, and the recognizers used in word recognition.

6.1.1 ATIS Corpus

Experiments in word recognition are performed on the Air Travel Information Service (ATIS) corpus that was used as a common ARPA spoken language testbed from 1990 to 1994 [47, 48]. The ATIS corpus is composed of spontaneous, continuous speech that has been orthographically transcribed. In the ATIS task, subjects obtain air travel information from a database using spoken natural language in order to solve air travel planning scenarios. Example queries include "How many meals does America West airlines serve between Washington D. C. and Columbus, Ohio?" and "What's the cheapest flight from San Francisco to Detroit today?"

In this thesis, ATIS is divided into independent test, development, and training sets. Except where otherwise noted, the ATIS results are reported on the test set from the last evaluation in December 1994 containing 981 utterances spoken by 24 speakers [47]. All of the intermediate experiments are run on the test set from the previous evaluation in December 1993 containing 965 utterances spoken by 27 speakers [48]. The training set contains the remaining 22,606 utterances spoken by 594 speakers. Table 6.1 shows the number of speakers, utterances, and words in the test, training, and development sets.

Set	# Speaker	# Utterance	# Word
Train	594	22,606	217,140
Development	27	965	8,643
Test	24	981	10,081

Table 6.1: The number of speakers, utterances, and words in the test, training, and development sets in ATIS.

6.1.2 PRONLEX Lexicon

Unlike TIMIT, ATIS is not phonetically transcribed. Word pronunciations are obtained from Release 0.2 of the PRONLEX American English pronunciation lexicon from the Linguistic Data Consortium (LDC). PRONLEX contains 90,694 words, which cover all but five of the words in the ATIS vocabulary. Pronunciations are created for these words based on existing variants. PRONLEX provides pronunciations in simple citation form without accounting for systematic phonological variations. Alternate pronunciations are given for words whose pronunciation varies in specific ways, such as by part of speech. For these words, the correct pronunciation was used. Otherwise, the first pronunciation was chosen. PRONLEX also marks all vowels with stress, non-main stress and lack of stress, and tags special classes of words, such as proper names, function and foreign words. These marks and tags are not used in this thesis. PRONLEX uses a set of 43 phones. Table 6.2 shows the long and short form for the phones along with example sounds as indicated by italicized letters in the example words.

Long	Short	Example	Long	Short	Example
aa	a	h*o*d	k	k	*k*id
ae	@	h*a*d	l	l	*l*awn
ah	A	c*u*d	m	m	*m*e
ao	c	l*aw*	n	n	*n*o
aw	W	h*ow*'d	ng	G	ha*ng*
ax	x	dat*a*	ow	o	h*o*ed
ay	A	h*i*de	oy	O	B*oy*d
b	b	b*e*d	p	p	*p*ot
ch	C	*ch*eck	r	r	*R*alph
d	d	*d*one	s	s	*s*ix
dh	D	*th*is	sh	S	*sh*in
eh	E	h*ea*d	t	t	*t*one
em	M	-	th	T	*th*in
en	N	butt*on*	uh	U	c*ou*ld
er	R	h*er*d	uw	u	wh*o*'d
ey	e	*ai*d	v	v	*v*ex
f	f	*f*ix	wh	H	*wh*ich
g	g	*g*aff	w	w	*w*itch
hh	h	*h*elp	y	y	*y*es
ih	I	h*i*d	z	z	*z*oo
iy	i	h*ee*d	zh	Z	plea*s*ure
jh	J	*j*udge			

Table 6.2: The set of 43 PRONLEX phones in long and short form along with example sounds as indicated by the italicized letters in the example words.

As in phonetic recognition, all word recognition error rates are computed using the NIST alignment program [16].

6.1.3 Word Recognizers

In comparison to phonetic recognition, the task of word recognition can have a much larger and more complex search space. As a result, the phonetic recognizer described in Chapter 3 cannot be directly extended to word recognition using current SAPPHIRE tools. In particular, there are two major limitations [28]:

- The current tools are not able to enforce context-dependent constraints across all phones in conjunction with either alternate pronunciation models or class n-gram language models. To maintain the focus on acoustic modeling, the word recognizers in this thesis use context-dependent modeling but do not use the more complex pronunciation and language models that are used in most evaluated systems.

- The current tools also cannot take advantage of the efficiencies in a frame-based search. As a result, it is computationally expensive to run the frame-based recognizer that is used for segmentation. To facilitate experimentation, the word recognizers in this thesis intentionally sacrifice performance to reduce computation.

Given these limitations, a consistent set of experiments in word recognition have been designed. The following two sections describe the acoustic and other models that are used in the word recognizers.

Acoustic Model

The word recognizers use acoustic models that are similar to the ones used in phonetic recognition except that, to reduce computation, the word recognizers use a maximum of 50, rather than 100, mixtures. The units that are modeled vary between the segment- and frame-based models:

- The segment-based units are context-independent. The word recognizers model all of the 41 phones in PRONLEX that have sufficient data in the ATIS training set. Due to lack of data, "en" is mapped to the sequence "ih n", and "zh" is mapped to “sh”. In addition, “em” has no tokens and is not modeled. Since there are no pronunciation rules, the word recognizers in ATIS model fewer units than the phonetic recognizers in TIMIT. Table 6.3 shows the set of 41 phones that are used in word recognition.

Label	Label	Label	Label
aa	dh	k	sh
ae	eh	l	t
ah	er	m	th
ao	ey	n	uh
aw	f	ng	uw
ax	g	ow	v
ay	hh	oy	wh
b	ih	p	w
ch	iy	r	y
d	jh	s	z
- (pause)			

Table 6.3: The set of 41 phones that are modeled in word recognition on ATIS.

- The frame-based units are diphone context-dependent. They include all 931 ATIS diphones that have at least 10 tokens in the training data plus one unit to cover all remaining diphones for a total of 932 units.

