Multi-Tape Finite-State Transducer for Asynchronous Multi-Stream Pattern Recognition with Application to Speech

by

Han Shu

M.Eng., Massachusetts Institute of Technology (1997)
B.S., Massachusetts Institute of Technology (1996)

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

Doctor of Philosophy
in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2006
[June 2006]

Author .....
Department of
Electrical Engineering and Computer Science
May 23, 2006

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

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

To all the families who have loved and influenced me:

the Shu, Xu, Chen, Ellis and Balaguru families

Multi-Tape Finite-State Transducer for Asynchronous Multi-Stream Pattern Recognition with Application to Speech

by

Han Shu

Submitted to the Department of
Electrical Engineering and Computer Science
on May 23, 2006, in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Electrical Engineering and Computer Science

Abstract

In this thesis, we have focused on improving the acoustic modeling of speech recognition systems to increase the overall recognition performance. We formulate a novel multi-stream speech recognition framework using multi-tape finite-state transducers (FSTs). The multi-dimensional input labels of the multi-tape FST transitions specify the acoustic models to be used for the individual feature streams. An additional auxiliary field is used to model the degree of asynchrony among the feature streams. The individual feature streams can be linear sequences such as fixed-frame-rate features in traditional hidden Markov model (HMM) systems, and the feature streams can also be directed acyclic graphs such as segment features in segment-based systems. In a single-tape mode, this multi-stream framework also unifies the frame-based HMM and the segment-based approach.

Systems using the multi-stream speech recognition framework were evaluated on an audio-only and an audio-visual speech recognition task. On the Wall Street Journal speech recognition task, the multi-stream framework combined a traditional frame-based HMM with segment-based landmark features. The system achieved word error rate (WER) of 8.0%, improved from both the WER of 8.8% of the baseline HMM-only system and the WER of 10.4% of the landmark-only system. On the AV-TIMIT audio-visual speech recognition task, the multi-stream framework combined a landmark model, a segment model, and a visual HMM. The system achieved a WER of 0.9%, which also improved from the baseline systems. These results demonstrate the feasibility and versatility of the multi-stream speech recognition framework.

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

Acknowledgments

This thesis would not have been possible without the help of many people. I want to acknowledge and express my deepest gratitude to the following people.

First, I would like to thank my advisor, Jim Glass, for his guidance and encouragement in my research and for his patience and support, especially when I moved from topic to topic. I am also very grateful to the other members of my thesis committee, Victor Zue, Michael Collins, and Herb Gish for their suggestions and advice.

This thesis would not have been possible without Lee Hetherington's collaboration on a number of key finite-state transducer based algorithms. The finite-state transducer toolkit developed by Lee was also essential for many experiments in this thesis and I want to thank him.

I would like to thank Karen Livescu for her help in providing the baseline PhoneBook recognizer and answering many PhoneBook-related questions. I would also like to thank T. J. Hazen for his help with AV-TIMIT and his 3-stream (landmark, segment, and visual) recognition system.

Everyone in the Spoken Language Systems group have made my life as a graduate student more pleasant. Chao Wang, Lee Hetherington, T. J. Hazen, and Stephanie Seneff have all provided generous advice and assistance. Thanks to Scott Cyphers for answering many of my not-so-short questions on the computing infrastructure, and to Marcia Davidson for running the administrative side smoothly. A very special thanks to Karen Livescu for always being willing to talk about speech recognition, politics, and life. Thanks to Jon Yi and Alex Park for hanging out and research discussions. Thanks to Ed Filisko for all the baking and cooking discussions and advice. Thanks to my other officemates Laura Miyakawa, Ernie Pusateri, Mitchell Peabody, Chih-yu Chao for making the office an intellectually stimulating and fun place to be. Thanks to the rest of SLS students for making SLS a fun place to be, especially to Ken Schutte,

Min Tang, Eugene Weinstein, Ghinwa Choueiter, and Alex Gruenstein for exchanging research ideas and for answering and asking many questions over the years.

My exposure to the interesting research problems of pattern recognition and speech recognition began in the Speech and Language group at BBN. Special thanks to John Makhoul, Rich Schwartz, Herb Gish, Owen Kimball, Spyros Matsouk, Long Nguyen, and Bruce Musicus for the guidance and helpful discussions.

Thanks to Youssef Marzouk and Richard Diaz for the years of support and friendship, and for their detailed comments on parts of this thesis. Thank you to Damani Walton for grabbing those late night meals with me even with zero minute notice, after I had spent hours slaving away at my thesis. Thanks to all my Chi Phi brothers for all the years of fun and support since my undergraduate days and continuing.

Thanks to my mother and father for sacrificing so much to immigrate to the US so that my brother and I could have a better (modern) education. Thanks to my brother for taking care of me and watching out for me often when I were not even aware. Thanks to Jichen Zhu for being helpful to me in many ways than she probably acknowledges. Thanks to the Ellis and Chen families who helped to raise me and installing wonderful values.

Thanks to my former headmaster Bernier Mayo at St. Johnsbury Academy for welcoming my brother and I to his wonderful high school in the Northeast Kingdom of Vermont. The great education we received there laid the foundation for the years of learning to come.

Finally, a special thanks to my wife Soundhari for her love and support since the first day we met on her Wellesley graduation day. Life in graduate school would have been much less fun without her.

Han Shu
May 24, 2006

This research was supported by DARPA under contract N66001-99-1-8904 monitored through Naval Command, Control and Ocean Surveillance Center and by the NSF under award 9872995-1483.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic speech recognition (ASR) remains one of the "holy grails" in the field of artificial intelligence. Despite substantial improvement over the last two decades, in part due to the use of mathematically rigorous modeling techniques, there still remains a significant performance gap between humans and machines [44].

Figure 1-1 illustrates the main processing stages of a state-of-the-art speech recognition system. The feature extraction module, the lexicon, the acoustic model module are the key modules for acoustic-phonetic modeling. The feature extraction module processes the input speech waveform to produce a sequence of feature vectors for robust ASR. The feature vectors should ideally maximize acoustic-phonetic difference while minimizing the differences due to individual speaker characteristics and the acoustic environment. The acoustic model contains parameters of the acoustic-phonetic classes learned from the feature vectors of the training set. The output feature vectors are mapped to a linear sequence of sub-phonetic or phonetic models. The lexicon holds mappings between words and their phonetic spellings. The language model typically characterizes the relative frequencies of the word sequences to be recognized. The decoder outputs the best word sequence with the input feature vector sequence, the acoustic model, the lexicon, and the language model.

It is clear that improvements in both acoustic-phonetic modeling and language modeling are needed to bridge the performance gap between human and machines. The performance gap on tasks where contextual knowledge cannot help, e.g., recog-

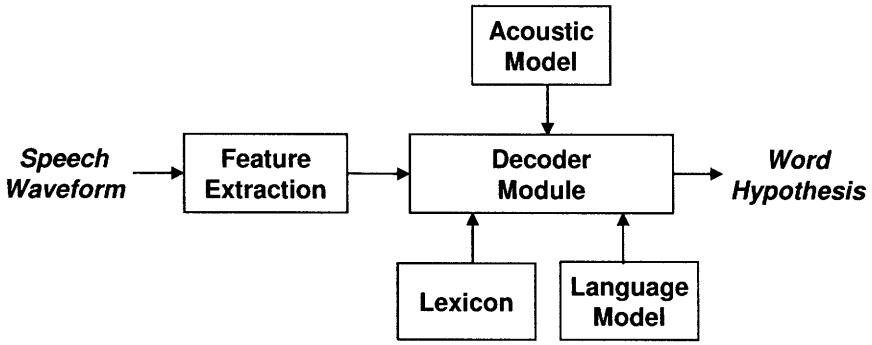


Figure 1-1: Flow chart for a typical state-of-the-art speech recognition system.

nizing isolated phones or nonsensical utterances [44], highlights the human's superior ability of acoustic-phonetic modeling.

Extensive research has been done to optimize the types of features extracted and the types of mathematical models used for modeling the feature vectors. The state-of-the-art ASR system uses a single-stream of frame-based features and hidden Markov models (HMMs) as its mathematical models. Despite significant advances in the search for the "optimal" features and training and decoding algorithms for various types of mathematical models, the equivalent human system still far outperforms the best machine versions. Many speech researchers agree that the paradigm of optimizing a single-stream of features modeled by HMMs will not ultimately lead to human-level performance [62].

In this thesis, we have developed a multi-stream speech recognition framework with multi-tape finite-state transducers. We first formulated a probabilistic recognition framework with multi-tape finite-state transducers, then we constructed the missing algorithms for the framework. Finally, we applied the framework to two different recognition tasks with a multiple streams of features. From these experiments, we demonstrated that this multi-stream speech recognition framework with multi-tape finite-state transducers is able to flexibly accommodate a large class of multi-stream features.

This thesis has been motivated by previous works on both single-stream and multi-stream speech recognition. In the following sections, we will first discuss the motiva-

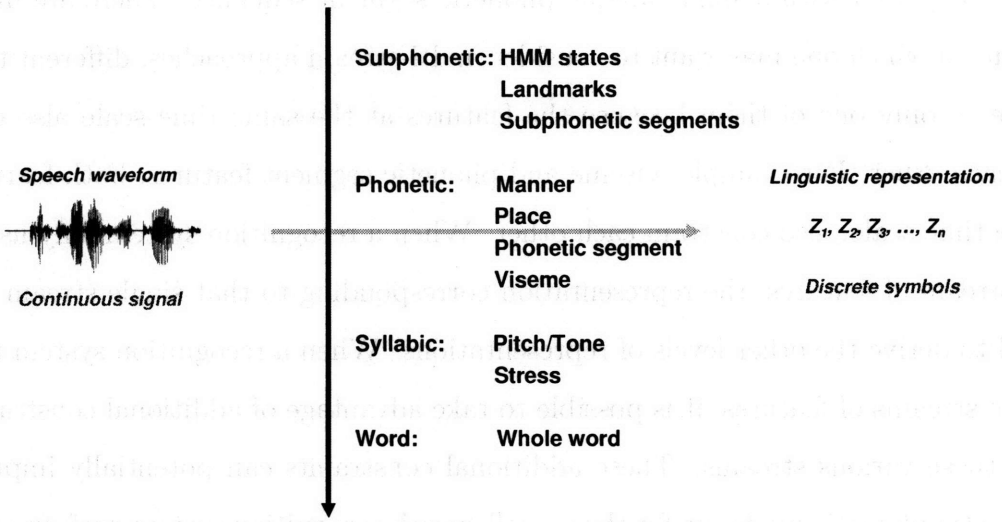


Figure 1-2: Various feature streams for automatic speech recognition.

tion in detail, then we will highlight some of the key issues the proposed multi-stream speech recognition framework must accommodate. These have both inspired and guided the formulation of the new multi-stream speech recognition framework using multi-tape finite-state transducers.

1.1 Motivation

1.1.1 Hierarchical Feature Representations

The process of speech recognition can be thought of as a decoding process which maps continuous speech signals to the underlying discrete linguistic representations such as words. For automatic speech recognition, various types of single stream features have been used. Figure 1-2 illustrates the type of features that can be extracted at various time scales: sub-phonetic, phonetic, syllabic, and word-levels. The sub-phonetic features, such as fixed frame-rate Mel-frequency Cepstral coefficients (MFCCs) [13], are at the finest time scale, and the features at the word level are at the coarsest time scale. The features at various time scales can be organized hierarchically.

The representations at the various time scales constrain each other. For example, the word sequence limits the set of phonetic segments. Similarly, the landmark

sequence is paired with a single unique phonetic segment sequence. There are many occasions in which one may want to consider multi-stream approaches, different time scale being only one of them because the features at the same time scale also constrain each other. For example, viseme and phonetic segment features, both features at same time scale, also constrain each other. When a recognition system only uses a single stream of features, the representation corresponding to that single stream can be used to derive the other levels of representations. When a recognition system uses multiple streams of features, it is possible to take advantage of additional constraints among these various streams. These additional constraints can potentially improve the acoustic-phonetic modeling for the overall speech recognition system performance.

In the state-of-the-art HMM systems, only a single stream of features is commonly used. The features are typically computed at the sub-phonetic time scale. Other types of features are not typically used simultaneously, so these systems do not attempt to exploit the constraints among the various features. In this thesis, we will develop a multi-stream framework to investigate whether applying the constraints can improve the overall recognition performance.

1.1.2 Previous Work: Multi-stream Speech Recognition

The information associated with individual streams of features can be combined either before or after the search performed by the decoder module. Approaches for multi-stream speech recognition can be divided into two main categories: *early integration* and *late integration.*

In the early integration approaches, the individual streams are *stacked* together to form a single stream of feature vectors. The dimension of the resulting feature streams is the sum of the dimensions of the individual feature streams. When the individual feature streams are time synchronous (e.g., fixed frame-rate features with the same frame rates), the stacking procedure is straightforward. However, when the feature streams are asynchronous (e.g., variable frame-rate features, or fixed frame-rate features with the different frame rates), simply stacking the features may be impossible. In addition, if the individual feature streams correspond to different

feature spaces (e.g., phonemes and visemes), combining the various feature spaces may also prove difficult.

For the late integration approaches, searches are performed on the individual feature streams, and the individual feature spaces are combined. The combination can also be performed at the phone or word level. *ROVER* [22] is a late integration method where the combination is done at the word hypothesis level. While late integration methods at the word hypothesis level like ROVER are simple to implement and low in computational cost, ROVER does not take advantage of all possible constraints among the feature spaces corresponding to the individual feature streams.

The segmental speech recognition system at MIT [27] integrates two feature streams, *landmarks* and *segments*. While the landmark feature stream is a linear sequence like MFCCs, the segment feature stream must be represented by a directed acyclic graph (DAG). The integration is done at the phonetic level, and synchronization between the two feature streams are enforced at phonetic boundaries during the search.

The multi-stream speech recognition by HMM recombination by Bourlard, Dupont, et al. [3, 4, 17, 18] and the multi-rate HMM framework of Çetin and Ostendorf [5] are examples of late integration methods where the integration is not done at the word hypothesis level. In [3, 4, 17, 18], the individual features streams are modeled together in a network. The different streams are represented by different HMMs, and the HMMs are connected together with special synchronization states. Between the special synchronization states, the individual feature streams are modeled by the HMMs independently.

The multi-rate HMM framework of Çetin and Ostendorf [5] can model feature streams of both fixed and variable frame rates. These streams can also be of different rates. The individual feature streams are modeled with HMMs. The parameters for the individual HMMs are trained separately. Graphical models [43] are used to model the constraints among the feature streams for decoding.

Both the multi-stream speech recognition by HMM recombination and the multi-rate HMM framework are flexible frameworks for multi-stream speech recognition.

Both frameworks can be used to specify the various constraints among the feature streams, and both can accommodate a large class of feature representations. However, they are not able to support the features represented by directed acyclic graphs such as segment features.

1.2 Asynchrony in Multi-stream Speech Recognition

1.2.1 Asynchrony in Audio-Visual Speech Recognition

Audio-visual speech recognition (AVSR) refers to speech recognition with both the usual acoustic speech signal and the video signal of the speaker's face, or at least of the mouth region. Within the last few years, AVSR has become a very active research area [8, 18, 19, 29, 31, 55]. The additional modality of the video images often improves the overall recognition performance. This improvement can be especially significant for speech in noisy environments. These two complementary input modalities are often modelled as two separate feature streams, one for the audio stream, and the other for the visual stream. Many researchers have found that these two feature streams are asynchronous [19, 28].

Figure 1-3 shows the spectrogram and the corresponding lip images of the spoken phrase "chosen few." In between, the reference time alignment of the phone sequence is also displayed. In this example, asynchrony can be seen during the phone [en] and the phone [f]. The lip image corresponding to the phone [en] shows that the lips are already in the position for producing the phone [f].