Other Models

All of the word recognizers use the same duration, pronunciation, and language models:

- The duration model is a word transition weight that is multiplied in at each word transition [67]. The weight is set by minimizing recognition error on the development set.

- The pronunciation model uses only a single pronunciation per word, to allow context-dependent constraints across all phones [70]. Each word sequence must begin with a pause, and pauses can be optionally inserted after each word to make different sequences. Furthermore, in a word sequence, when two of the same phonetic units appear in sequence, they are collapsed into one unit.

- The language model is a bigram. To reduce computation, the vocabulary is limited to the 1078 words which occur at least twice in the training data. As a result, the out-of-vocabulary (OOV) rate on the test set is 0.2%. A bigram language model is trained on the training set and smoothed with a unigram language model and a uniform distribution [2, 54]. The resulting word bigram has a perplexity of 19.7 on the December 1994 set. In testing, the bigram is exponentially weighted, with the weight being set by minimizing recognition error on the development set.

6.2 Near-Miss Segmentation

In comparison to phonetic recognition, it is difficult to explore segmentation in word recognition for three reasons. First, word recognition experiments demand substantially more computation and have relatively slow turnaround. Second, ATIS does not provide reference phone transcriptions for evaluation. Instead, the transcriptions are computed by the recognizer itself and do not provide an independent reference. Third, the selected ATIS phones are not suitable for the acoustic segmentation algorithm. For example, the near-miss word recognizer models a stop closure and burst as a single unit. However, the transition between a stop closure and burst is typically marked by a sharp spectral discontinuity and therefore is difficult to hypothesize as a single unit using an acoustic segmentation algorithm. Due to these difficulties, near-miss segmentation is not compared directly to acoustic segmentation in word recognition. Instead of comparison, this section focuses on characterizing the use of near-miss segmentation in word recognition.

6.2.1 Landmark-Based Recognizer

In experiments in phonetic recognition, a frame-based phonetic recognizer was used to generate segment graphs that were shown to significantly improve performance. However, in preliminary ATIS experiments, these performance improvements did not generalize to word recognition when a similar frame-based phonetic recognizer was used to generate segment graphs. This is hypothesized to be due to the inability of the phonetic recognizer to apply word constraints when creating segments for word recognition. A comparison of forced and best phone paths computed with and without word constraints revealed large inconsistencies that verified the need for applying word constraints during segmentation. As a result, for word recognition, a frame-based word recognizer is used to generate segment graphs.

Unfortunately, a medium vocabulary word recognizer is computationally much more expensive than a phonetic recognizer. To reduce computation during segmentation, this thesis uses a landmark-based word recognizer that considers only a subset of all possible frames. The landmark-based word recognizer developed in this chapter uses a simple spectral change algorithm to detect landmarks. A landmark is detected whenever the Euclidean distance between two spectral frames exceeds a threshold. Across the December 1994 test set, the landmark-based word recognizer detects 30.9 landmarks per second, while the forced alignments computed by the landmark-based word recognizer have 8.1 boundaries per second. In contrast, the frame-based phonetic recognizer used in TIMIT processes all 100 frames per second. As a result, the word recognizer gains a considerable savings in computation. During recognition, the landmark-based word recognizer uses diphone context-dependent acoustic models and a word bigram language model to achieve a 6.2% recognition error rate.

6.2.2 Segmentation

The landmark-based word recognizer is used to generate segment graphs for the remaining word recognition experiments. To reduce computation, the A^* threshold is set conservatively so that only a small number of paths are explored during search, and therefore only small number of segments are included in the segment graphs. Across the December 1994 test set, the segment graphs computed by the landmark-based word recognizers have 18 segments per second, while the forced alignments computed by the landmark-based word recognizer have 7.9 segments per second. In contrast, the frame-based phonetic recognizer produces 32.7 segments per second in TIMIT. As a result, the word segment graphs are expected to sacrifice the performance of the word recognizer.

Figure 6-1 shows a histogram of the number of segments in each near-miss subset across the December 1994 test set. The near-miss subsets are computed with respect to the best path computed by the landmark-based word recognizer. The mean near-miss subset size is 2.3 segments. In contrast, the TIMIT phonetic recognition experiments used 5.0 segments per subset. The histogram shows that most of the near-miss subsets used in word recognition are singular.

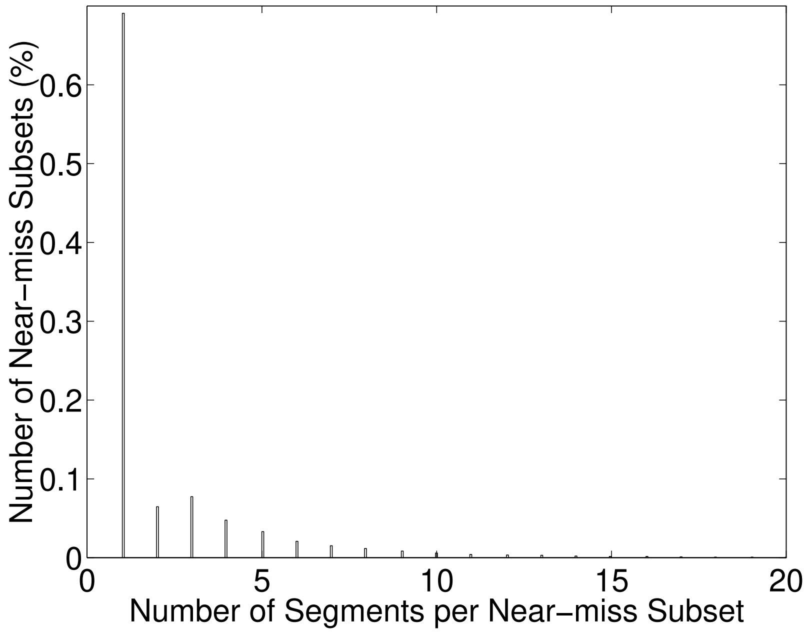


Figure 6-1: This figure shows a histogram of the number of segments in each near-miss subset across the December 1994 test set. The near-miss subsets are computed with respect to the best path computed by the landmark-based word recognizer.

Table 6.4 shows the mean number of segments in the near-miss subsets by phone across the December 1994 test set. Overall, for computational reasons, the experiments in word recognition on ATIS use much smaller segment graphs than the experiments in phonetic recognition on TIMIT.

# Near-Miss	Phones	# Near-Miss	Phones
1.2	wh	2.3	n
1.3	p	2.4	ng
1.4	sh	2.5	a dh h y
1.5	m r	2.6	u v w
1.6	c er l r x	2.8	e g uh
1.7	ah b ih s	2.9	ax iy y
1.8	ch	3.2	jh
2.0	t	3.6	-
2.1	ay eh k o z	3.8	th
2.2	ae aw d f i	4.4	oy

Table 6.4: This table shows the mean number of segments in the near-miss subsets by phone across the December 1994 test set.

6.3 Near-Miss Search

This section explores the use of near-miss modeling in word recognition. The experiments are organized in two sets. The first set explores strategies for assigning near-miss subsets, while the second set explores strategies for modeling near-miss units.

6.3.1 Near-Miss Subsets

The following two sections examine different near-miss assignment strategies.

Overlap

This experiment explores the space of near-miss assignment strategies in which all segments are assigned based on the same relative time, ranging from their begin times to their end times. A near-miss assignment strategy is measured by the temporal overlap of its assignments. The temporal overlap of a segment is measured with respect to a reference on-path segment and is defined as the percentage of the segment that is overlapped by the reference segment. The reference segment sequence for each graph is the best path through the graph found by a landmark-based word recognizer.

Figure 6-2 shows the average temporal overlap per segment over all segments in the December 1994 test set as a function of the relative time that is used in near-miss assignment. The upper dotted line gives an upper bound on the temporal overlap computed by choosing the best assignment for each segment, while the lower dotted line gives a lower bound on the temporal overlap computed by choosing the worst assignment for each segment.

In comparison to analogous TIMIT experiment in Figure 4-4, the sparse segment graphs in ATIS do not allow as much variation in near-miss assignment. Due to the sparser segment graphs in ATIS, the temporal overlaps are larger than they are in TIMIT. However, the trends are the same and verify that in the space of near-miss assignment strategies based on a relative time, the best strategy is to assign each segment based on its midpoint. In fact, the midpoint strategy can achieve close to the optimal temporal overlap that can be achieved when each segment is allowed to freely choose its time to maximize its overlap with respect to its reference on-path segment.

Recognition

The goal of maximizing temporal overlap is based on the hypothesis that a larger degree of temporal overlap may lead to improved recognition performance. To verify this hypothesis, the midpoint and begin time strategies are compared in word recognition using 1-state near-miss units:

- Midpoint assignment results in 8.1% word recognition error rate.
- Begin time assignment results in 9.9% word recognition error rate.

As in TIMIT, the midpoint strategy results in a lower recognition error rate than the begin time strategy. All of the experiments in the remainder of this chapter use

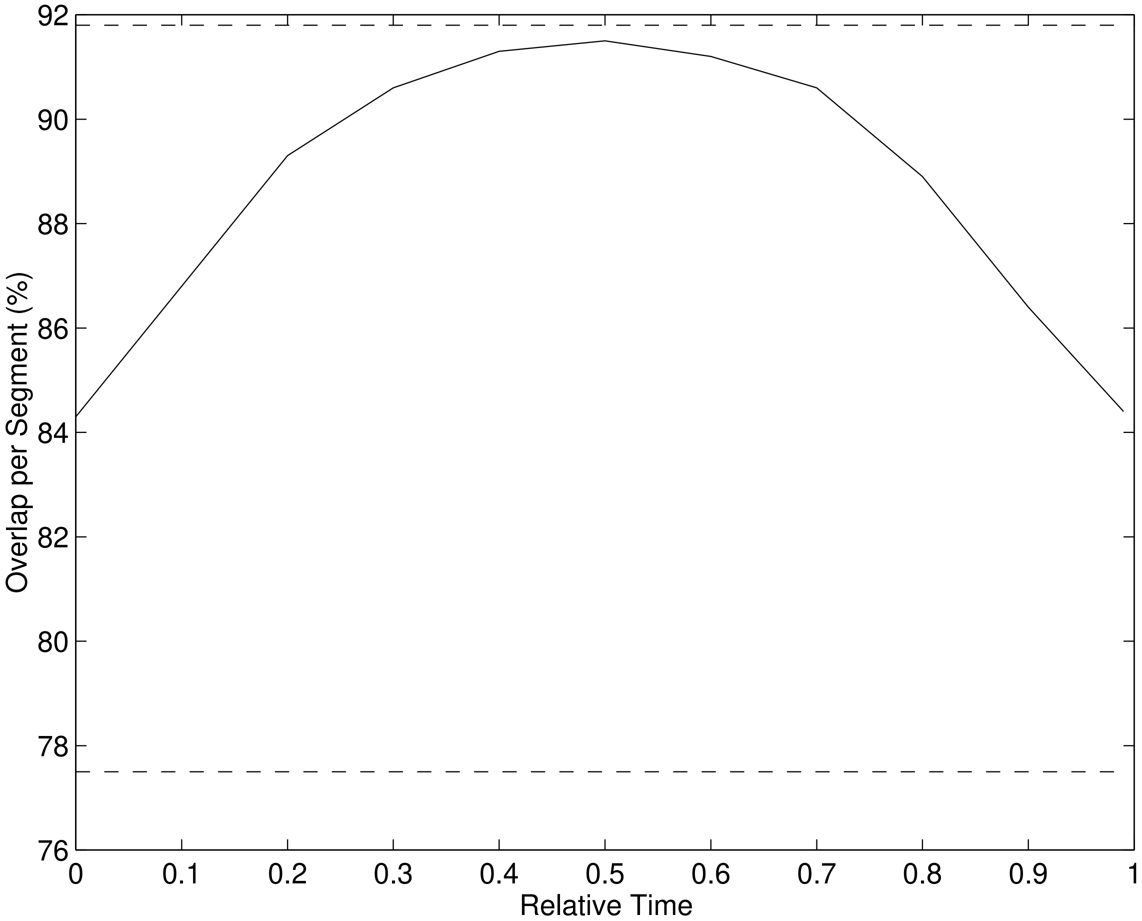


Figure 6-2: This figure shows the average temporal overlap per segment over all segments in the December 1994 test set as a function of the relative time that is used in near-miss assignment. The upper dotted line gives an upper bound on the temporal overlap computed by choosing the best assignment for each segment, while the lower dotted line gives a lower bound on the temporal overlap computed by choosing the worst assignment for each segment.