1.2.2 Asynchrony Between Frame-based and Landmark Features

For a frame-based speech recognition system using HMMs, the exact phone boundary locations do not play a role in the computation of the features. However, as part of

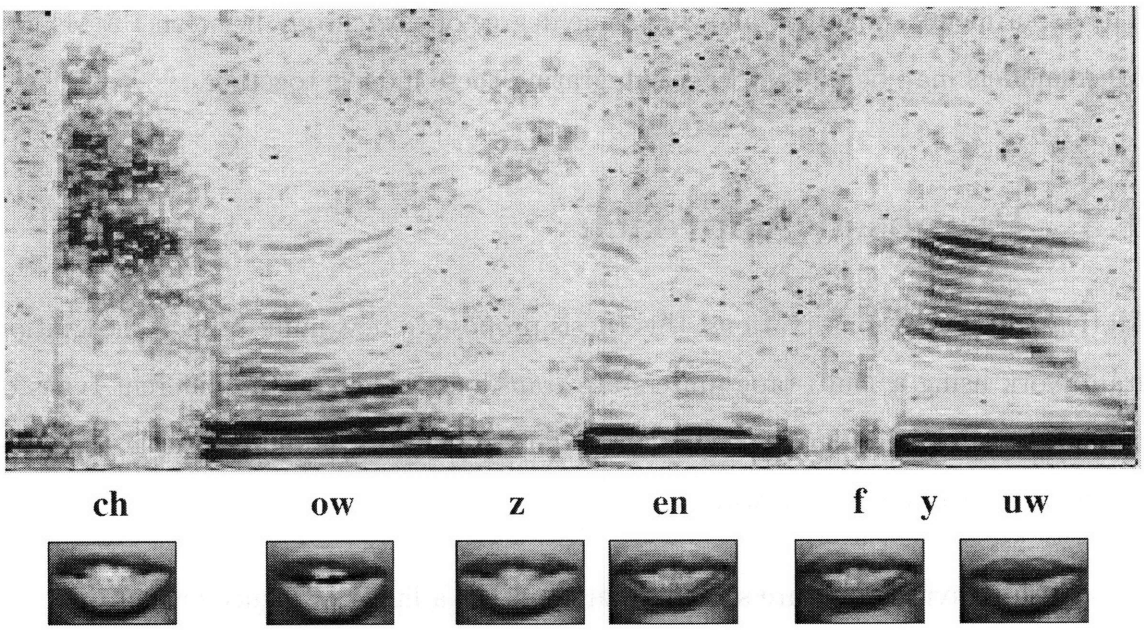


Figure 1-3: Asynchrony in audio-visual speech recognition. Audio-visual example of "chosen few" adapted from Saenko et al. [58]. From top to bottom, the panels are: spectrogram; reference time alignment of the phone sequence; and lip images.

the decoding process, the phone boundaries are implicitly computed in conjunction with discovering the best word sequence. HMM-based systems are optimized to maximize recognition performance, not necessarily the accuracy in the phone boundary locations. Toledano et al. have reported that the phonetic alignments preferred by the context-dependent or context-independent HMMs are not consistent with human transcribed phone boundaries [48, 69].

In contrast, the exact phone boundary locations affect the landmark feature computation in a segment-based landmark system. In this system, the phone boundary locations are hypothesized first, before the computation for the landmark features. Anecdotally by comparing landmark-based phone boundary locations with manually transcribed ones, landmark-based phone boundary locations are better aligned to the human transcribed phone boundaries than the ones hypothesized with HMMs.

The phonetic boundary locations preferred by the HMM-based system and by the segment-based landmark system are different. We carried out a set of experiments to test whether it is important to accommodate this difference when combining these two types of features. The details of the experiments are described in Section 6.2.

The experiments show that allowing some degree of asynchrony between HMMs and other models may be critical when integrating these models together.

1.3 Proposed Approach

Motivated by these observations, this thesis proposes a new multi-stream recognition framework using a multi-tape finite-state transducer to model the different types of feature streams at different time-scales. This framework is more flexible than the previous approaches in two ways:

- The individual feature streams can be either a linear sequence or a graph.
- The asynchrony across the feature streams is controllable by a multi-tape finite-state transducer.

The graph features are important for segment-based systems. They are more general than linear sequence features. The ability to accommodate both types of features enables the multi-stream framework to support a bigger class of combination of features. The proposed multi-stream framework uses the multi-tape finite-state transducer formalism to specify the various constraints. Both the finite-state transducer and the multi-tape finite-state transducer will be introduced in detailed in the later chapters.

Before we developed the multi-stream recognition framework, we first formulated a single-stream recognition framework using an FST cascade for features that can be either a linear sequence or a graph. With the single-stream recognition framework, we generalized the single-stream framework to the multi-stream framework by using a multi-tape finite-state transducer.

1.4 Contributions

The primary contributions of this thesis are:

- We formulated a multi-stream recognition framework with a multi-tape finite-state transducer. This multi-stream framework accommodates multiple streams of features which can be a mixture of sequential and graph features, and it also allows controllable asynchrony across the feature streams. We demonstrated the capabilities on the WSJ task with HMM frame-based features and segment-based landmark features and on a audio-visual recognition task with HMM frame-based features and segment-based landmark and segment features.

- We introduced a single-stream recognition framework based on the finite-state transducer cascade with support for both sequential and graph features. With the existing beam search and newly developed EM-based training for this framework, it freed the dependency on initialization models for the framework and enabled direct comparison among various kinds of recognition systems (e.g., frame-based and segment-based) supported by the framework.

- We developed a novel EM-based weight training algorithm for learning FSTs weights from data. We applied this algorithm for the problem of learning pronunciation weights for the FSTs inside the FST cascade, we showed improved recognition performance with learned pronunciation weights over the unweighted baseline system.

1.5 Thesis Outline

The remainder of this thesis is structured as follows. Chapter 2 provides relevant background information. Chapter 3 formally introduces weighted finite-state transducers and constructs a unifying probabilistic framework for single-stream recognition with frame-based and segment-based acoustic models. Chapter 4 develops a novel EM-based weight training algorithm for finite-state transducer weights. We also present experimental results of this algorithm for the problem of learning pronunciation weights from training data. Chapter 5 formulates a novel EM-based training for acoustic models represented with finite-state transducers. We also present experimen-

tal results for both frame-based and segment-based acoustic models training with this algorithm. Chapter 6 formally introduces weighted multi-tape finite-state transducers and associated algorithms. Chapter 7 presents the multi-stream recognition framework with multi-tape finite-state transducers. We also show through experiments the flexibility of the framework for modeling multiple streams of features. Finally, Chapter 8 summaries the thesis and discusses future directions and conclusions.

Chapter 2

Background

This chapter provides a brief introduction to automatic speech recognition for both frame-based and segment-based approaches and the finite-state transducers. In Section 2.1 we will first review the standard probabilistic formulation for the frame-based approach using hidden Markov models (HMMs), then we will discuss the acoustic and language models and the associated training and decoding algorithms. In Section 2.2, we will highlight aspects of the segment-based system that is different from the frame-based approach. In Section 2.3, we will first formally define FSTs and semirings, then we will discuss the probabilistic interpretation of FSTs, including the FST operations needed to convert a joint probability transducer to a conditional probability transducer. Finally we will illustrate how various constraints are represented by FSTs in a typical speech recognition systems.

2.1 Automatic Speech Recognition

In the typical formulation for automatic speech recognition, the goal is to find the sequence of words $\vec{W}^* = \{w_1, w_2, \ldots, w_M\}$ which gives the maximum *a posteriori* probability given the acoustic observations $\vec{O} = \{o_1, o_2, \ldots, o_N\}$, that is:

$$\vec{W}^* = \arg\max_{\vec{W}} P(\vec{W}|\vec{O}). \tag{2.1}$$

With Bayes' rule,

$$\vec{W}^* = \arg\max_{\vec{W}} \frac{P(\vec{O}|\vec{W})P(\vec{W})}{P(\vec{O})} \tag{2.2}$$

$$= \arg\max_{\vec{W}} P(\vec{O}|\vec{W})P(\vec{W}) \tag{2.3}$$

$$= \arg\max_{\vec{W}} P(\vec{O},\vec{W}) \tag{2.4}$$

where $\vec{W}$ ranges over all possible word sequences.

In most ASR systems, a sequence of sub-word units, $\vec{U}$, and a sequence of sub-phone states, $\vec{S} = \{s_1, s_2, \ldots, s_N\}$, are decoded along with the optimal word sequence. Equation 2.4 becomes:

$$\vec{W}^* = \arg\max_{\vec{W}} \sum_{\forall \vec{S},\vec{U}} P(\vec{S},\vec{U},\vec{W},\vec{O}) \tag{2.5}$$

$$\approx \arg\max_{\vec{W},\vec{S},\vec{U}} P(\vec{S},\vec{U},\vec{W},\vec{O})). \tag{2.6}$$

The approximation in this equation is commonly known as the "Viterbi approximation." The expression $P(\vec{S},\vec{U},\vec{W},\vec{O})$ can be decomposed into the form with the chain rule of probability:

$$P(\vec{S},\vec{U},\vec{W},\vec{O}) = P(\vec{O}|\vec{S},\vec{U},\vec{W})P(\vec{S}|\vec{U},\vec{W})P(\vec{U}|\vec{W})P(\vec{W}). \tag{2.7}$$

With appropriate conditional independence assumptions,

$$P(\vec{S},\vec{U},\vec{W},\vec{O}) = P(\vec{O}|\vec{S})P(\vec{S}|\vec{U})P(\vec{U}|\vec{W})P(\vec{W}). \tag{2.8}$$

Thus, Equation 2.6 becomes,

$$\vec{W}^* = \arg\max_{\vec{W},\vec{S},\vec{U}} P(\vec{O}|\vec{S})P(\vec{S}|\vec{U})P(\vec{U}|\vec{W})P(\vec{W}). \tag{2.9}$$

Note that the people familiar with HMMs may not be use to this formulation. This formulation is similar to the unified view of the frame-based and segment-based

approaches presented in by Ostendorf et al. [53]. The term $P(\vec{O}|\vec{S})$ is the *feature observation model.* The term $P(\vec{S}|\vec{U})$ is a weighted mapping between the sequences of sub-word units to sequences of sub-phone units, and we will refer to it as *model topology.* The term $P(\vec{U}|\vec{W})$ is the *pronunciation model* which describes the sequences of sub-word units that can be generated for a given word sequence, typically accomplished by a dictionary lookup table and phonological rules to model systematic phonological variations in fluent speech. Sometimes $P(\vec{O}|\vec{W}) \approx P(\vec{O}|\vec{S})P(\vec{S}|\vec{U})P(\vec{U}|\vec{W})$ is referred to as the *acoustic model*, and $P(\vec{W})$ is the *language model.*

2.1.1 Language Model

$P(\vec{W})$ models the relative frequencies of word sequences. Common types of language models are finite-state grammar (or context-free grammar) and n-gram models. Both types can also be used together [50, 71]. Finite-state grammars are often used for recognition tasks with a small vocabulary, and are manually created. The statistical language model $P(\vec{W})$ can be factored with the chain rule of probability:

$$P(\vec{W}) = P(STOP|w_1, w_2, \ldots, w_N) \prod_{i=1}^{N} P(w_i|w_1, w_2, \ldots, w_{i-1}), \tag{2.10}$$

where $STOP$ denotes the termination symbol at the end of a word sequence, and N denotes the length of the word sequence $\vec{W}$. [11] contains a detailed discussion on why the $STOP$ symbol is needed. n-gram models assume that the current word w_i is only dependent of the $n-1$ previous words, that is:

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) = P(w_i|w_{i-n+1}, w_{i-n+2}, \ldots, w_{i-1}). \tag{2.11}$$

Thus,

$$P(\vec{W}) = P(STOP|w_N, w_{N-1}, \ldots, w_{N-n+2}) \prod_{i=1}^{N} P(w_i|w_{i-n+1}, w_{i-n+2}, \ldots, w_{i-1}). \tag{2.12}$$

$P(w_i|w_{i-n+1}, w_{i-n+2}, \ldots, w_{i-1})$ are the parameters of the n-gram model. The trigram and bigram are the most popular language models in the state-of-the-art systems. They are typically trained from a text corpus using a number of standard language modeling toolkits [9, 66].

2.1.2 Features

The acoustic observations $\vec{O}$ are acoustic features extracted from the speech waveform. For multimodal speech recognition or visual speech recognition, the acoustic features can also be extracted from the video data of the speaker, primarily from around the mouth region. Typically the feature vectors in HMM-based speech recognition systems consists of Mel-frequency Cepstra coefficients (MFCCs) [13] and their first and second differences. Sometimes energy is used in place of the 0^{th} MFCC. The MFCCs are typically computed with a Hamming sampling window of about 25ms in duration. The first and second differences are estimated over several of these window frames from the MFCCs. In this thesis, the MFCCs of the frame-based feature vectors are 14-dimensional. Together with the deltas and the delta-deltas [24], the feature vectors are 42-dimensional in total. These feature vectors are computed at a fixed frame-rate, most commonly at 10ms/frame. Because these features vectors are computed on a frame by frame basis, they are often referred to as *frame-based* features. It is important to note that these acoustic observation vectors form a *single*, *linear* temporal sequence. Since the duration of a typical phone can vary from 20ms to over 200ms, the number of fixed frame-rate feature vectors within the same phonetic segment is usually much greater than one. These feature vectors within the same phonetic segment are typically highly correlated. However, HMMs have an inherent conditional independence assumption on the observation feature vectors. Thus, the fixed frame-rate feature vector employed by HMM-based recognizers fundamentally limits the range of acoustic models that can be explored for encoding acoustic-phonetic information. While many researchers have focused on improving frame-based HMM ASR systems, some have tried to avoid this limitation by constructing segment-based ASR systems [15, 26, 53]. We will discuss these segment-based feature vectors in detail in

Section 2.2.

2.1.3 Acoustic Models

The acoustic models $P(\vec{O}|\vec{W})$ in Equation 2.4 have three components, the feature observation model $P(\vec{O}|\vec{S})$, the model topology $P(\vec{S}|\vec{U})$, and the pronunciation model $P(\vec{U}|\vec{W})$. In the context of a typical HMM-based system, the pronunciation model $P(\vec{U}|\vec{W})$ is simply based on a dictionary lookup and is not weighted. The sub-word units $\vec{U}$ are either context-independent or context-dependent phone models. The sub-phone units $\vec{S}$ correspond to the individual HMM states. The model topology $P(\vec{S}|\vec{U})$ is typically a 3- or 5- state left-to-right model with optional skip states. Figure 2-1 illustrates a 3-state model with a skip transition. The weights in the model topology are the same as the "state transition probabilities" of the HMMs. The feature observation models $P(\vec{O}|\vec{W})$ correspond to the state observation probability distribution functions (PDFs) of HMMs.

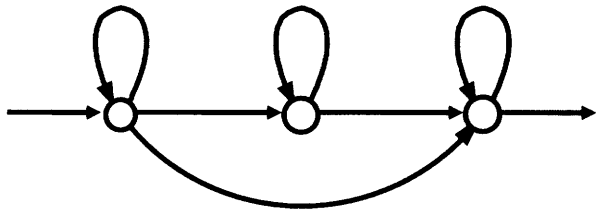


Figure 2-1: A 3-state left-to-right hidden Markov model with a skip transition from the first state to the last state.

The feature observation model $P(o_i|s_j)$ is typically in the form of Gaussian mixture models (GMMs) because of their modeling power and their computational efficiency. $P(o_i|s_j)$ with M Gaussian components is expressed as,

$$P(o_i|s_j) = \sum_{k=1}^{M} c_{j,k}\ P(o_i|g_{j,k}), \tag{2.13}$$

where

$$\sum_{k=1}^{M} c_{j,k} = 1. \tag{2.14}$$

Each component $P(o_i|g_{j,k})$ is a Gaussian density function,

$$P(o_i|g_{j,k}) = N(\vec{\mu}_{j,k}, \Sigma_{j,k}) = \frac{1}{(2\pi)^{D/2}\,|\Sigma_{j,k}|^{1/2}} e^{-\frac{1}{2}(o_i-\vec{\mu}_{j,k})^T \Sigma_{j,k}^{-1}(o_i-\vec{\mu}_{j,k})}, \tag{2.15}$$

where D is the dimension of the feature vector o_i.