the midpoint strategy.

6.3.2 Near-Miss Units

This section explores strategies for modeling near-miss units. Table 6.5 shows word recognition error rates over the December 1994 test set for 0-state, 1-state, and 2-state near-miss units. The 3-state near-miss units are not used because the smaller segment graphs do not provide as many tokens for training. To improve robustness, all contextual units are smoothed against a 0-state near-miss unit. As in phonetic recognition, the smoothing weights the higher order models by 0.4 and the 0-state model by 0.6. In word recognition, there are 41 lexical units plus one backoff unit, yielding a total of 1 0-state unit, 42 1-state units, and 83 2-state units. The results show that near-miss modeling can reduce word recognition error rate by modeling phonetic and temporal constraints in off-path segments.

State	Error (%)	Δ (%)
0	10.5	-
1	8.4	20.0
2	8.1	22.9

Table 6.5: This table shows word recognition error rates over the December 1994 test set for 0-state, 1-state, and 2-state near-miss units.

6.3.3 Combined Recognition

This section evaluates the combined near-miss recognizer using both frame- and segment-based models.

Recognition

Table 6.6 shows word recognition error rates over the December 1994 test set for a recognizer using diphone context-dependent landmark-based models with and without the addition of context-independent segment-based models. The segment-based units consist of lexical units and 2-state near-miss units. The addition of context-independent segment-based units to context-dependent landmark-based units reduces word error from by 11.3% from 6.2% to 5.5%.

Unit	Error (%)	Δ (%)
CD Landmark	6.2	-
CD Landmark + CI Segment	5.5	11.3

Table 6.6: This table shows word recognition error rates over the December 1994 test set for a recognizer using diphone context-dependent landmark-based units with and without the addition of context-independent segment-based units. The segment-based units consist of lexical units and 2-state near-miss units.

Error Analysis

This section presents an error analysis of the combined recognizer using context-independent segment-based models and diphone context-independent frame-based models. Table 6.7 shows the breakdown of the recognizer word and sentence error rates into substitution, deletion, and insertion rates.

Level	Error (%)	Sub (%)	Del (%)	Ins (%)
Word	5.5	3.2	1.5	0.8
Sentence	31.4	23.2	12.0	6.6

Table 6.7: This table shows the breakdown of the recognition error rate of the combined recognizer into substitution, deletion, and insertion rates.

Table 6.8 shows the ten most frequent substitutions, deletions, and insertions, along with their frequency of occurrence. As shown, many of the errors involve function words and may not affect understanding.

6.3.4 Comparison

The above evaluation provides a consistent set of experiments which verify that near-miss modeling can improve performance in word recognition. This section compares

Sub			#	Del	#	Ins	#
a	→	the	11	a	36	a	6
and	→	in	8	in	12	the	6
fly	→	flight	8	and	10	York	5
meal	→	meals	5	the	10	two	4
Newark	→	New	5	would	8	to	3
it	→	the	5	I	7	on	3
the	→	these	5	are	6	and	3
I	→	I'd	5	an	4	do	2
a	→	eight	4	how	4	I'm	2
miles	→	tomorrow's	4	now	4	I	2

Table 6.8: This table shows the ten most frequent substitutions, deletions, and insertions for the context-dependent recognizer, along with their frequency of occurrence.

the near-miss modeling system with the seven systems that participated in the last ATIS evaluation [47]. However, this comparison is difficult due to the fact that near-miss modeling uses less complex models than most of these evaluation systems, which typically represent the work of teams of researchers over several years:

- The near-miss modeling system uses gender-independent, context-dependent acoustic models trained on ATIS data only and a bigram language model with a vocabulary of 1078 words.

- The MITRE Glacier system is an HMM system that uses gender- and context-independent acoustic models trained on both ATIS and Resource Management data and a class bigram language model with a vocabulary of 1851 words [3].

- The MIT SUMMIT system is a segment-based system that uses gender- and context-dependent acoustic models trained on ATIS data only and a class quad-gram language model with a vocabulary of 2460 [23].

- The AT&T system is an HMM system that uses gender- and context-dependent acoustic models trained on both ATIS and Wall Street Journal (WSJ) data and a class trigram language model with a vocabulary of 1530 words [4].

- The CMU system is an HMM system that uses gender- and context-dependent acoustic models trained on both ATIS and WSJ data and a class trigram language model with a vocabulary of 3217 words [66].
- The SRI DECIPHER system is an HMM system that uses gender- and context-dependent acoustic models trained on both ATIS and WSJ data and a class trigram language model [10].
- The BBN system, which was also used by Unisys, was not described.

The near-miss modeling system varies in complexity from the others and is difficult to compare. In particular, the near-miss modeling system uses the simplest language model of all systems, a bigram model with neither word classes nor compound words. Table 6.9 summarizes the results reported on the ATIS December 1994 test set. The error rate using the near-miss modeling system is higher than the error rates reported for all other systems except the MITRE system which uses the simplest acoustic models.

System	Error (%)
Near-miss	5.5
MITRE [3]	14.8
MIT [23]	5.2
AT&T [4]	3.5
CMU [66]	3.4
SRI [10]	2.5
BBN [47]	3.5
Unisys [47]	4.1

Table 6.9: This table shows the recognition error rates reported in the last ATIS evaluation in December, 1994.

Near-miss vs. MIT

In comparison to the MIT system, the near-miss modeling system has a slightly higher error rate but uses simpler models, including gender-independent rather than gender-dependent acoustic models and most importantly, a bigram instead of a class quadgram language model. In separate experiments, MIT has reported a 6.5% error reduction with the addition of gender-dependent acoustic models, a 11.8% reduction with the use of a class bigram instead of a bigram language model, and a 17.4% reduction with the addition of a class quadgram language model, for a total 31.9% error reduction [70]. As a result, it is reasonable to conclude that the near-miss modeling system represents an improvement to our previous system.

Near-miss vs. HMM

In comparison to the remaining three HMM systems that are described, the near-miss modeling system has a significantly higher error rate but uses gender-independent rather than gender-dependent acoustic models, trains on only ATIS rather than both ATIS and WSJ data and uses a bigram instead of class trigram language model. In separate experiments, AT&T has reported a 5.7% error reduction with the addition of gender-dependent acoustic models, an 8.8% reduction with the use of a trigram instead of a bigram, a 6.2% reduction with the use of word classes, a 4.5% reduction with the addition of compound words, and a 3.8% reduction with the addition of WSJ data, for a total 21.9% error reduction.

Near-miss vs. AT&T

A better comparison may be to contrast the near-miss modeling system with the AT&T system on the December 1993 test set, which both systems use as a development set and for which both systems report error rates using gender-independent, context-dependent acoustic models trained on ATIS data only and a language model with neither word classes nor compound words. The largest remaining difference is the near-miss modeling system uses a bigram, while the AT&T systems uses a trigram. Nevertheless, the near-miss modeling system reports a 7.1% recognition error rate, while the AT&T system apparently obtains an 8.1% error rate under these conditions. The AT&T error rate is computed from the reported baseline error rate of 10.3%, accounting for an 8.8% reduction with a trigram, a 7.7% reduction with a 16 kHz sampling rate as used in the near-miss modeling system, and a 6.8% reduction with cepstral mean normalization as used in the near-miss modeling system. Overall, this comparison suggests that near-miss modeling is potentially competitive with the state-of-the-art HMM systems.

6.4 Summary

This chapter has characterized and evaluated near-miss segmentation and search on the task of word recognition using the ATIS corpus. Due to computational limitations, the word recognition system uses a simple configuration, in particular small segment graphs and a simple bigram language model with neither word classes nor compound words. A consistent set of experiments show that the relative improvements in phonetic recognition generalize to word recognition. Although it is difficult to directly compare the near-miss modeling system with other systems that participated in the ATIS evaluation, this chapter presents evidence that the near-miss modeling system is a significant improvement from our previous ATIS system which uses significantly more complex models to achieve only a slightly lower error rate. In addition, this thesis presents evidence that the near-miss modeling system achieves a lower error rate than a developmental version of a state-of-the-art HMM system that is more comparable in complexity.

Chapter 7

Conclusion

This chapter summarizes the contributions of this thesis and suggests directions for future research.

7.1 Contributions

The major contributions of this thesis are the near-miss search and segmentation algorithms that together provide the framework for near-miss modeling. The following sections describe the contributions in each of these areas.

7.1.1 Near-Miss Search

The near-miss search is based on a general algorithm for assigning each segment to its own and zero or more additional near-miss subsets. In particular, for each segment, the near-miss search chooses any time in the segment span and assigns the segment to the near-miss subset of any segment that spans the chosen time. Overall, the near-miss search has three salient characteristics.

First, the near-miss search provides a probabilistic framework for segment-based recognition. Probabilistically, a path should account for all of the feature vectors in a graph-based representation. The near-miss search guarantees that the probabilistic framework can be maintained by accounting for the near-miss subsets in any path.

Since each sequence accounts for all times once and only once, each segment must be assigned to each sequence once and only once.

Second, the near-miss search search also provides an efficient implementation of a segment-based search. When extending a segment, the near-miss search can score not only the segment itself but also all of the other segments in its near-miss subset. Such a search can be efficiently implemented using the Viterbi algorithm.

Third, the near-miss search is the first known segment-based framework that has the ability to enforce contextual constraints across all segments in the graph. As a result, the near-miss search can provide a more general framework for the exploration of different modeling strategies.

Empirically, the experiments in this thesis have explored the space of assignments in which all segments are assigned based on the same relative segment time. Within this space, a simple midpoint strategy that assigns all segments based on their midpoints is shown to achieve the lowest recognition error rates. In addition, the experiments have explored three types of contextual constraints that can be modeled in the off-path segments. The first no-state strategy provides a baseline by using a single additional near-miss unit for all off-path segments regardless of context. The second one-state strategy models phonetic constraints by using one additional near-miss unit per lexical unit to enforce that the phonetic context of each off-path segment corresponds to the phonetic context of the on-path segment. In comparison to the baseline case of not modeling off-path context, this strategy for modeling phonetic constraints of off-path segments is shown to reduce the error rate by 6.5% for phonetic recognition and 20% for word recognition. The third multi-state strategies model both phonetic and temporal constraints. These multi-state strategies use multiple additional near-miss units per lexical unit to enforce that the phonetic and temporal context of each off-path segment corresponds to the on-path segment. In comparison to modeling phonetic constraints alone, this strategy for additionally modeling the temporal constraints of off-path segments is shown to further reduce error rate by 4% for both phonetic and word recognition.