2.1.3.1 Parameter Learning for Acoustic Models

All the parameters in the acoustic models can be learned from a set of training acoustic data. The training problem is typically formulated as an optimization problem, maximizing an objective function via the acoustic model parameters. Various types of criteria, such as maximum likelihood (ML) [56], maximum mutual information (MMI) [70], and minimum probability of error (MPE) [49] have been explored with various degree of success. ML training assumes a generative model of the underline stochastic processing. In contrast, MMI and MPE training does not assume a generative model, and are often referred to as discriminative training methods. Within the last few years, a number of researchers have reported better performance using discriminative training methods with the additional cost of algorithmic complexity and training time [70]. For simplicity in this thesis, we will focus only on the ML criterion for the multi-stream speech recognition framework. In the future, exploring the discriminative training methods for the multi-stream recognition framework will be an interesting research direction.

Let Θ be the set of parameters to be learned and $\vec{X}$ be the set of training examples. Let $\mathcal{L}(\vec{X}|\Theta)$ be the likelihood function. The parameter learning problem is converted into the problem of finding the optimal Θ^* where,

$$\Theta^* = \arg\max_{\Theta} \mathcal{L}(\vec{X}|\Theta). \tag{2.16}$$

For the problem of learning parameters for HMMs, this optimization problem cannot be solved analytically. For these types of cases, one can use the *Expectation-Maximization* (EM) algorithm [14]. The EM algorithm is an iterative procedure

which is guaranteed to find a local maximum. In the following two sections, we will first review how to learn the parameters of GMMs with the EM algorithm, then we will show how EM is used for training HMM model parameters.

2.1.3.2 EM Training of Gaussian Mixture Models

First we review the EM training of Gaussian mixture models from a set of training data [2], $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$. From Equation 2.13, the set of parameters for a GMM is $\Theta = \{c_{j,k}, \vec{\mu}_{j,k}, \Sigma_{j,k}\}$. The key insight for this problem is that the variable k is the only *unobservable* part of the data. The variable k specifies which Gaussian component the data sample is generated from. If k is known for every training data sample, then the parameter set Θ can be easily estimated. The standard EM algorithm learns the parameter set Θ from the training data set in two steps [2]. First, the *expectation step (E step)* determines the posterior probability that each Gaussian mixture component could have been generated from each data point. This posterior probability is expressed as:

$$P(k|\vec{x}_n, g_{j,k}) = \frac{c_{j,k}\, p(\vec{x}_n|g_{j,k})}{\sum_{l=1}^{M} c_{l,k}\, p(\vec{x}_n|g_{l,k})}. \tag{2.17}$$

Second, the *maximization step (M step)* re-estimates the parameters set based on the posteriors calculated in the E step. The parameter set $\{c_{j,k}, \vec{\mu}_{j,k}, \Sigma_{j,k}\}$ is re-estimated for each mixture component on the entire training set according to the following equations:

$$c_{j,k}^{new} = \frac{1}{N} \sum_{n=1}^{N} P(k|\vec{x}_n, g_{j,k}), \tag{2.18}$$

$$\vec{\mu}_{j,k}^{new} = \frac{\sum_{n=1}^{N} P(k|\vec{x}_n, g_{j,k})\, \vec{x}_n}{\sum_{n=1}^{N} P(k|\vec{x}_n, g_{j,k})}, \tag{2.19}$$

$$\Sigma_{j,k}^{new} = \frac{\sum_{n=1}^{N} P(k|\vec{x}_n, g_{j,k})\, (\vec{x}_n - \vec{\mu}_{j,k}^{new})\, (\vec{x}_n - \vec{\mu}_{j,k}^{new})^T}{\sum_{n=1}^{N} P(k|\vec{x}_n, g_{j,k})}. \tag{2.20}$$

The Gaussian components are initialized by a well-known procedure called "split and merge" [72]. First, a single Gaussian is learned from all the training data. Over

subsequent iterations, each Gaussian component is split into two, then the GMM is trained with the EM algorithm, and finally Gaussian components are merged with others if there are not enough data samples associated with those component. This is repeated until a desired number of Gaussians is reached. Figure 2-2 outlines the "split and merge" procedure.

```
1   C ← minimum required number of data samples belonging to a Gaussian component
2   T ← target number of Gaussians
3   estimate a single Gaussian from all of the training data
4   M ← 1
5   while M < T
6       /* split every Gaussians component */
7       for k ← 1 to M
8           g_{j,k+M} ← g_{j,k}
9           c_{j,k+M} ← (1/2) c_{j,k}
10          c_{j,k} ← (1/2) c_{j,k}
11          shift μ_{j,k} along the direction of the largest variance of Σ_{j,k}
12          shift μ_{j,k+M} along the negative direction of the largest variance of Σ_{j,k}
13      end
14      M ← 2M
15
16      /* EM train the Gaussian mixtures */
17      Update variables P(k|x_n, g_{j,k}), c_{j,k}^{new}, μ_{j,k}^{new}, and Σ_{j,k}^{new} according to Equations 2.17-2.20
18
19      /* merge Gaussian components if needed */
20      for k ← 1 to M
21          if Σ_{n=1}^{N} P(k|x_n, g_{j,k}) < C
22              remove g_{j,k} from the set of Gaussian components
23              q ← index of Gaussian component closest to this k^{th} component
24              c_{j,q} ← c_{j,q} + c_{j,k}
25              M ← M − 1
26          end
27      end
28  end
```

Figure 2-2: Pseudo-code for the "split and merge" procedure.

A procedure called "K-means clustering" [16] can also be used for the initialization. Since the first step of K-means clustering is a random initialization of the centroids, the resulting Gaussian mixture models can vary in performance from different initializations. Experimentally the split and merge procedure matches the best

performance of multiple training runs with different K-means initializations. For this thesis, only the split and merge procedure will be used.

2.1.3.3 EM Training of Hidden Markov Models

Now we will review the EM training of HMMs from a set of training acoustic data $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$ [2, 56]. Let Θ be the set of parameters for the HMMs. Typically the training acoustic data is labelled with the reference word transcriptions. The detailed time alignment information between the reference words and the observation can be manually transcribed. However, this task of manually transcribing the time alignment information is very labor intensive and very subjective. Usually only the reference word transcription is supplied for training, and the detailed time alignments between the acoustic feature vectors and the sub-phonetic units are "implicitly" generated as part of the training process. In "lightly-supervised" training, the accuracy requirement for the reference word sequence is further relaxed [42]. In this thesis, we will only focus on the case where accurate reference word transcription is known for the training data.

Because the alignments between the acoustic feature vectors and the HMM states are also *unobservable*, the EM training of HMMs is a more difficult problem than the GMM training problem. Let a random variable q_t correspond to the HMM state that observation vector $\vec{x}_t$ is aligned with. The update equations for EM training of HMMs parameters are:

$$c_{j,k}^{new} = \frac{\sum_{t=1}^{N} P(q_t = j, m_{q_t,t} = k|X, \Theta)}{\sum_{t=1}^{N} \sum_{l=1}^{M} P(q_t = j, m_{q_t,t} = l|X, \Theta)}, \tag{2.21}$$

$$\vec{\mu}_{j,k}^{new} = \frac{\sum_{t=1}^{N} P(q_t = j, m_{q_t,t} = k|X, \Theta)\ \vec{x}_t}{\sum_{t=1}^{N} P(q_t = j, m_{q_t,t} = k|X, \Theta)}, \tag{2.22}$$

$$\Sigma_{j,k}^{new} = \frac{\sum_{t=1}^{N} P(q_t = j, m_{q_t,t} = k|X, \Theta)\ (\vec{x}_n - \vec{\mu}_{j,k}^{new})\ (\vec{x}_t - \vec{\mu}_{j,k}^{new})^T}{\sum_{t=1}^{N} P(q_t = j, m_{q_t,t} = k|X, \Theta)}. \tag{2.23}$$

Notice that this set of equations is very similar to EM-based GMM training update Equations 2.18-2.20, except for the term $P(q_t = j, m_{q_t,t} = k|X, \Theta)$. This term eval-

uates the posterior probabilities of the observation feature vector $\vec{x}_t$ aligning to the k^{th} Gaussians component of the j^{th} HMM state observation models. This term can also be decomposed into,

$$P(q_t = j, m_{q_t,t} = k|X, \Theta) = P(m_{q_t,t} = k|q_t = j, X, \Theta)\ P(q_t = j|X, \Theta). \qquad (2.24)$$

The second term of the product denotes the posterior probability of the observation feature vector $\vec{x}_t$ aligning to the j^{th} HMM state observation model. The first term of the product denotes the posterior probability of observation feature vector $\vec{x}_t$ aligning to the k^{th} Gaussians component. This can be viewed as each observation feature vector is time aligned with a *set* of HMM states. This alignment is weighted by the posterior probabilities, $P(q_t = j|X, \Theta)$.

The brute force method of calculating the terms in Equation 2.24 is exponential with the length of the training data. Taking advantage of the independence assumptions of HMMs, the Baum-Welch algorithm [1] can reduce the computation complexity to being linear with the length of the training data.

2.1.3.4 Viterbi Training

The parameters of HMMs can also be trained with a procedure call "Viterbi training" [36]. For every iteration of Viterbi training, each observation feature vector $\vec{x}_t$ is aligned to a *single* HMM state instead of a *set* of HMM states as in the EM training. Using a procedure called "Viterbi alignment" [56], the single-best HMM state alignment sequence can be evaluated for each sequence of observation feature vectors. The parameters of the HMMs are similarly updated using Equations 2.21-2.23 except that the term $P(q_t = j|X, \Theta)$ is *approximated* with an indicator function. This indicator function evaluates to 1 when j is equal to the state index of the Viterbi-aligned state for $\vec{x}_t$, and it evaluates to 0 otherwise. In comparison to the EM training algorithm for HMMs, the Viterbi training typically converges faster, and is also computationally less expensive. However, the performance of the Viterbi training algorithm is sensitive to the choice of initialization models. In contrast, the EM training algorithm is less

sensitive to the initialization model and can be trained with a flat initialization model. For frame-based speech recognizers, the EM-based training algorithm [14, 57] has been shown to have a smoother convergence property than the Viterbi training [57].

2.2 Segment-based Automatic Speech Recognition

In the previous section, we reviewed the standard probabilistic formulation for the frame-based approach using HMMs, as well as the acoustic and language models and the associated training and decoding algorithms. In this section, we will highlight aspects of the segment-based system that is different from the frame-based approach.

2.2.1 Features

Unlike frame-based features, the features in segment-based ASR systems are computed on time intervals that are not necessarily equal. Two different types of feature vectors for the segment-based approach have been developed, namely *segment features* and *landmark features* [27]. The segment features are computed from the portion of the speech waveform belonging to a hypothesized phonetic segment, and the landmark features are computed from fixed-size waveform intervals centered at landmarks. The landmark feature framework is motivated by the belief that acoustic cues important for phonetic classification are located at acoustic landmarks corresponding to oral closure (or release) or other points of maximal constriction (or opening) in the vocal tract [65]. The segment feature framework promotes flexible modeling of phonetic segments without the conditional independence assumption imposed by HMMs. Figure 2-3 illustrates examples of frame-based, landmark, and segment features. The segment and landmark features can be used individually or jointly for modeling the dynamics of the acoustic observations.

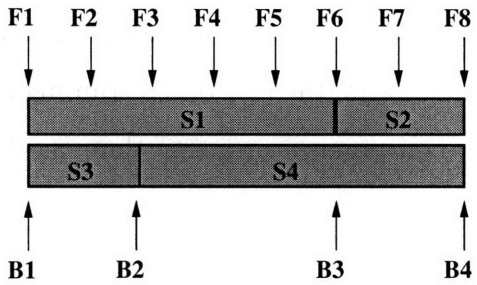


Figure 2-3: Examples of frame-based, landmark, and segment features. The feature vectors, $F1$, $F2$, ..., $F8$, are the framed-based which are sampled at fixed-size intervals. The landmark feature vectors, $B1$, $B2$, $B3$, and $B4$ are sampled at variable size intervals. The segment feature vectors, $S1$, $S2$, $S3$, and $S4$, each spanning two landmark features.

2.2.2 Segmentation Network

A segment-based ASR system either implicitly or explicitly hypothesizes segmentations of the speech waveform, although SUMMIT typically uses explicit segmentation, especially for real-time performance. It is worth noting that the segmentation is not simply a single sequence of non-overlapping segments; rather it is a network of segment sequences, which allows multiple segmentation sequences to be encoded together. Figure 2-4 shows an example of a segmentation network used by SUMMIT recognizer. The segmentation network can be computed directly from acoustics [59] or using "segmentation by recognition" where a phone graph is produced by a reduced set of acoustic phone models. Sainath and Hazen have recently developed a new algorithm for computing segmentation network based on sinusoidal model of speech that are more robust to noise. [60] The use of *network* of segmentation paths reduces the accuracy requirement of the algorithm for hypothesizing the segmentation paths, thus increasing the robustness of the overall segment-based system. Frame-based HMM ASR systems do not generate a segmentation network. The frame-based approach can be viewed as using an implicit fully-connected segmentation network.

It is clear that when the segmentation network can be computed "correctly," the segmentation network can reduce computation and improve the word error rate (WER). However, a "correct" segmentation network can be difficult to obtain. This thesis will attempt to answer the question of whether the use of a segmentation

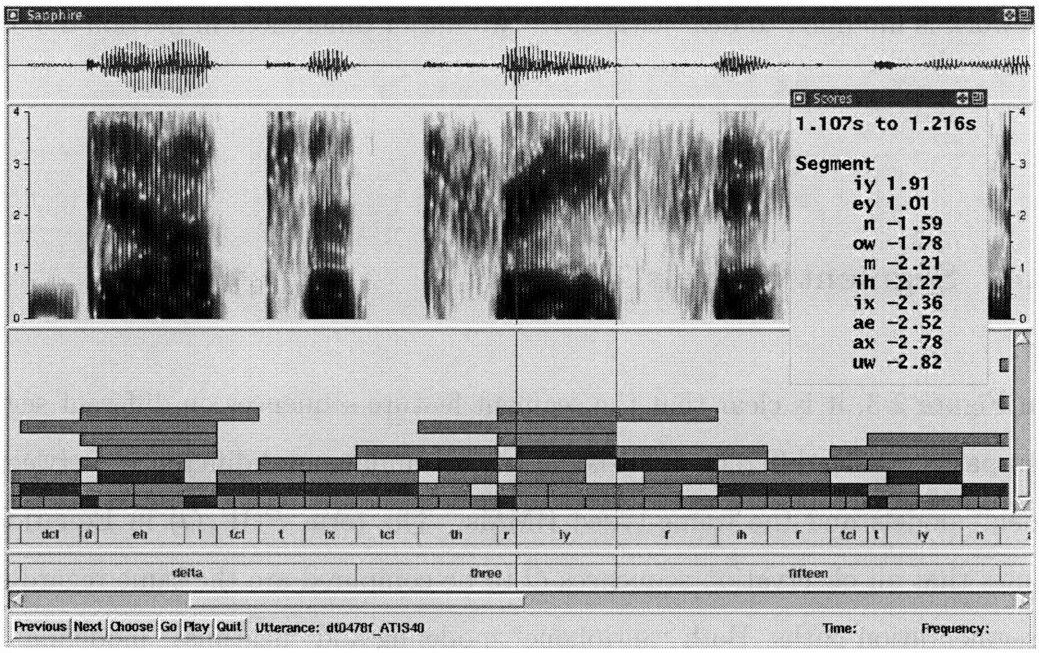


Figure 2-4: Graphical output from the SUMMIT segment-based ASR system. The top two panels display the speech waveform and corresponding spectrogram, respectively. The third panel shows the computed segmentation network consisting of hypothesized phonetic segments. The highlighted segments form a single segmentation path, which is also the segmentation path the decoder found to have the highest score. The fourth panel shows the hypothesized phone sequence aligned to the highlighted segmentation path from the previous panel. The fifth panel shows the corresponding hypothesized word sequence.

network is beneficial.

2.2.3 Landmark Models

The segment-based landmark models are a generalization of the acoustic models for frame-based feature vectors. These two types of acoustic models differ in two aspects. First, the observation feature vector for landmark models is not limited to a fixed-frame-rate feature vector, but is rather sampled non-uniformly. Whether uniformly sampled or not, it is important to note that in both types of systems all the input feature vectors are the same on different segmentation paths. Second, the segmentation network in segment-based systems constrains the search space, whereas HMM-based system do not. The segmentation network constraint can be relaxed, so that it is exactly a fully connected network like HMMs. We will address how the probabilistic

framework is modified to deal with these two major differences in Section 2.3.3.

2.2.4 Segment Models

From Figure 2-3, it is clear that the segment feature sequences on different segmentation paths can be different. This is one of the fundamental differences between the segment models and the frame-based HMMs. The term $P(\vec{W}, \vec{O})$ in Equation 2.4 assumes that all observation sequences $\vec{O}$ to be compared are the same regardless of the segmentation paths. Both "antiphone" modeling and "nearmiss" modeling [7, 27] have been developed to address this. For brevity, only "antiphone" modeling is described here. "Nearmiss" modeling is described in detail in [27]. In "antiphone" modeling, instead of scoring only the segment features on the segmentation path, $\vec{O}_{on}$, the segments off the segmentation path, $\vec{O}_{off}$, are also scored. To simplify computation, the algorithm uses a single "antiphone" model, to score all the off-path segments. The conditional independence between on-path segment features and off-path segment features given a word sequence is also assumed. The term $P(\vec{W}, \vec{O})$ in Equation 2.4 becomes,

$$
\begin{aligned}
P(\vec{W}, \vec{O}) &= P(\vec{O}|\vec{W})P(\vec{W}) && (2.25)\\
&= P(\vec{O}_{on}\vec{O}_{off}|\vec{W})P(\vec{W}) && (2.26)\\
&= P(\vec{O}_{on}|\vec{W})P(\vec{O}_{off}|\hat{w})P(\vec{W}) && (2.27)\\
&= P(\vec{O}_{on}|\vec{W})P(\vec{O}_{off}|\hat{w})\frac{P(\vec{O}_{on}|\hat{w})}{P(\vec{O}_{on}|\hat{w})}P(\vec{W}) && (2.28)\\
&= K\,\frac{P(\vec{O}_{on}|\vec{W})}{P(\vec{O}_{on}|\hat{w})}P(\vec{W}), && (2.29)
\end{aligned}
$$

where $\hat{w}$ represents the non-lexical unit. For a phone-based system, the lexical units are phone-level units. In this case, the non-lexical units do not correspond to any phone-level units. $K = P(\vec{O}_{off}|\hat{w})P(\vec{O}_{on}|\hat{w})$, is constant for the same set of segment

observation vector, $\vec{O} = \vec{O}_{on} \cup \vec{O}_{off}$. Thus, Equation 2.4 becomes,

$$\vec{W}^* = \arg\max_{\vec{W}} \frac{P(\vec{O}_{on}|\vec{W})}{P(\vec{O}_{on}|\hat{w})} P(\vec{W}), \tag{2.30}$$

and the term $P(\vec{O}|\vec{S})$ in Equation 2.9 becomes $\frac{P(\vec{O}_{on}|\vec{S})}{P(\vec{O}_{on}|\hat{s})}$, where $\hat{s}$ represents "non-segment" units. Note that with "antiphone" modeling, the calculation is limited to the observation on the segmentation path, $\vec{O}_{on}$. The computation of "antiphone" is only slightly more complicated.

2.2.5 Viterbi Training of Segment-based Models

The baseline segment-based system uses Viterbi training for learning the parameters of the segment-based acoustic models [27]. The use of segment-based features and segmentation networks complicates the probabilistic modeling because they alter the sample space of all possible segmentation paths and the feature observation space. Viterbi training avoids these complications by only learning from the single best forced alignment with a given initial model. It is important to note that EM training was used for the segment-based recognition systems in [15, 53]; however these systems do not have the same difficulties from their feature vectors and segmentation network. In these studies the feature vectors are uniformly sampled, as in a typical frame-based recognition system. The segmentation networks are also similar to those of a frame-based system, an implicit fully-connected segmentation network. In this thesis, we will develop a common probabilistic formulation for both the segment-based and frame-base approaches by an innovative use of the finite-state transducer. The training of both approaches will be based the EM training algorithm, and the decoding algorithm will be based on the Viterbi algorithm [23]. Both the training and decoding will allow more sophisticated modeling, such as a model with more states. This will also enable us to consider a fusion of segment-based and frame-based processing methods.

2.3 Finite-State Transducers

Finite-state transducers (FSTs) have been shown to be useful in a number of speech and language processing applications [51]. FST operations such as composition, determinization, and minimization make manipulating FSTs very simple. The algorithms used for these operations can be "implemented once and used everywhere." An FST is an extension of a finite-state acceptor (FSA) where the arcs encode an input and output symbol pair. The individual paths specified by the FST represent mappings between the input and output label sequences. In general, the length of the input and output sequences can be different because labels can be empty.

When each arc is also associated with a weight or a score, it is commonly referred to as a *weighted* FST (wFST) in the literature. In this thesis, we will ignore this distinction, and interchangeably use FST and wFST to denote a weighted finite-state transducer. The interpretation of the weights on the arcs depends on how they are manipulated algebraically, and the algebraic structure is a semiring. This will be discussed in detail in Section 2.3.1.1.

Mohri, et al. have demonstrated a traditional HMM-based speech recognizer using FSTs [51]. Hetherington have successfully utilized FSTs to specify various constraints for the SUMMIT segment-based speech recognizer [33]. FSTs have also been successfully used for other speech and language processing applications, such as speech synthesis and natural language parsing and tagging [37].

In the remaining of the section, we will first formally define FSTs and semirings. In Section 2.3.2, we will discuss the probabilistic interpretation of FSTs, including the FST operations needed to convert a joint probability transducer to a conditional probability transducer. In Section 2.3.3, we illustrate how various constraints are represented by FSTs in a typical speech recognition systems.

2.3.1 Formal Definition of FSTs and the Semiring

2.3.1.1 Weight Semirings

A weight semiring[1] $K = (\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ defines the set $\mathbb{K}$ containing the weights, the operators $\oplus$ and $\otimes$, with the identity elements $\overline{0}$ and $\overline{1}$. The operators $\oplus$ and $\otimes$ are both communicative and associative, for all $a, b, c \in \mathbb{K}$, $a \oplus b = b \oplus a$, $a \otimes b = b \otimes a$, $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$, and $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$. The identity elements $\overline{0}$ and $\overline{1}$ have the following properties, for all $a \in \mathbb{K}$, $a \oplus \overline{0} = a$, $a \otimes \overline{1} = a$, $a \otimes \overline{0} = \overline{0}$ [41, 51].

Two semirings commonly used within speech and language applications include the real semiring $(\mathbb{R}, +, \times, 0, 1)$ and the tropical semiring $(\mathbb{R}_+ \cup \infty, \min, +, \infty, 0)$. The real semiring, abbreviated here $(+, \times)$, can be used to represent probabilities directly, where we take the product of probabilities in series and the sum of probabilities in parallel. The tropical semiring, abbreviated here $(\min, +)$, can be used to represent negative log (i.e., $-\log$) probabilities where we take the sum of $-\log$ probabilities in series and the minimum, or most probable, $-\log$ probability in parallel. The $(\min, +)$ tropical semiring corresponds to how scores are typically manipulated in a traditional Viterbi dynamic programming search.

2.3.1.2 Weighted Finite-State Transducer[2]

A weighted finite-state transducer (wFST) T over the semiring K is defined by a tuple $T = (\Sigma, \Omega, Q, E, i, F, \lambda, \rho)$ where Σ is the *input alphabet*, Ω is the *output alphabet*, Q is the finite set of *states*, E is the finite set of *transitions*, i is the *initial state* where $i \in Q$, F is the set of *final states* where $F \subset Q$, λ is the initial weight associated with the initial state i, and ρ is the *final weights function*, $\rho : f \in F \rightarrow \Re$.

A transition t is defined by a tuple, $t = (p[t], n[t], l_i[t], l_o[t], w[t])$. The transition t is an arc from the *source state* $p[t]$ to the *destination state* $n[t]$ with the *input label*

[1] Unlike the algebraic structure *ring*, a semiring may not contain the *additive inverse* in the set. For all $a \in \mathbb{K}$ and $a \oplus -a = \overline{0}$, $-a \in \mathbb{K}$ is not always true. A semiring can be thought as a ring without negative elements.

[2] The formulation is adapted from [51].

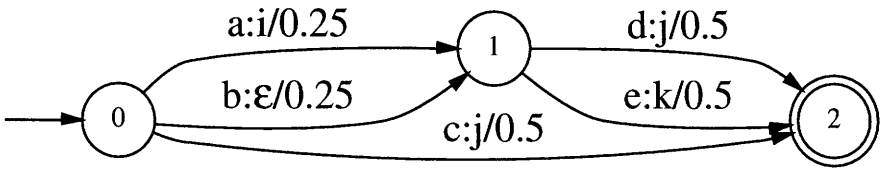


Figure 2-5: An example weighted finite-state transducer. The input alphabet is a, b, c, d, e. The output alphabet is i, j, k. There are three states, labelled 0, 1, 2. The initial state is 0, and the set of final state is 2. The arcs are each labelled with *input label : output label / weight.* There are a total of five possible paths represented by this FST. The path with the state sequence $0, 2$ has the highest weight which maps the input sequence c with the out sequence with the weight of 0.5.

of $l_i[t]$, *output label* of $l_o[t]$, and weight of $w[t]$.

A set of N consecutive transitions connecting the initial state and a final state forms a *permissible path* (or simply *path*), written $\pi = t_1 t_2 \cdots t_N$, with $p[t_1] = i$, $n[t_N] \in F$, and for all $j = 1, 2, \cdots N-1$, $n[t_j] = p[t_{j+1}]$.

The input label sequence associated with path π (or simply *input sequence*) is $l_i[\pi] = l_i[t_1] l_i[t_2] \cdots l_i[t_N]$. Similarly, the output label sequence associated with path π (or simply *output sequence*) is $l_o[\pi] = l_o[t_1] l_o[t_2] \cdots l_o[t_N]$. It is important to note that since an input or output symbol can be ϵ, representing the "empty" symbol, the input and output sequences associated with the same path can have different lengths. Figure 2-5 illustrates an example weighted finite-state transducer.

The semiring K specifies how the weights in the wFST can be manipulated. The $\otimes$ and $\oplus$ operators are used to combine weights in series and parallel, respectively. The *path weight* for the path π is $w[\pi] = \lambda \otimes w[t_1] \otimes w[t_2] \otimes \cdots \otimes w[t_N] \otimes \rho(n[t_N])$. The path weight for a set of paths is $w[\pi_1, \pi_2, \cdots, \pi_N] = w[\pi_1] \oplus w[\pi_2] \oplus \cdots \oplus w[\pi_N]$.

2.3.2 Probabilistic Interpretation of Weighted FSTs

The weighted FST specifies a weighted mapping between the input and output label sequences. The FST weights can have a probabilistic interpretation. With an appropriate choice of the semiring, the probabilistic interpretation of the FST weights can be maintained with FST operations, such as composition, ϵ-removal, etc.

2.3.2.1 Marginal, Joint, and Conditional Probabilities

Let X and Y be random variables representing the input and output label sequences of an FST, respectively. The weights in FST can represent the marginal probabilities $P(x)$ and $P(y)$, the joint probability $P(x, y)$, or the conditional probabilities $P(x|y)$ and $P(y|x)$.

Let $T_{X,Y}$ denote a joint probability FST. It is important to note that $\bigoplus_{x,y} T_{X,Y} = 1$. With the $(+, \times)$ semiring, it is sufficient to have all the weights leaving individual states sum to 1.

Let T_X and T_Y denote the marginal probability FSTs of the input and output label sequences respectively. The joint probability FST $T_{X,Y}$ can be marginalized with FST operators, namely a projection operation followed by the determinization operation:

$$T_Y = det(\text{project}_Y(T_{X,Y})) \ , \tag{2.31}$$

$\text{project}_Y(\cdot)$ is the projection operation which replaces all the *input* labels with their corresponding *output* labels on individual arcs. Similarly, $\text{project}_X(\cdot)$ replaces all the *output* labels with their corresponding *input* labels on individual arcs. The operator $\det(\cdot)$ determinizes the FST such that there is only one path for any distinct input label sequence. Since the input and output labels for T_X and T_Y are identical, they are really weighted FSAs. The determinization (and included ϵ removal) is important so that the marginal FST is properly specified. With $\det(\cdot)$, there is at most one path for any given y, with the determinization having performed the necessary $\oplus$ sum. It is important to note that it is not always possible to determinize a cyclic weighted FSA [51], and thus it is not always possible to compute a marginal FST from a joint FST. However, we have yet to encounter this situation in practice.

Let $T_{X|Y}$ denote the conditional probability FST. It is important to note that $\bigoplus_x T_{X|Y} = 1$ for all y or $\bigoplus_y T_{Y|X} = 1$ for all x. These conditions can be tedious to verify. It is typically easier to satisfy these conditions by the familiar Bayes' Rule

Probabilistic Equivalent	FST Operations
$P(y) = \sum_X P(x, y)$	$T_Y = det(\text{project}_Y(T_{X,Y}))$
$P(x\|y) = P(x, y)/P(y)$	$T_{X\|Y} = T_{X,Y} \circ [T_Y]^{-1}$
$P(x, y) = P(x\|y)P(y)$	$T_{X,Y} = T_{X\|Y} \circ T_Y$

Table 2.1: Manipulations of marginal, joint, and conditional FSTs and their probabilistic equivalent.

$P(x|y) = P(x, y)/P(y)$ with the FST equivalent:

$$T_{X|Y} = T_{X,Y} \circ [T_Y]^{-1} \ , \tag{2.32}$$

where $[\cdot]^{-1}$ replaces every non-$\overline{0}$ transition and final weight w by its reciprocal $\overline{1} \otimes^{-1} w$. For the $(+, \times)$ semiring this reciprocal is $1/w$, and for the $(\min, +)$ semiring it is $-w$.

Given a conditional probability FST $T_{X|Y}$ and marginal probability FST T_Y, we can compute the joint probability FST as:

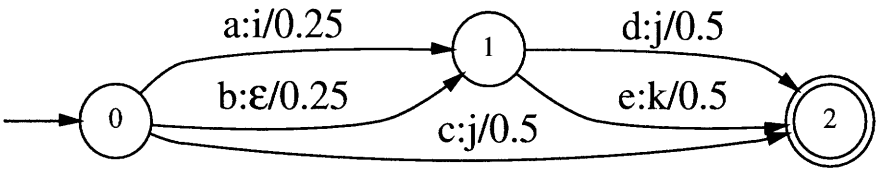
$$T_{X,Y} = T_{X|Y} \circ T_Y \tag{2.33}$$

as illustrated in Figure 2-6. Figure 2-6 shows various FSTs representing joint, conditional, and marginal probabilities. Note that the topology of the conditional FST $T_{X|Y}$ in (b) is different from the topology of the joint FST $T_{X,Y}$ in (a). In general the topology of a given joint distribution FST differs from the topology of its corresponding conditional distribution FST. Furthermore, some FST topologies are not able to support arbitrary conditional probability distributions due to ϵ outputs.