7.1.2 Near-Miss Segmentation

The near-miss segmentation algorithm is also based on a general idea for applying all of the constraints in recognition towards segmentation. In particular, the near-miss segmentation runs a recognizer and includes only those segments that are on paths that score within a threshold of the best scoring path. The near-miss segmentation algorithm has several promising characteristics. First, it can potentially apply all of the constraints that are traditionally used in recognition towards segmentation and therefore can generate more accurate graphs. In comparison to our current acoustic segmentation, the near-miss segmentation is shown to generate segment graphs that are both more accurate in terms of alignment and more efficient in terms of size. Second, it only includes the most probable segments and therefore is both adaptive to variation and well-matched to near-miss modeling. In comparison to acoustic segmentation, the near-miss segmentation is shown to generate segment graphs that are less uniform and contain many singular segments. Third, it can use any recognition strategy depending on computational requirements. For example, the near-miss word recognizer uses a landmark-based strategy, rather than a frame-based strategy, to save computation. Fourth, it can be used to generate any type of units. For example, the near-miss word recognizer models a stop closure and release as a single unit rather than two acoustically distinct units. Fifth, near-miss segmentation provides a framework for the combination of frame- and segment-based approaches. In this thesis, a frame-based recognizer is used in the first pass and combined with a segment-based recognizer in the second pass. Finally, it provides useful information to the second pass search. In this thesis, the acoustic and language model scores that are used in frame-based recognition are effectively re-used in segment-based recognition.

7.1.3 Near-Miss Modeling

Although the near-miss search and segmentation algorithms can be applied separately, they are well-matched to provide a cohesive framework for a new segment-based approach to speech recognition referred to in general as near-miss modeling. By overcoming the major impediments to segment-based recognition, near-miss modeling provides a segment-based framework with much unexplored potential. First, near-miss modeling provides a framework in which the relative advantages of frame- and segment-based approaches can be explored. The experiments in this thesis have shown an effective use of a frame-based recognizer for segmentation followed by a segment-based recognizer for acoustic modeling. Second, near-miss modeling provides a framework for the exploration of modeling strategies that model speech using a graph-based representation rather than a sequential representation. Such graph-based modeling strategies are commonly believed to offer potential improvements in speech recognition but are difficult to incorporate within existing frameworks and therefore are not often used. Empirically, only simple modeling strategies have been examined. Nevertheless, near-miss modeling achieves significant reductions in recognition error rate. Most notably, this thesis reports a 25.5% phonetic recognition error rate on the TIMIT core test set over 39 classes that is believed to be the lowest error rate reported on this task. With future research, near-miss modeling is expected to provide even greater improvements.

7.2 Future Work

There are many directions in which this work can be pursued. The following sections describe three general directions of pursuit related to search, segmentation and modeling.

7.2.1 Search

As a new framework, there are many aspects of near-miss modeling that can be studied in greater detail. As described in Chapter 6, many performance sacrifices have been made in the design of the word recognizer in order to enforce context-dependent constraints across all phones and to allow more rapid experimentation. Currently, our group is building better finite state automata (FSA) tools that will significantly speed up computation and eliminate the need for these performance sacrifices [39].

Among other improvements, these FSA tools will allow the incorporation of context-dependent acoustic models in a more efficient manner, the use of more complex pronunciation and language models, and the implementation of a frame-based search to take advantage of efficiencies in frame-based processing. These improvements will enable a more complete evaluation of near-miss modeling in word recognition.

As mentioned in Chapter 4, one extension of near-miss modeling addresses the segmental independence assumption. By definition, the near-miss segments overlap in time and are certainly not independent of one another as assumed in this thesis. It is possible to identify subsets of the near-miss subsets, called joint near-miss subsets, which always appear together across all near-miss subsets and therefore are always scored together. These joint near-miss subsets offer an opportunity to jointly model the correlation across near-miss subsets. The above strategy suggests other criteria than temporal overlap for assigning near-miss subsets. For example, it may be desirable to maximize the number of segments that can be jointly modeled in joint near-miss subsets. Depending on the modeling strategy, it may be desirable to use other near-miss assignment strategies.

7.2.2 Segmentation

Segmentation remains a difficult problem that trades off performance for computation. This section describes three ways in which the tradeoffs may be improved. The near-miss segmentation algorithm is a general framework for segmentation that can use any recognizer to generate a segment graph. The goal of this thesis is to demonstrate an effective framework for segment-based recognition rather than exploring the performance-computation tradeoffs in configuring a first pass recognizer for segmentation. If computation is a concern, it may be useful to explore the tradeoffs in using a landmark-based recognizer rather than a frame-based recognizer. It may also be useful to explore the tradeoffs in using context-independent or broad-class modeling strategies. For word recognition, it may be that the use of an intermediate representation such as syllables, that lies above the phonetic level but below the word level, can provide sufficient constraint without requiring as much computation.

Other than applying more constraint, another method for improving segmentation is to select units that are easier to segment. Alternative units can be linguistic units such as syllables that may account for more phonetic variation. Another alternative unit is a "multi-phone" unit that spans a sequence of acoustically variable phones. Multi-phone units offer a means of accounting for segmentation errors. In addition to improving segmentation, multi-phone units can also improve acoustic modeling by allowing the acoustic model to capture correlation and structure across sequences of acoustically variable phones. In this sense, they can be described as segment-based context-dependent units that span temporal context, in addition to phonetic context.

Another means of accounting for segmentation errors is through pronunciation modeling. Only a single pronunciation was used for all sentences in this thesis. However, research has shown that more complex pronunciation models can improve recognition performance [50, 56]. Many phonological variations are systematic and may be best accounted for through explicit phonological rules [43]. In addition, pronunciation rules can be learned automatically [56].