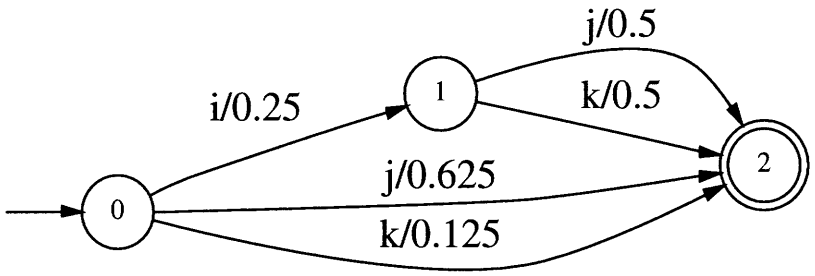
Table 2.1 summaries the FST operations used for manipulate among marginal, joint, and conditional FSTs and their probabilistic equivalent.

2.3.2.2 Cascade of FSTs

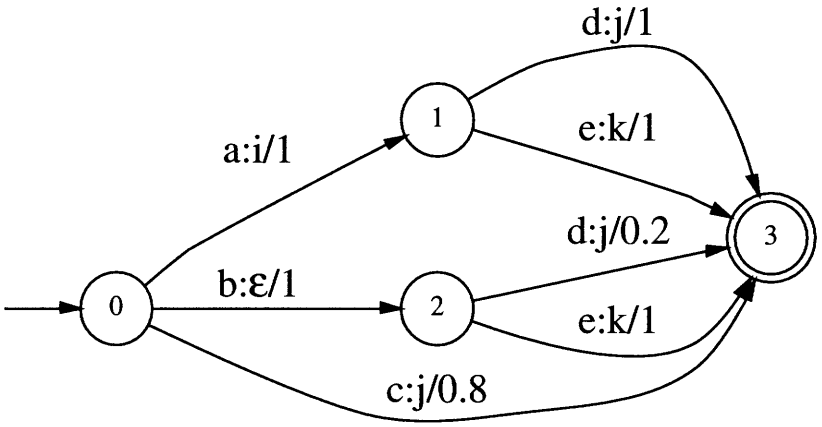
Consider a cascade of FSTs $W_{X,Z} = T_{X|Y} \circ U_{Y|Z} \circ V_Z$. Define $W(x, z)$ to be the $\oplus$ sum over the weights of all paths through $W_{X,Z}$ with input sequence x and output sequence z, and define $T(x|y)$, $U(y|z)$, and $V(z)$ to be analogous $\oplus$ sums for the FSTs $T_{X|Y}$ and $U_{Y|Z}$ and finite-state acceptor (FSA) V_Z, respectively. From the definition



(a) $T_{X,Y}$



(b) T_Y



(c) $T_{X|Y}$

Figure 2-6: Example FSTs in the $(+, \times)$ semiring: (a) $T_{X,Y}$ representing joint probability $P(x, y)$, notice that all the arcs leaving from state 0 and state 1 sum to 1 respectively. (b) T_Y representing marginal probability $P(y)$. T_Y is computed from the $T_{X,Y}$ using the equation $T_Y = det(\text{project}_Y(T_{X,Y}))$. Instead of 5 distinct paths in $T_{X,Y}$, there are only 4 distinct paths in T_Y because the state sequence 0, 1, 2 and the state sequence 0, 2 share the same output sequence of j. It is worthy to note that all the arcs leaving from state 0 and state 1 sum to 1 respectively. (c) $T_{X|Y}$ representing conditional probability $P(x|y)$. $T_{X|Y}$ is computed using $T_{X|Y} = T_{X,Y} \circ [T_Y]^{-1}$. Notice that probabilities of the two paths, the state sequence 0, 2, 3 and the state sequence 0, 3, sharing the same output sequence j sum to 1.

of weighted composition we have:

$$W(x,z) = \bigoplus_y T(x|y) \otimes U(y|z) \otimes V(z) \ . \tag{2.34}$$

If $W_{X,Z}$, $T_{X|Y}$, $U_{Y|Z}$, and V_Z represent probabilities in the real semiring $(+,\times)$, then $W(x|z) = P(x|z)$, $T(x|y) = P(x|y)$, $U(y|z) = P(y|z)$, and $V(z) = P(z)$. With the conditional independence assumption $P(x|y,z) = P(x|y)$, Equation 2.34 becomes the familiar chain rule:

$$P(x,z) = \sum_y P(x|y)P(y|z)P(z) \ .$$

For the tropical semiring $(\min,+)$ with $-\log$ probabilities, Equation 2.34 yields the following approximation:

$$\log P(x,z) \approx \max_y \left[\log P(x|y) + \log P(y|z) + \log P(z)\right] \ .$$

This is analogous to the approximation made by a traditional Viterbi dynamic programming decoder when it considers best paths rather than summing over all paths.

It is important to note that for a cascade of FSTs to chain together to represent a probability such as $P(w,z)$ above, the intermediate FSTs must represent conditional probabilities as do $T_{X|Y}$ and $U_{Y|Z}$ in this example.

2.3.3 FSTs for Automatic Speech Recognition

2.3.3.1 FST Cascade for Recognition

In an example segmental speech recognition system [27], constraints such as the acoustic model, model topology, context dependency, phonological rules [10, 25, 32], lexicon, and language model are all represented by weighted finite-state transducers (FSTs). Specifically, these constraints are represented by FSTs A, M, C, P, L, and G, respectively.

The acoustic model A associates acoustic feature vectors with sub-phone units, and the weights on A are *functions* of acoustic feature vectors and sub-phone units

(typically, these functions are Gaussian mixture models). For frame-based HMM systems, the FST A represents a linear sequence of frame-based feature vectors. Similarly for segment-based landmark features, they can also be represented as a linear sequence. For other features such as segment features where a directed acyclic graph is needed, they can also be represented by an FST. Thus, the acoustic model A can be used to represented both types of features: linear sequence features or directed acyclic graph features. The FST A may be *implicitly* represented in recognizer.

The model topology FST M currently used by SUMMIT is different from that of an HMM. In summary, the segment-based SUMMIT ASR system implemented with FSTs is a very flexible framework. It can be easily configured to implement an HMM by appropriately altering the FSTs A and M. M can be weighted, and the weights are typically called *state transition weights.* Figure 2-7 illustrates the different configurations of FST M using for landmark models, segment models, and HMM. The three configurations are illustrated in sub-figure a), b), and c) respectively.

The context dependency C is used to describe the phonetic context of phone units used for the acoustic model. Some example types of the context dependencies are context-independent phones, diphones, triphones, or clustered versions of these. In this thesis, we used the decision-tree based clustering similar to the one describe in [52]. C is typically unweighted.

The phonological rules P can be *optional* used in this framework. Whether the usage of phonological rules improves the recognition performance is still an open question. P is typically unweighted. In Chapter 3, we will describe an algorithm for training the weights for P from training data and demonstrated that a weighted P can improve the overall recognition performance.

The lexicon L is converted from the baseform dictionary. Since some words can have more than one pronunciation associated with them, L can also be weighted. In the most common systems, L is however typically unweighted. G is the language model as described in Section 2.1.1.

With these FSTs, the joint probability in Equation 2.8 has an FST equivalent,

$$\underbrace{P(\vec{S},\vec{U},\vec{W},\vec{O})}_{\downarrow} = \underbrace{P(\vec{O}|\vec{S})}_{\downarrow} \cdot \underbrace{P(\vec{S}|\vec{U})}_{\downarrow} \cdot \underbrace{P(\vec{U}|\vec{W})}_{\downarrow} \cdot \underbrace{P(\vec{W})}_{\downarrow} \tag{2.35}$$

$$A \circ M \circ C \circ P \circ L \circ G = A \circ M \circ (C \circ P \circ L) \circ G$$

The recognition problem of Equation 2.6 is thus converted to the equivalent problem of searching for the best path in $A \circ M \circ C \circ P \circ L \circ G$. $MCPLG = M \circ C \circ P \circ L \circ G$ is independent of the input utterance to be recognized, so it is typically cached for computational reasons. $A \circ (MCPLG)$ composition is really performed implicitly during the decoding search.

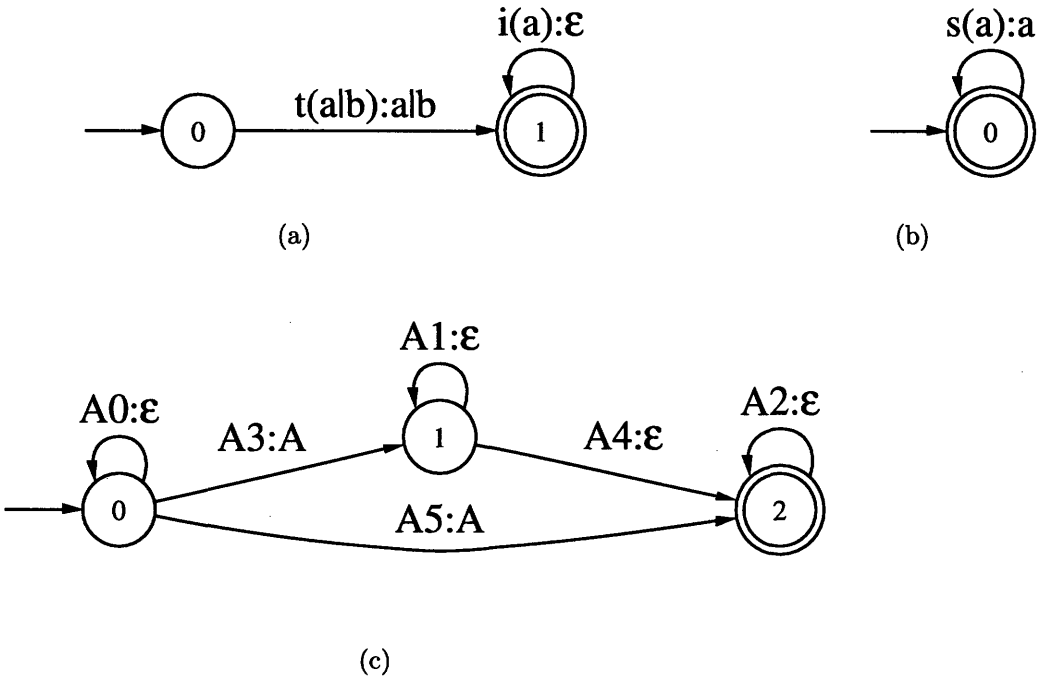


Figure 2-7: Illustration of the model topology FSTs M. (a) is used by the current SUMMIT landmark features, (b) is used by the current SUMMIT segment features, and (c) is for a 3-state HMM with skip transitions.

2.4 Summary

In this chapter, we have given a brief introduction to automatic speech recognition for both frame-based and segment-based approaches. We have highlighted the differences between two approaches and have explained the challenges to construct an unified framework for both approaches. We also formally introduced the weighted finite-state transducer, formulated a probabilistic interpretation of finite-state transducer weights, and showed how probabilities on the FST arcs can be manipulated to maintain the probabilistic interpretation. Finally, we constructed the typical probabilistic formulation for automatic speech recognition using the weighted FSTs.

In the coming chapters, we will develop the unified framework for both frame-based and segment-based systems with an innovative use of the finite-state transducer, and then we will extend this unified framework for the multi-stream recognition using multi-tape finite-state transducers.

Chapter 3

EM Training of FST Weights

In the previous chapter, we have shown how FSTs are used for specifying a weighted mapping between input and output sequences and how the weights can be manipulated as probabilities. In practice, the appropriate weights for the FST are often unknown and need to be assigned. When the number of arcs in the FST is small, experts with significant domain knowledge are able to assign the weights manually. In most cases, manually assigning FST weights can require a significant amount of effort. Often, a set of training data that relates to the FST's input and output symbols is available. In this chapter, we extend the well-known Expectation Maximization (EM) algorithm [2, 14] to the problem of learning FST weights from a set of training data. In this chapter, we present a novel FST weight training algorithm for generic FSTs. Eisner concurrently developed a similar FST weight training algorithm [20, 21]. This new algorithm expand the existing repertoire of FST operations such as composition and N-best search.

First in Section 3.1, we present the novel method that learns weight for arbitrary FSTs using the EM algorithm. In Section 3.2, we apply the proposed algorithm to the problem of learning pronunciation weights for a speech recognizer. This work was previously presented in [30, 64].

3.1 EM Weight Training

We pose the FST weight training problem as an optimization problem, maximizing the likelihood of the training data by optimally assigning the FST weights. A similar algorithm for training weights for FSTs was developed by Eisner concurrently [20, 21]. Eisner's algorithm makes use of a novel "expectation semiring". For the "isolated training" and "training with cascade" problems presented in the following sections 3.1.1 and 3.1.2, Eisner's algorithm is equivalent with the algorithms presented in this chapter except for some additional memory and computation required for the "expectation semiring". With the additional memory and computation for the "expectation semiring", Eisner's algorithm is also able to learn FST weights of FSTs X and Y *jointly* from training pairs corresponding to the input and output of $X \circ Y$. Our algorithm however does not handle this case. It may be possible to extend our algorithm to handle this case in the future.

In Section 2.3.2, we discussed how some FST topologies are not able to support arbitrary conditional probability distributions due to ϵ outputs. For this reason, we chose to train a joint distribution FST using EM and afterwards convert it to a conditional distribution FST using Equation 2.32 if required.

In the following two subsections, we will describe the EM training algorithm for FSTs in detail. First, we will describe the more straightforward version of the algorithm where the two random variables associated with the training FST are directly observable, i.e. the training data is a set of input and output sequence pairs of the training FST. Second, we will extend the algorithm to the case where the input and output sequences are not directly observable.

3.1.1 Isolated Training

In this simpler version of the EM training of FST weights algorithm, a set of joint input/output sequence pairs (x_i, y_i) is given as training data for the joint probability FST $T_{X,Y}$. The goal is to assign the weights for $T_{X,Y}$ so to maximize the joint likelihood of the training data set (x_i, y_i). Figure 3-1 illustrates the setup of this

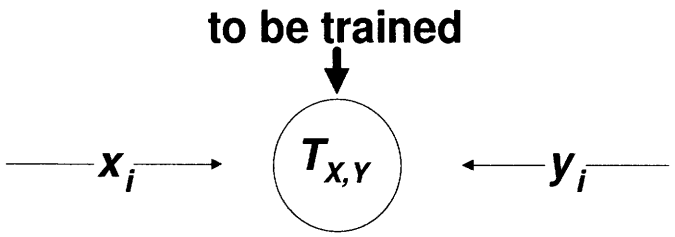


Figure 3-1: Illustration of FST $T_{X,Y}$ with paired input/output training sequence pair (x_i, y_i).

training problem.

We use the EM algorithm [14] to train a joint probability model for an FST $T_{X,Y}$. For a given joint input/output sequence pair (x_i, y_i), multiple paths through $T_{X,Y}$ may be permitted. We initialize the weights of $T_{X,Y}$ such that for each state, all leaving transitions are equally likely. For the initialization step, the final weights of the final states are also treated as leaving transitions from the states. These final weights are also initialized before the training. We use the $(+, \times)$ semiring for the weights to represent probabilities directly during the EM training, and if desired convert trained weights to the $(\min, +)$ semiring to represent log probabilities after training. Finally, if we require a conditional probability model, we convert the joint FST to its corresponding conditional FST using the method of Section 2.3.2.1.

During the expectation step of each EM iteration, for each input/output sequence pair (x_i, y_i) in the training corpus, we compute the expected number of times each transition in $T_{X,Y}$ is traversed as follows:

1. Compute $T_i = x_i \circ T_{X,Y} \circ y_i$, essentially the part of $T_{X,Y}$ supporting input sequence x and output sequence y. T_i may contain more than one path.

2. Normalize the weights in T_i such that probabilities of all paths sum to 1.

3. Update the expected transition counts for $T_{X,Y}$ that correspond to transitions in T_i.

For the maximization step, we convert the transition and state final counts to a joint probability distribution by normalizing counts so that the total weights of all

transitions (and state finality) leaving each state is 1. To allow the trained joint distribution to generalize to unseen input/output sequences that it accepts, we typically apply a floor to all counts so that transitions are not assigned zero probabilities.