7.2.3 Modeling

Overall, the motivation for this thesis is to realize the potential of segment-based modeling strategies to improve recognition performance. A segment-based framework offers flexibility in choosing what feature vectors to extract and where to extract them from the speech signal. The recognizers in this thesis use simple cepstral averages and log duration. More complex feature extraction strategies can focus near-miss modeling on characteristics that are important for phonetic contrasts. Near-miss modeling may provide a framework for the incorporation of knowledge-based feature vectors, such as formants [60]. Near-miss modeling may also benefit from automatically generated feature vectors for example by using a generalized feature selection algorithm that combines speech knowledge and automatic learning to maximize discrimination between pairs of confusable phonetic classes [40, 51]. In addition, near-miss modeling provides a framework for hierarchical speech recognition [8, 25]. Rather than extracting the same feature vectors across all phones, a hierarchical strategy can extract different feature vectors for different phones. In near-miss modeling, the first pass recognizer provides the second pass recognizer with a phone hypothesis than can be the basis of hierarchical feature extraction.

Although this thesis demonstrates the effectiveness of near-miss modeling using a simple segment model, other more complex segment models, including models that may not have been effective in other recognition frameworks, may further improve performance within the near-miss modeling framework. For example, statistical trajectory models may better model the trajectory across a segment [14, 24, 46, 58]. Alternatively, discriminative classifiers such as neural networks have been shown to improve performance in HMM systems but may be better incorporated within the near-miss modeling framework [1, 33, 57, 62].

Finally, near-miss modeling suggests that a better framework for modeling and recognizing speech is to use a multi-level graph-based representation rather than a flat sequence-based representation. The near-miss segmentation can be used to generate a multi-level representation, and the near-miss search can be used to process this representation into near-miss subsets which can be directly modeled. This thesis has only explored a small sampling of the many contextual constraints that can be modeled across a segment graph. For example, it is not necessary to require that the on-path segment be modeled as a lexical unit, while the off-path segments be modeled as separate near-miss units. In general, each near-miss subset is a small subgraph that can be modeled in any way. For example, it may be effective to explore the use of a finite state model within this framework. This and other such attractive ideas are critical to the future progress of speech recognition.

Bibliography

[1] S. Austin, G. Zavaliagkos, J. Makhoul, and R. Schwartz. Speech recognition using segmental neural nets. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 625–628, 1992.

[2] L. Bahl, F. Jelinek, and R. Mercer. A maximum likelihood approach to continuous speech recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190, March 1983.

[3] S. Bayer, E. Bernstein, D. Duff, L. Hirschman, S. Luperfoy, and M. Peet. Spoken language understanding: report on the MITRE spoken language system. In *Proceedings of the Spoken Language Technology Workshop*, pages 243–248, 1995.

[4] E. Bocchieri, G. Riccardi, and J. Anantharaman. The 1994 AT&T ATIS CHRONUS recognizer. In *Proceedings of the Spoken Language Technology Workshop*, pages 265–268, 1995.

[5] M. Bush and G. Kopec. Network-based connected digit recognition. *Transactions on Acoustics, Speech and Signal Processing*, 35(10):1401–1413, October 1987.

[6] J. Chang. *Speech Recognition System Robustness to Microphone Variations.* Master's thesis, Massachusetts Institute of Technology, 1995.

[7] Y. Chow and R. Schwartz. The n-best algorithm: an efficient procedure for finding top n sentence hypotheses. In *Proceedings of the Speech and Natural Language Workshop*, pages 199–202, 1989.

[8] R. Chun. *A Hierarchical Feature Representation for Phonetic Classification.* Master's thesis, Massachusetts Institute of Technology, 1996.

[9] G. Chung and S. Seneff. Hierarchical duration modelling for speech recognition using the ANGIE framework. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 1475–1478, 1997.

[10] M. Cohen, Z. Rivlin, and H. Bratt. Speech recognition in the ATIS domain using multiple knowledge sources. In *Proceedings of the Spoken Language Technology Workshop*, pages 257–260, 1995.

[11] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[12] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, August 1980.

[13] P. Denes and E. Pinson. *The Speech Chain*. Anchor Press, 1963.

[14] V. Digalakis. *Segment-based Stochastic Models of Spectral Dynamics for Continuous Speech Recognition*. PhD thesis, Boston University, 1992.

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

[16] W. Fisher, G. Doddington, and K. Goudie-Marshall. The DARPA speech recognition database: specification and status. In *Proceedings of the Speech Recognition Workshop*, pages 93–96, 1986.

[17] G. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, March 1973.

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

[19] M. Gales and S. Young. Segmental hidden Markov models. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 1579–1582, 1993.

[20] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, and N. Dahlgren. DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM. Technical Report NISTIR 4930, National Institute of Standards and Technology, 1993.

[21] J. Glass. *Finding Acoustic Regularities in Speech: Applications to Phonetic Recognition*. PhD thesis, Massachusetts Institute of Technology, 1988.

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

[23] 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 Spoken Language Technology Workshop*, pages 252–256, 1995.

[24] W. Goldenthal and J. Glass. Statistical trajectory models for phonetic recognition. In *Proceedings of the International Conference on Spoken Language Processing*, pages 1871–1874, 1994.

[25] A. Halberstadt and J. Glass. Heterogeneous acoustic measurements for phonetic classification. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 401–404, 1997.

[26] T. Hazen and J. Glass. A comparison of novel techniques for instantaneous speaker adaptation. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 2047–2050, 1997.

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

[28] L. Hetherington and M. McCandless. SAPPHIRE: an extensible speech analysis and recognition tool based on tcl/tk. In *Proceedings of the International Conference on Spoken Language Processing*, pages 1942–1945, 1996.

[29] L. Lamel and J. Gauvain. High performance speaker-independent phone recognition using CDHMM. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 121–124, 1993.

[30] L. Lamel, R. Kassel, and S. Seneff. Speech database development: design and analysis of the acoustic-phonetic corpus. In *Proceedings of the Speech Recognition Workshop*, pages 100–109, 1986.

[31] K. Lee. *Large-Vocabulary Speaker-Independent Continuous Speech Recognition*. PhD thesis, Carnegie Mellon University, 1988.

[32] K. Lee and H. Hon. Speaker-independent phone recognition using hidden Markov models. *Transactions on Acoustics, Speech and Signal Processing*, 37(11):1641–1648, November 1989.

[33] H. Leung, I. Hetherington, and V. Zue. Speech recognition using stochastic segment neural networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 613–616, 1992.

[34] R. Lippman. Speech perception by humans and machines. In *Proceedings of the Workshop on the Auditory Basis of Speech Perception*, pages 309–316, 1996.

[35] F. Liu, R. Stern, A. Acero, and P. Moreno. Environment normalization for robust speech recognition using direct cepstral comparison. In *Proceedings of the International Conference on Spoken Language Processing*, pages II:61–64, 1994.

[36] A. Ljolje. High accuracy phone recognition using context clustering and quasi-triphone models. *Computer Speech and Language*, 8(2):129–151, April 1994.

[37] J. Marcus. Phonetic recognition in a segment-based HMM. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 479–482, 1993.

[38] H. Meng, V. Zue, and H. Leung. Signal representation, attribute extraction and, the use of distinctive features for phonetic classification. In *Proceedings of the Speech and Natural Language Workshop*, pages 176–182, 1991.

[39] M. Mohri and M. Riley. Weighted determinization and minimization for large vocabulary speech recognition. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 131–134, 1997.

[40] M. Muzumdar. *Automatic Acoustic Measurement Optimization for Segmental Speech Recognition*. Master's thesis, Massachusetts Institute of Technology, 1996.

[41] M. Oerder and H. Ney. Word graphs: an efficient interface between continuous-speech recognition and language understanding. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 119–122, 1993.

[42] A. Oppenheim and R. Schafer. *Discrete-time signal processing*. Prentice-Hall, 1989.

[43] B. Oshika, V. Zue, R. Weeks, H. Nue, and J. Auerbach. The role of phonological rules in speech understanding research. *Transactions on Acoustics, Speech and Signal Processing*, 23(12):104–112, January 1975.

[44] M. Ostendorf, V. Digalakis, and O. Kimball. From HMM's to segment models: a unified view of stochastic modeling for speech recognition. *Transactions on Speech and Audio Processing*, 4(5):360–378, September 1996.

[45] M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz, and J. Rohlicek. Integration of diverse recognition methodologies through reevaluation of n-best sentence hypotheses. In *Proceedings of the Speech and Natural Language Workshop*, pages 83–87, 1991.

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

[47] D. Pallett, J. Fiscus, W. Fisher, and J. Garofolo. 1994 benchmark tests for the DARPA spoken language program. In *Proceedings of the Human Language Technology Workshop*, pages 5–36, 1995.

[48] D. Pallett, J. Fiscus, W. Fisher, J. Garofolo, B. Lund, and M. Przybocki. 1993 benchmark tests for the ARPA spoken language program. In *Proceedings of the Human Language Technology Workshop*, pages 15–40, 1994.

[49] M. Phillips and J. Glass. Phonetic transition modelling for continuous speech recognition. *Journal of the Acoustical Society of America*, 95(5):2877, June 1994.

[50] M. Phillips, J. Glass, and V. Zue. Automatic learning of lexical representations for sub-word unit based speech recognition systems. In *Proceedings of the European Conference on Speech Communication and Technology*, 1991.

[51] M. Phillips and V. Zue. Automatic discovery of acoustic measurements for phonetic classification. In *Proceedings of the International Conference on Spoken Language Processing*, pages 795–798, 1992.

[52] K. Ponting and S. Peeling. The use of variable frame rate analysis in speech recognition. *Computer Speech and Language*, 5:169–179, 1991.

[53] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

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

[55] L. Rabiner and R. Schafer. *Digital Processing of Speech Signals.* Prentice-Hall, 1978.

[56] M. Riley and A. Ljolje. Lexical access with a statistically-derived phonetic network. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 585–588, 1991.

[57] A. Robinson. An application of recurrent neural nets to phone probability estimation. *Transactions on Neural Networks*, 5(2):298–305, March 1994.

[58] S. Roucos, M. Ostendorf, H. Gish, and A. Derr. Stochastic segment modelling using the estimate-maximize algorithm. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 127–130, 1988.

[59] M. Russell and R. Moore. Explicit modeling of state occupancy in hidden Markov models for automatic speech recognition. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 5–8, 1985.

[60] P. Schmid. *Explicit N-best Formant Features for Segment-Based Speech Recognition.* PhD thesis, Oregon Graduate Institute of Science and Technology, 1996.

[61] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul. Context-dependent modelling for acosutic-phonetic recognition of continuous speech. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 1205–1208, 1985.

[62] N. Strom. A tonotopic artificial neural network architecture for phoneme recognition probability estimation. In *Proceedings of the Workshop on Speech Recognition and Understanding*, pages 156–163, 1997.

[63] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *Transactions on Information Theory*, 13:260–269, April 1967.

[64] A. Waibel. *Prosody and Speech Recognition.* PhD thesis, Carnegie-Mellon University, 1986.

[65] X. Wang, S. Zahorian, and S. Auberg. Analysis of speech segments using variable spectral/temporal resolution. In *Proceedings of the International Conference on Spoken Language Processing*, pages 1221–1224, 1996.

[66] W. Ward and S. Issar. The CMU ATIS system. In *Proceedings of the Spoken Language Technology Workshop*, pages 249–251, 1995.

[67] V. Zue. The use of speech knowledge in automatic speech recognition. *Proceedings of the IEEE*, 73(11):1602–1614, November 1985.

[68] V. Zue and R. Cole. Experiments on spectrogram reading. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 116–119, 1979.

[69] V. Zue, J. Glass, M. Phillips, and S. Seneff. The MIT SUMMIT speech recognition system: a progress report. In *Proceedings of the Speech and Natural Language Workshop*, pages 179–189, 1989.

[70] V. Zue, S. Seneff, J. Polifroni, M. Phillips, C. Pao, D. Goddeau, J. Glass, and E. Brill. The MIT ATIS system: December 1993 progress report. In *Proceedings of the Spoken Language Technology Workshop*, pages 66–71, 1994.