Note that by construction, the probabilities of all the paths in $T_{X,Y}$ sum to 1 both before the expectation step and after the maximization step. Both the initialized $T_{X,Y}$ before the expectation step and the resulting $T_{X,Y}$ after each maximization step are joint probablities of symbol sequences in X and symbol sequences in Y. In step 1 of the expectation step, we compute T_i which contains only the set of paths in $T_{X,Y}$ that are compatiable with both the input sequence x_i and the output sequence y_i. The composition of T_i with x_i on the left and y_i on the right computes this set of paths. The weights of the T_i arcs are the same as the weights on the corresponding arcs in $T_{X,Y}$. The FST T_i is not processed by any other FST operations such as optimization or determinization so that the correspondence between the arcs of T_i and $T_{X,Y}$ can be maintained easily. This correspondence is needed in step 3. The training pair x_i and y_i are here assumed to be a *single* sequence of symbols. In general, both x_i and y_i can be finite state networks since finite state networks can be readily represented by FSTs.

Since the FST T_i contains only a subset of the paths in $T_{X,Y}$, all paths in T_i typically sums to less than 1. The goal in step 2 is to convert T_i into an FST representing the conditional probabilities of input sequences in X_i given output sequences in Y_i. This is accomplished by normalizing the weights in T_i such that the path probabilities do sum to 1. Specifically, the weights of arcs with the same starting state are normalized by the sum of these arc weights.

In step 3, the accumulators, one per each arcs in $T_{X,Y}$, are updated with the corresponding normalized arc weight in T_i. As stated earlier, the correspondence between the arcs in T_i and $T_{X,Y}$ are maintained during the step 1 and 2 for the easy of step 3. Note that all three steps of the expectation stage are performed one-by-one on each training pair.

Both the expectation steps and the maximization step are performed iteratively until the weights in $T_{X,Y}$ converges. After each iteration, the total likelihood of the

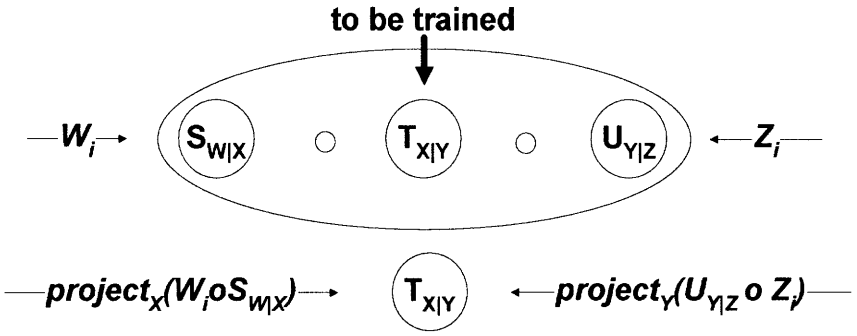


Figure 3-2: Training FST $T_{X|Y}$ within the FST cascade $S_{W|X} \circ T_{X|Y} \circ U_{Y|Z}$ with paired input/output training sequence pair (w_i, z_i), where the weights for FSTs $S_{W|X}$ and $U_{Y|Z}$ are known. This training within an FST cascade is equivalent to the isolated training problem with appropriate operations on the training pair.

training data computed with the resulting $T_{X,Y}$ is guarenteed to improve until it reachs a local maxima.

3.1.2 Training Within Cascade

We have outlined how to train an isolated joint FST from example pairs of direct input and output sequences. It is also possible to train a FST in the middle of a cascade such as $S_{W|X} \circ T_{X|Y} \circ U_{Y|Z}$. Figure 3-2 illustrates the setup of this training problem. In this case, we may wish to train $T_{X|Y}$ from example sequence pairs (w_i, z_i). A straightforward way to accomplish this is to compute the weighted FSAs $x_i = \text{project}_X(w_i \circ S_{W|X})$ and $y_i = \text{project}_Y(U_{Y|Z} \circ z_i)$. These FSAs represent all possible input and output sequences (x, y) for $T_{X|Y}$ compatible with the given (w_i, z_i). Then, x_i and y_i FSAs can be used as in Section 3.1.1.

3.2 Experiments and Results

To test the FST EM training algorithm, we apply the algorithm to the problem of learning pronunciation weights. With phonological rules and multiple phonemic pronunciations, the pronunciation graph for spontaneous speech can have high branching factors. Pronunciation weighting has been shown to be beneficial for segment-based speech recognition [67]. In [67] within-word pronunciation weights were ML estimated

from training examples. Here we experiment with learning various pronunciation weights on phonological rules and phonemic pronunciations via the proposed FST EM algorithm.

3.2.1 Task

For the recognition task, we chose to use a weather information domain for the experiment data [73]. The training set consisted of 116K utterances, totaling 105 hours of speech. The acoustic model employed 1,573 clustered diphone models using mixtures of diagonal Gaussians, with a total of 35K component Gaussians. The test set contained 1,711 utterances with 9,659 words, totaling 1.6 hours of speech. The decoding dictionary consisted of 2,014 words, covering the test set. Bigram and trigram class-based language models were used in the first pass and the second pass respectively. Both models were trained with 236 hand-crafted class definitions and transcriptions of the training set and 2,661 utterances containing out of training vocabulary words.

In the baseline recognition configuration, the context dependency FST C, the phonological rules FST P, and the lexicon FST L were unweighted FSTs, all weights are 0 in the $(\min, +)$ semiring (i.e., $\overline{1}$). The phonological rules FST P specifies a many-to-many mapping between phone sequences and phoneme sequences. Due to multiple alternative pronunciations for the same word, the lexicon FST L also specifies a many-to-many mapping between phoneme sequences and words. The language model G was a weighted FSA, and the weights were imported from a separately trained n-gram. The baseline recognition word error rate (WER) was 9.4%. There is no need to assign weights to C because it represents a one-to-one mapping from phone sequences to their corresponding sequence of context-dependent labels. We suspected that training the weights of P and L would decrease the word error rate. The FST EM algorithm detailed in Section 3.1 enabled learning weights for P and L without implementing custom training methods.

To learn the weights, we first computed the "reference phone labels" on the training set with baseline acoustic models and the baseline U with unweighted P and L. For simplicity the one-best "reference phone labels" were used in place of a phone lat-

U	WER	Rel. Red.
$C \circ P \circ L \circ G$	9.4%	-
$C \circ tr(P) \circ L \circ G$	9.0%	4.3%
$C \circ P \circ tr(L) \circ G$	8.8%	6.4%
$C \circ tr(P) \circ tr(L) \circ G$	8.7%	7.4%
$C \circ tr(P \circ L) \circ G$	8.7%	7.4%

Table 3.1: Recognition results and relative reduction (Rel. Red.) in WER for various pronunciation weight training configurations. The operator $tr()$ denotes the weights of the FST were trained with the FST EM training algorithm. The size of the first four FST cascades are the same, and the size of the last one is different since P and L are composed together first.

tice. Together, the "reference phone labels" and the reference word transcription on the training set formed the example sequence pair (x_i, z_i) in Section 3.1.2 needed for EM training. For each experiment, we first trained a joint FST, then we computed the corresponding conditional FST using Equation 2.32. Because decoding in the speech recognizer was done via the $(\min, +)$ semiring, we also evaluated Equation 2.32 with the same semiring, so that the resulting conditional FSTs would contain at least one input sequence with a path weight of 0 (i.e., $\overline{1}$) for any possible output sequence. The usage of the $(\min, +)$ semiring ensures that regardless how many input sequences correspond to the same output sequence, the input sequence with the highest probability always has a path weight of 0. With this property, the output sequences with many alternative input sequences are not penalized unnecessarily. We also tried to evaluate Equation 2.32 via the $(+, \times)$ semiring which does penalize output sequences with many alternative input sequences, and the WER increased.

The results of training various pronunciation weights were summarized in Table 3.1 and described in more detail in the following subsections. For the purpose of fair comparison, we used the same beam pruning parameters for all the conditions.

3.2.2 Training Phonological Rules P

The baseline recognizer uses 168 hand-crafted phonological rules that map 63 "phonemic" input symbols to 71 "phonetic" output symbols [32]. Each rule is in the form

of:

$$\{\lambda_1, \lambda_2, \ldots, \lambda_m\}\ \phi\ \{\rho_1, \rho_2, \ldots, \rho_n\} \Rightarrow \psi.$$

It states that phoneme ϕ with left input context of $\lambda_1, \lambda_2, \ldots, \lambda_m$ and right input context $\rho_1, \rho_2, \ldots, \rho_m$ can be mapped to a regular expression of phones, ψ. Additionally, ψ has the capability to specify output (surface) context constraints. "<{}" and ">{}" are used for left and right output context constraints respectively. The rules apply to both within-word and cross-word phoneme sequences. All the pronunciation rules can be found in Appendix B. For example, one rule for flappable /t/ is expressed by[1]:

$$\{\text{VOWEL}\}\ t\ \{\text{VOWEL}\} \Rightarrow dx \mid tcl\ t.$$

This rule only applies to intervocalic /t/'s. In this case, /t/ can be mapped to a flap [dx] or [t] closure, [tcl], followed by a [t] release. The weights associated with this rule would model how often an intervocalic /t/ is flapped.

P has about 800 states and 12,000 transitions. To speed up training, we decomposed P into a cascade of three FSTs, $P = S \circ R \circ I$, where both S and I represented deterministic mappings between input and output sequences. Thus, learning weights on R alone is equivalent to learning weights on P in whole. R contained only 249 states and 884 transitions. After EM training R, and building a new U with $C \circ S \circ tr(R) \circ I \circ L \circ G$, where $tr(R)$ denotes the conditional probability FST R after EM training with data. This new recognizer with trained P, $tr(P)$, obtained a word error rate rate of 9.0%, a relative reduction of 4.3% from the baseline.

3.2.3 Training Phonemic Pronunciations L

L represents the phonemic pronunciations of words. L has 5,542 states and 8,312 transitions. The training procedure learned relative frequencies of the different pronunciations which were used by the training data. The phonemic pronunciations in

[1]All the symbols used for phonemes and phones are based on the ARPAbet, described in Appendix B. Here we use the convention that phonemes are enclosed in "/ /" and phones are enclosed in "[ ]".

training L were not shared between similar words, e.g., the paths for the word "rain" and the word "raining" are not shared. Thus, this learning process only trained word-dependent phonemic pronunciation weights. The new recognizer with $tr(L)$ achieved a WER of 8.8%, a relative reduction of 6.4% from the baseline.

3.2.4 Training P and L Separately

In the two previous subsections, we trained weights for P and L separately. We can use both of them simultaneously by constructing U with an FST cascade using both $tr(P)$ and $tr(L)$, $C \circ tr(P) \circ tr(L) \circ G$. The WER obtained using this new U was 8.7%, better than using either $tr(P)$ or $tr(L)$ alone, but only slightly.

3.2.5 Training $P \circ L$

Both P and L have relatively few branching points that need to be trained. To increase the number of parameters to be learned, we chose to train word-dependent pronunciation weights by composing P with L, (i.e. $P \circ L$). The resulting $P \circ L$ contains 14,428 states and 127,113 transitions, which was significantly bigger than the size of P or L. EM training to obtain the joint probability FST required only slightly more computation than training either P or L alone. The size of U with either $tr(P)$ or $tr(L)$ was similar to the baseline U. However, the U with $tr(P \circ L)$ had 50 times more transitions than the baseline U because the marginal distribution FSA increased in size dramatically. The marginal FSA which models word sequences learned a complicated model with long range dependencies. After projection of the joint probability FST, 27,467 transitions out of 127,113 of the resulting FSA were ϵ transitions. The determinization (including ϵ removal) of the marginal distribution FSA dramatically increased its size to nearly 6 million transitions. The resulting U size actually increased to 20 million transitions. Clearly, the application of Equation 2.32 to compute the exact conditional may be computationally impractical, and an approximation may be necessary for larger FSTs. Despite the increased number of parameters in $P \circ L$, the WER achieved was the same 8.7% achieved by $tr(P) \circ tr(L)$.

3.3 Summary

We have presented a novel method to train FSTs directly via the EM algorithm in this chapter. The method operates on any generic FST, even those with ϵ transitions. Because some FST topologies are not able to support arbitrary conditional probability distributions due to ϵ outputs, we chose to train a joint probability FST first, then compute the corresponding conditional probability FST from the trained joint FST.

We applied the EM training of FST weights for the pronunciation weighting problem in the weather information task. By learning pronunciation weights on P, L, and $P \circ L$ with the FST EM algorithm, we showed that WER can be reduced. To our knowledge, this is the first application of an FST training algorithm. In our experiments, weights on word-dependent phonemic pronunciations reduced WER more than weighting phonological rules. However, a trained pronunciation rules P has the advantage that it can provide pronunciation weights for unseen words. This property is desirable because it can provide some degree of vocabulary-independent pronunciation weighting. In the future, we plan to address this issue by training syllable-based pronunciation weights and also automatically learning pronunciation rules [63].

Since Equation 2.32 does not guarantee that the resulting conditional probability FST will be similar in size to the joint probability FST, there will be cases where the exact application of Equation 2.32 is impractical, e.g., $P \circ L$. To overcome this problem, different methods to approximate the marginal FST might be needed. Recall that the "reference phone labels" are computed using U. We plan to experiment with iteratively computing new "reference phone labels" based on the U with trained pronunciation weights. Training the pronunciation weights in this iterative way might reduce word error rate further.

To our knowledge, the other known FST EM training algorithm was concurrently developed by Eisner [20, 21]. The application of this algorithm for the problem of pronunciation weight training is the first successful use of this type of algorithm. The FST EM algorithm can have many applications other than pronunciation weight learning. It has also been used in an FST-based speech synthesis system [61]. To

facilitate wide use of this algorithm, we have included this as part of an open source FST toolkit [33].

Chapter 4

EM Training of Acoustic Models

4.1 Introduction

In Section 2.3.3, we described how the acoustic model can be represented by a conditional probability FST $A_{Os|Ss}$. It is important to note that unlike the FSTs we discussed in Chapter 3, the arc weights on $A_{Os|Ss}$ are not simply numbers, they are actually functions of both the input and output labels.

Acoustic models of speech recognition systems specify the likelihood of an observation feature vector sequence for a proposed sequence of sub-phone units. The parameters for the likelihood functions are typically learned from a set of training data with EM training or Viterbi training. Since there is no existing generic training operations for acoustic models represented by FSTs, the parameters for the acoustic models are typically learned outside the FST framework then imported in. In this chapter, we extend the EM FST weight learning algorithm for the FSTs where the arc weights are functions. This generalization enables the training of the acoustic model represented by FSTs directly. This new algorithm is an extension of the novel FST weight learning algorithm for FSTs where the arc weights are simply numbers presented in Chapter 3.

4.2 EM Weight Training of Acoustic Models

The EM training of acoustic models consists of two steps. First, the "expectation" step (or E step) computes the posterior probabilities, $\gamma_n(i)$ defined as:

$$\gamma_n(i) = P(q_n = i|\vec{O}, \lambda) \quad \forall i = 1, 2, \ldots, K, \tag{4.1}$$

where the random variable q_n is equal to integer i when the observation o_n belongs to the i^{th} acoustic model, $\vec{O}$ is a sequence of N observations, $\{o_1, o_2, \ldots, o_N\}$, λ is the parameter set for the current acoustic models, and K is the number of acoustic models. The posterior, $\gamma_n(i)$, is the probability that n^{th} observation belongs to the i^{th} acoustic model. Second, the "maximization" step (or M step) will train observation probability density functions (PDFs) with the posterior-weighted observations for each acoustic model. In the following sections, we will describe the details of these two steps.

4.2.1 Computation of the Posterior Probabilities

To compute the posterior probabilities, we can employ the standard equation using the forward probability, $\alpha_n(i)$, and backward probability, $\beta_n(i)$ [2, 34],

$$\gamma_n(i) = \frac{\alpha_n(i)\beta_n(i)}{\sum_{i=1}^{K} \alpha_n(i)\beta_n(i)}, \tag{4.2}$$

where $\alpha_n(i)$ and $\beta_n(i)$ are defined as,

$$\alpha_n(i) = P(o_1 o_2 \ldots o_n, q_n = i|\lambda), \tag{4.3}$$

$$\beta_n(i) = P(o_{n+1} o_{n+2} \ldots o_N|q_n = i, \lambda). \tag{4.4}$$

Let W be the linear FST representing the sequence of reference words, $\vec{W}$. Given a sequence of observations, o_i, and its corresponding reference word sequence, W, one can construct an FST, Z, that specifies all possible mappings between each observa-

tion, o_i, and each state variable q_n. We refer to the FST Z as the *training lattice.* Similar to the FST operations used in Section 3.1.2 for training FST weights within an FST cascade, the training lattice FST Z can be computed by,

$$Z = o_i \circ A \circ project_I(M \circ C \circ P \circ L \circ W). \tag{4.5}$$

The term $project_I(M \circ C \circ P \circ L \circ W)$ on the right-hand side is an acceptor for the (possibly infinite) sequences of sub-phone units implied by the word sequence, $\vec{W}$. As described in Section 2.3.3, FSTs M, C, P, and L represent various constraints used by the recognizer. The training lattice Z is computed for each training utterance. By construction, all input label sequences of the training lattice are the same as the linear sequences o_i. The forward and backward variables $\alpha_n(i)$ and $\beta_n(i)$ can be computed on the network specified by Z. Finally, $\gamma_n(i)$ can be computed from $\alpha_n(i)$ and $\beta_n(i)$ according to Equation 4.2.

1 Compute training lattice Z using Equation 4.5
2 Compute the forward variable using Equation 4.3
3 Compute the backward variable using Equation 4.4
4 Compute the posterior probabilities using Equation 4.2

Figure 4-1: Outline for computing the posterior probability for each training utterance.

4.2.2 Training Observation PDFs from Posterior-Weighted Feature Vectors

To train GMMs from posterior-weighted feature vectors, the above procedure needs to be modified slightly. Let $W = \{w_1, w_2, \ldots, w_N\}$ be the posterior probabilities associated with the data set X. Note that the w_n variables are the same as the $\gamma_n(i)$ from Section 4.2.1. Equation 2.17, used for the E step, needs to be modified to take into account the posteriors. Specifically, Equation 2.17 becomes,

$$P(k|\vec{x}_n, g_k) = \frac{c_k \; w_k \; p(\vec{x}_n|g_k)}{\sum_{l=1}^{M} c_l \; w_l \; p(\vec{x}_n|g_l)}. \tag{4.6}$$

The rest of the procedure and equations remain unchanged except that the algorithm goes over all the data instead of only data labeled by Viterbi search.

4.3 EM Training for Frame-based and Segment-based Acoustic Models

In the typical frame-based HMM system, there is no segmentation network which constrains the mapping between observation vectors and acoustic models. However, in a segment-based ASR system, the segmentation network does constrain the possible mappings between observations and acoustic models. The segmentation network constraint can be represented by an FST. We recall that the FST A in Equation 2.35 encodes the set of mappings between sequences of observation vectors and the sub-phone state sequences. We can incorporate the segmentation network constraint into Equation 2.35 by treating A as the composition of two FSTs, $A_S \circ A_M$, where the FST A_S represents the segmentation network constraint with the output symbol "#p" for marking phonetic landmarks, and the FST A_M simply translates the output symbol "M" into the set of all possible sub-phone states. The FST A with the segmentation network constraint has smaller branching factor than without the segmentation network constraint. Figure 4.3 shows a sample segmentation network, and its corresponding FST representations for landmark features and segment features, A_S.

The EM training of frame-based acoustic models simply follows the steps described in the previous section. For the EM training of segment-based acoustic models, the segmentation network constraint needs to be taken into account. Specifically, the FST A in Equation 4.5 needs to be replaced by $A_S \circ A_M$ for the training lattice computation. This is the key difference between EM training for frame-based models and for segment-based models.

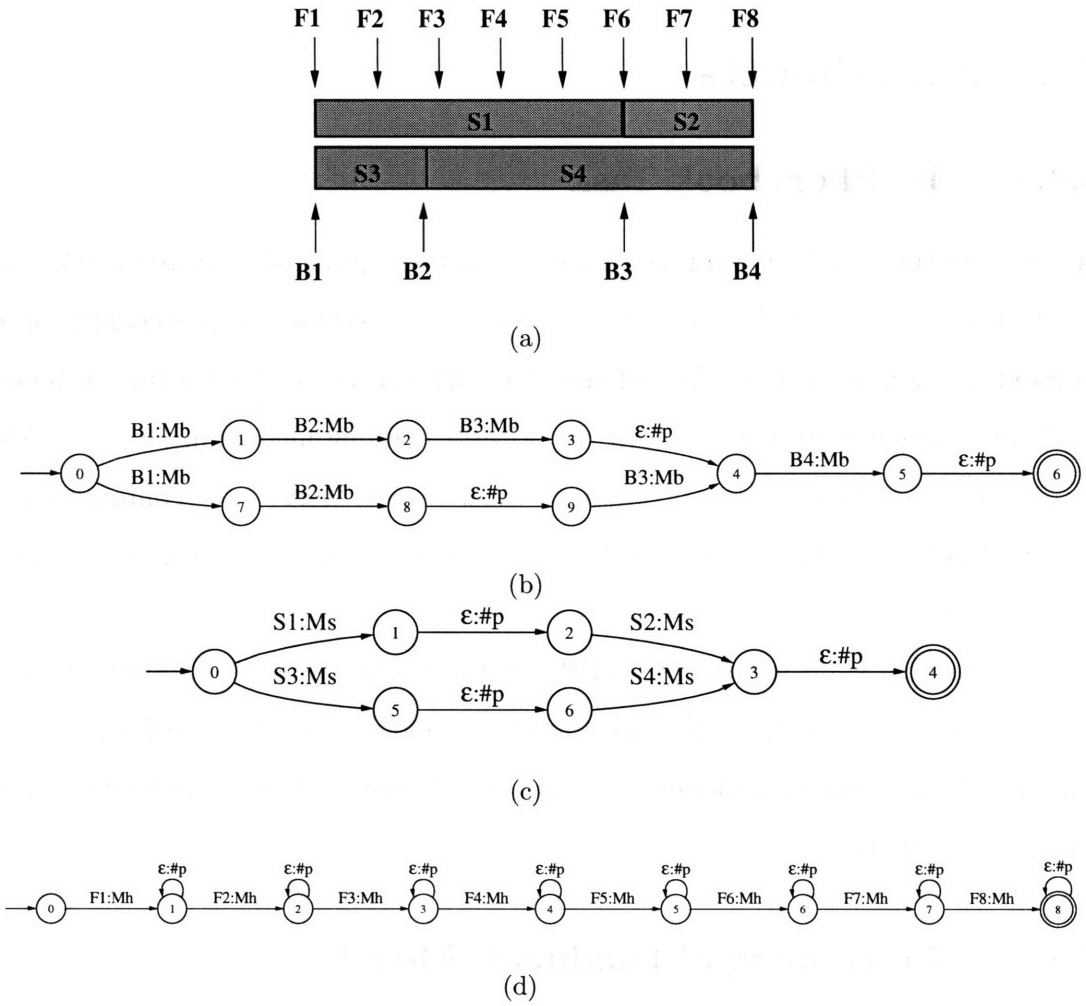


Figure 4-2: Illustration of a sample segmentation network and its corresponding FST representation. Here only the FST A_S is shown since FST A_M simply translates the output symbol Mb, Ms, Mh into the set of all possible sub-phone states. The segmentation network in (a) contains four phonetic segments with four landmark feature vectors, $B1$, $B2$, $B3$, and $B4$, and four segment feature vectors, $S1$, $S2$, $S3$, and $S4$. The feature vectors, $F1$, $F2$, ..., $F8$ are the corresponding fixed frame-rate feature vectors using by HMMs. (b) shows the corresponding FST A_S for landmark features with two identical input sequences, $B1B2B3B4$, and the symbol Mb represents the set of all landmark models. The symbol $\#p$ denotes phone landmark locations. (c) shows the corresponding FST A_S for segment features with two different input sequences each with two segments, $S1S2$ and $S3S4$, and the symbol Ms represents the set of all segment models. (d) shows the corresponding FST A_S for a frame-based HMM. Since the symbol $\#p$ in (d) does not provide any constraint, the size of the corresponding $A = A_S \circ A_M$ is typically bigger than that of segment-based models in (b) and (c). It is important to note the all the input sequences for landmark models as illustrated in (a) are the same, where the input sequences for segment models as illustrated in (b) can be different.

4.4 Experiments

4.4.1 The Phonebook Task

We applied the new EM training algorithm on the segment-based landmark models for the PhoneBook task [54]. The PhoneBook telephone-based corpus consists of read, isolated words from a vocabulary of close to 8,000 words. In the baseline systems the landmark models were trained with the Viterbi training algorithm [46]. As defined in [46], we focused on the more difficult task of the "large" set containing about 80,000 training utterances and 7,000 test sentences, with a decoding vocabulary of 8,000 words.

The baseline word error rate (WER) on the training is 4.3%, and on the test is 9.9%. This baseline is with landmark acoustic models only. We are focused on EM training of the landmark models here, so we will only compare with the results of landmark models.

4.4.2 EM Training of Landmark Models

Training Method	# Params	Training WER	Test WER
Viterbi	1.55M	4.3%	9.9%
EM	1.64M	2.7%	9.4%

Table 4.1: Word error rates (WER) of segment-based recognizer training using Viterbi training and EM training on the training set and test set.

Table 4.1 summarizes the results of WERs of the baseline systems and of EM trained models. The EM trained acoustic models achieved a relative error reductions of 37% on training, and a relative error reductions of 5% on test. The WER improved significantly on training, but on test the improvement was much smaller.

Although the WER improvement on the test set is small, EM training has a desirable advantage over Viterbi training. Viterbi training requires an initial set of acoustic models for forced alignment of the training data, whereas EM training is bootstrapped with flat initialization models—mixtures with single zero-mean unit-variance Gaussian components. The performance of Viterbi trained acoustic models

is thus dependent on the quality of the initial models. Since the initial models are typically learned from additional data, the implicit training set is arguably bigger than the stated training set. More importantly, in some cases the initialization required by Viterbi training is difficult to obtain. For example, when Tang et al. experimented with a two stage recognition system in which the first stage is a recognizer using a reduced phone set [68], the requirement of good initialization models limited the types of reduced phone sets to be a many-to-one mapping of an existing recognizer's phone set. Because EM training does not require any pre-trained initial acoustic model, the set of reduced phone set are not limited. However, EM training is slower since it has to iterate through the training data a number of times. On the PhoneBook task, EM training is about ten times slower than the Viterbi training baseline.

4.5 Summary

In this chapter, we have extended the EM training algorithm for FST weights to EM training for acoustic models that can be represented by FSTs. Since we can represent both frame-based and segment-based acoustic models as FSTs, this training algorithm completes the common framework for both frame-based and segment-based speech recognition systems. With this common framework, one can use the same generalized algorithms for training and decoding of frame-based or segment-based speech recognizers.

This common framework enables a direct comparison of the frame-based and the segment-based approaches. We have preliminarily explored the effect of the segmentation network on the overall systems performance. For example, with a less constrained segmentation network, and Viterbi trained duration models, we achieve a PhoneBook test WER of 7.6%, which we believe is the lowest reported result on this task. This result also suggests that the "standard" SUMMIT acoustic segmentation algorithm for generating the segmentation network is too restrictive. While the resulting segmentation network improves the decoding time, it is doing so at the expense of recognition error rate.

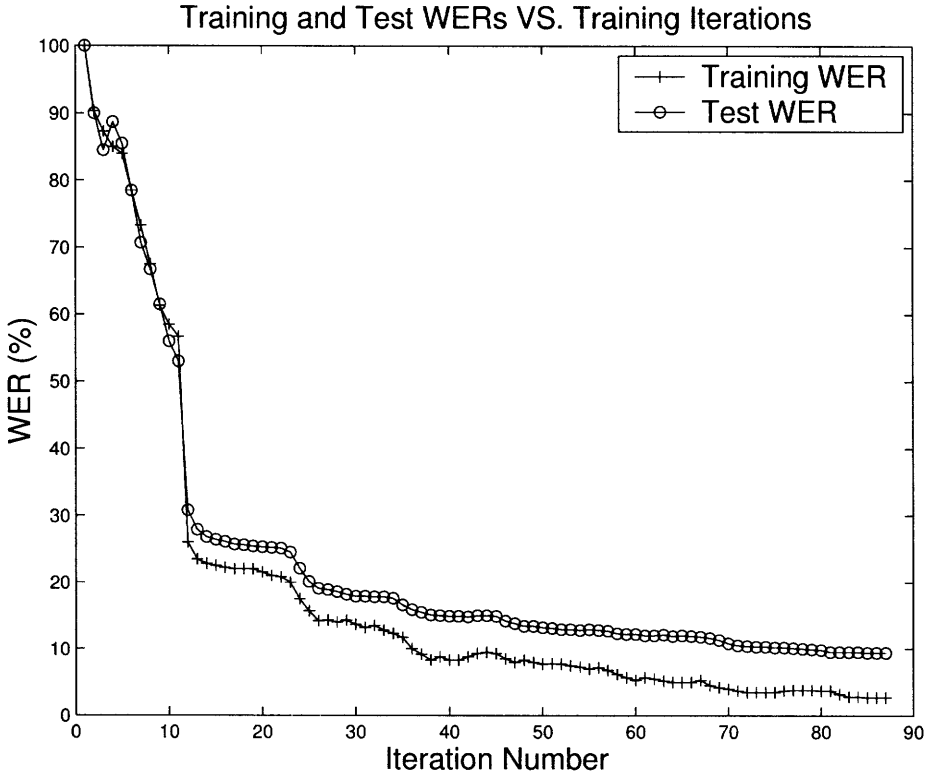


Figure 4-3: Training and test WERs as a function of training iterations. The upper curve is the test WERs, and the lower curve is the training WERs. The WERs of 100.0% from the first iteration is from the flat initialization models. As the training iteration increases, the number of parameters in the acoustic models also increases. At the 11^{th} iteration, the context dependent acoustic models are bootstrapped from the context independent acoustic models, and the over 20% drop in WER is due to this increase in the number of model parameters. After a total of 87 iterations, the training WER converges to 2.7%, and the test WER converges to 9.4%.

We are ultimately interested in exploring the benefits of combining frame-based and segment-based acoustic modeling. In the coming chapters, we will describe a multi-stream recognition framework implemented using the multi-tape FST. The multi-stream framework will allow us to investigate the fusion of the frame-based and the segment-based approaches and to explore a richer class of models.

Chapter 5

Multi-tape Finite-state Transducers

A multi-tape finite state transducer (mFST) is an extension of a finite state transducer where the arcs encode a n-tuple of symbols instead of an input and output symbol pair. The individual paths specified by the mFST still represent mappings between input and output label sequences. However, the input and output label sequences themselves can be n-tuples.

When the arcs of the mFST have associated weights, they are commonly referred to as *weighted* multi-tape FSTs in the literature. In this thesis, we will ignore this distinction, and interchangeably use mFST and weighted mFST to denote a weighted multi-tape FST. Similar to FST, the interpretation of the mFST weights depends on the particular choice of semiring. The same types of semirings used for FSTs can be used for mFSTs.

While not as widely used as FSTs, mFSTs have been successfully used for a number of speech and language applications. Johnston and Bangalore have used an mFST for multi-model (gestures and speech) parsing and tagging [35]. Kiraz has used an mFST for nonlinear morphology for Semitic languages [38].

In the remaining of the chapter, we will first formally define mFSTs and the composition operation for mFSTs. In Section 5.3, we present a novel Viterbi mFST search algorithm to find the path with the best weight in the composition result of

two mFSTs. The search algorithm is essential for the multi-stream framework for speech recognition which we will formulate in the coming chapter.

5.1 Formal Definition

A weighted multi-tape FST of n-dimensional tape $T^{(n)}$ over the semiring K is defined by a tuple $T^{(n)} = (\Sigma, \Omega, Q, E^{(n)}, i, F, \lambda, \rho)$ where Σ is the *input alphabet*, Ω is the *output alphabet*, Q is the finite set of *states*, $E^{(n)}$ is the finite set of *transitions*, i is the *initial state* where $i \in Q$, F is the set of *final states* where $F \subset Q$, λ is the initial weight associated with the initial state i, and ρ is the *final weights function*, $\rho : f \in F \rightarrow \Re$.

A transition $t^{(n)}$ is defined by a tuple, $t^{(n)} = (p[t^{(n)}], n[t^{(n)}], l_i[t^{(n)}], l_o[t^{(n)}], w[t^{(n)}])$. The transition $t^{(n)}$ is an arc from the *source state* $p[t^{(n)}]$ to the *destination state* $n[t^{(n)}]$ with the *input label* of $l_i[t^{(n_i)}] \in (\Sigma^*)^{n_i}$, *output label* of $l_o[t^{(n_o)}] \in (\Sigma^*)^{n_o}$, and weight of $w[t^{(n)}]$. The integers n_i and n_o are the dimensions of the input and output labels respectively, where $n = n_i + n_o$.

A set of N consecutive transitions connecting the initial state and a final state forms a *permissible path* (or simply *path*), written $\pi^{(n)} = t_1^{(n)} t_2^{(n)} \cdots t_N^{(n)}$, with $p[t_1^{(n)}] = i$, $n[t_N^{(n)}] \in F$, and for all $j = 1, 2, \cdots N-1$, $n[t_j^{(n)}] = p[t_{j+1}^{(n)}]$.

The input label sequence associated with path $\pi^{(n)}$ (or simply *input sequence*) is $l_i[\pi^{(n)}] = l_i[t_1^{(n_i)}] l_i[t_2^{(n_i)}] \cdots l_i[t_N]^{(n_i)}$. Similarly, the output label sequence associated with path $\pi^{(n)}$ (or simply *output sequence*) is $l_o[\pi^{(n)}] = l_o[t_1^{(n_o)}] l_o[t_2^{(n_o)}] \cdots l_o[t_N^{(n_o)}]$. It is important to note that implicitly $\epsilon \in \Sigma$, the symbol in individual tapes of the input or output sequence can be ϵ, representing the "empty" symbol.

The *path weight* for the path $\pi^{(n)}$ is $w[\pi^{(n)}] = \lambda \otimes w[t_1^{(n)}] \otimes w[t_2^{(n)}] \otimes \cdots \otimes w[t_N^{(n)}] \otimes \rho(n[t_N^{(n)}])$. The path weight for a set of paths is $w[\pi_1^{(n)}, \pi_2^{(n)}, \cdots, \pi_N^{(n)}] = w[\pi_1^{(n)}] \oplus w[\pi_2^{(n)}] \oplus \cdots \oplus w[\pi_N^{(n)}]$.

5.2 Generalized Composition

For single-tape FSTs, the composition operation of two FSTs A and B results in another FST T, $T = A \circ B$. FST T has the property that there exists one path π_T in T that maps the input label sequence $l_i[\pi_T]$ to the output label sequence $l_o[\pi_T]$ if and only if there exists a path π_A in FST A that $l_i[\pi_A] = l_i[\pi_T]$ and a path π_B in FST B that $l_i[\pi_B] = l_o[\pi_A]$ and $l_o[\pi_B] = l_o[\pi_T]$. The weight of the path in T is the $\otimes$ product of the weight $w[\pi_A]$ in A and the weight $w[\pi_B]$ in B.

We can generalize the composition operation for mFSTs in the following way. The generalized composition operation of two mFSTs $A^{(p)}$ and $B^{(q)}$ results mFST $T^{(n)}$, $T^{(n)} = A^{(p)} \circ B^{(q)}$. This generalized composition operation is valid only if the dimension of the output label of mFST $A^{(p)}$, p_o, is exactly the same as the dimension of the input label of mFST $B^{(q)}$, q_i. When the condition of $p_o = q_i$ is satisfied, the dimension of the input label of $T^{(n)}$ is the same as that of $A^{(p)}$, $n_i = p_i$, and the dimension of the output label of $T^{(n)}$ is the same as that of $B^{(q)}$, $n_o = q_o$.

The resulting mFST $T^{(n)}$ has the property that there exists one path $\pi_T^{(n)}$ in $T^{(n)}$ that maps the input label sequence $l_i[\pi_T^{(n)}]$ to the output label sequence $l_o[\pi_T^{(n)}]$ if and only if there exists a path $\pi_A^{(p)}$ in FST $A^{(p)}$ that $l_i[\pi_A^{(p)}] = l_i[\pi_T^{(n)}]$ and a path $\pi_B^{(q)}$ in FST $B^{(q)}$ that $l_i[\pi_B^{(q)}] = l_o[\pi_A]$ and $l_o[\pi_B^{(q)}] = l_o[\pi_T^{(n)}]$. The weight of the path in $T^{(n)}$ is the $\otimes$ product of the weight $w[\pi_A^{(p)}]$ in $A^{(p)}$ and the weight $w[\pi_B^{(q)}]$ in $B^{(q)}$.

Depending on the magnitude of p_o and q_i, this generalized composition operation can be very memory intensive. Since this operation can be done with an *on-demand* (or *lazy*) implementation, it is often desirable to carry out the composition operation in this manner. In the next section, we will present a novel Viterbi beam search algorithm for computing the path with best path weight in a mFST which itself is the result of a generalized composition of two mFSTs.

```
1   /* First initialize */
2   PriorityQueue ← initial state of A^(p)
3   DPNodes(initial state of A^(p)) ← [initial state of B^(q)]
4
5   /* Inner Loop */
6   foreach state2 ← pop(PriorityQueue)
7       foreach state1 ← [statesBackwardReachableFrom(state2)]
8           arc_A ← arc in A^(p) connecting state1 with state2
9           foreach DPNode ← DPNodes(state1)
10              foreach arc_B ← arcsLeavingFrom(DPNode.state)
11                  if arc_B is compatible with arc_A
12                      tmpDPNode.weight = DPNode.weight ⊗ w[arc_A] ⊗ w[arc_B]
13                      if tmpDPNode in search beam for DPNodes(state2)
14                          add tmpDPNode to DPNodes(state2)
15                      endif
16                  endif
17              end
18          end
19      end
20      propagate ε for all DPNodes(state2)
21      PriorityQueue ← [statesForwardReachableFrom(state2)]
22  end
23
24  /* Backtrace */
25  return BestPath ← backtrace start at DPNodes(final states of A^(p))
```

Figure 5-1: Pseudo-code for the Viterbi mFST search algorithm.

5.3 Viterbi Beam Search

Let mFST $T^{(n)}$ be the generalized composition result of two mFSTs $A^{(p)}$ and $B^{(q)}$, $T^{(n)} = A^{(p)} \circ B^{(q)}$. Here we generalize the Viterbi beam search used for single tape FST for the mFST case. The algorithm uses the standard Viterbi beam search to find the best single path in $T^{(n)}$. It is worth noting that the mFST $T^{(n)}$ does not have to be explicitly computed. The generalized composition, $A^{(p)} \circ B^{(q)}$, is computed on demand as part of the search.

Figure 5-1 shows the pseudo-code for the algorithm. We first initialize the priority queue with the initial state of $A^{(p)}$. The priority queue stores a list of states in $A^{(p)}$ that we "pull towards" during the search. *DPNodes* is an array indexed by the

topologically sorted states of $A^{(p)}$. Associated with each element of *DPNodes* array is a hash of states in $B^{(q)}$, and a score is associated with each hash element. The score stores the score of the current best theory associated with a pair of states, the state in $A^{(p)}$ and the state in $B^{(q)}$. We initialize the *DPNode* at the initial states of $A^{(p)}$ and $B^{(q)}$ with a score of $\overline{0}$.

After initialization, we loop through all the states in $A^{(p)}$. For each of these states (*state2* in the pseudo-code), we identify all the states (*state1* in the pseudo-code) that are backward reachable from *state2*. Note that the *state1* and *state2* forms a valid transition in $A^{(p)}$. For each of these transitions, we then identify all the transitions in $B^{(q)}$, new *DPNode* will be created for the *DPNode* array at *state1* if it survives the score- and count-based beam criterions.

At the end of the loop, the search will be at the final states in $A^{(p)}$ and $B^{(q)}$. Back trace information in the *DPNodes* provides the information for the best path in the generalized composition result of two mFSTs $A^{(p)}$ and $B^{(q)}$.

5.4 Summary

In this chapter, we have formally defined multi-tape FSTs and the composition operation for multi-tape FSTs. We also presented a novel Viterbi mFST search algorithm to find the path with the best weight in the composition result of two mFSTs. In the coming chapter, we will formulate the the multi-stream framework for speech recognition with mFSTs, and will also demonstrate the use of the Viterbi mFST search algorithm developed in this chapter for the the multi-stream framework.

Chapter 6

Multi-stream Speech Recognition with mFSTs

6.1 Introduction

In the previous chapter, we generalized the FST formulation to multi-tape FSTs. We also presented associated algorithms for generalized composition and Viterbi beam search. In this chapter, we use the mFST extension to construct a new approach to general multi-stream speech recognition. The multi-dimensional input labels of the mFST transitions specify the acoustic models to be used for the individual feature streams. An additional auxiliary field is used to specify the degree of asynchrony allowed among the feature streams. A novel aspect of this approach is that individual feature streams can be either linear sequences or directed acyclic graphs (DAGs). Traditional fixed- or variable-rate frames are examples of the linear sequence type of feature. Segment features on a hypothesized phonetic segment graph are an example of the DAG type of feature.

We first show the importance of modeling asynchrony in a multi-stream framework in Section 6.2. We then present the new multi-stream framework in detail in Section 6.3. In Section 6.4, we demonstrate the types of applications made possible by this new multi-stream framework with two examples. The first example involves combining the variable-rate landmark and segment features used in our baseline seg-

mental speech recognizer. The second example combines standard HMMs with landmark models. In Section 6.5 we report on experiments with this new framework: first, combining landmark models and standard HMMs on the Wall Street Journal speech recognition task; second, performing an audio-visual speech recognition experiment using the AV-TIMIT task.

6.2 Experiments for Combining Frame-based and Landmark Features

In Section 1.2.2, we discussed that the phone boundary locations hypothesized by HMM-based systems and segment-based landmark system can be very different. HMM-based systems are optimized to maximize recognition performance, not necessarily the accuracy in the phone boundary locations. Toledano et al. have reported that the phonetic alignments preferred by the context-dependent or context-independent HMMs are not consistent with human transcribed phone boundaries [48, 69]. In a segment-based landmark system, the phone boundary locations are hypothesized first, before the computation for the landmark features. Anecdotally, landmark-based phone boundary locations are better aligned to the human transcribed phone boundaries than the ones hypothesized with HMMs.

We carried out a set of experiments to test whether it is important to accommodate this difference when combining these two types of features. The experiment was carried out on the WSJ corpus, specifically the standard H2-C2 task on the Eval'92 test set [40, 52]. For training, we used the WSJ SI84 corpus. The training set contained 14 hours of speech with 7,138 sentences. The language model is a bigram with a decoding vocabulary of 5,000 words. The Eval'92 test set has 330 sentences with 5,353 words with 0.29% OOV rate.

For this task, the WER for the baseline HMM system is 9.5%, and the WER for the baseline landmark-based system is 10.4%. We first computed the "reference" phonetic boundary locations by aligning speech waveform with its corresponding reference word

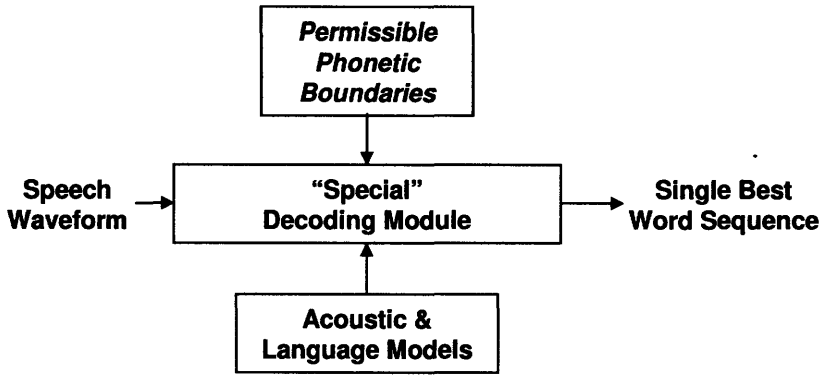


Figure 6-1: Flow chart for the "special" decoder which uses a set of permissible phonetic boundary locations.

transcription. We obtained the reference phonetic boundary locations for both the HMM-based and landmark-based systems. We will refer to these as *forced HMM phonetic boundaries* and *forced landmark phonetic boundaries.* We constructed a special decoder where the search space is additionally constrained by a sequence of permissible phonetic boundary locations. With this decoder, all hypotheses are guaranteed to be compatible with the input sequence of phonetic boundary locations. Figure 6-1 illustrates how this "special" decoder is used. When this decoder was used with forced HMM phonetic boundaries and HMMs models, we were able to verify that the WER is the same as the baseline system. The same was found to be true for the landmark models with forced landmark phonetic boundaries. However, when we used this decoder with forced HMM phonetic boundaries and landmark models and with forced landmark phonetic boundaries and HMMs, both decoding results degraded relative to their respective baselines. Table 6.1 summarizes these results. These results highlight the difference in phonetic boundary locations preferred by these two different systems. Thus, when building a system combining both these types of features, one needs to accommodate this difference in phonetic boundary locations. If not dealt with properly, this difference can potentially have an adverse effect on the overall system. Therefore, allowing some degree of asynchrony between HMMs and other models may be critical to successful integration with a multi-stream framework.

Type of Acoustic Model	Type of Phone Boundaries	WER
HMM	None	9.5%
HMM	Forced using HMMs	9.5%
HMM	Forced using landmarks	10.5%
Landmarks	None	10.4%
Landmarks	Forced using Landmarks	10.4%
Landmarks	Forced using HMMs	12.6%

Table 6.1: Study of the compatibility of phonetic boundary locations preferred by HMMs and landmark models.

6.3 Multi-stream, Multi-tape FST Framework

6.3.1 FST Cascade with mFSTs

In Section 2.3.3, we described how FSTs are used for the recognition problem with one input feature stream. In this section, we generalize this to arbitrary F feature streams allowing asynchrony using a multi-tape FST representation. Recall that for the single feature stream case, the recognition problem is equivalent to the problem of searching for the best path in $AMCPLG$, where $AMCPLG = A \circ M \circ C \circ P \circ L \circ G$. In this equation, all the FSTs are single-tape FSTs.

By generalizing some of the FSTs in the cascade $AMCPLG$ to be mFSTs, we can use a similar framework for the problem of recognition with an arbitrary number of feature streams. Figure 6-2 illustrates this generalization. mFST $A^{(p)}$ represents the acoustic models with multi-stream feature vectors. The dimensions of the input and output labels of the mFST $A^{(p)}$ are both F, $p_i = p_o = F$. mFST $M^{(q)}$ represents the model topology for the multi-stream feature vectors. The dimension of the input label of the mFST $M^{(q)}$ is F, $q_i = F$, and the dimension of the output label is 1, $q_o = 1$. FSTs C, P, L, and G remain single-tape as before.[1] In the next two subsections, we will describe how the mFSTs $A^{(p)}$ and $M^{(q)}$ are formulated, then we will discuss how we modified the Viterbi search for the FST cascade $AMCPLG$ when $A^{(p)}$ and $M^{(q)}$ are multi-tape FSTs.

[1] C, P, and L can also be represented with mFSTs, but for simplicity we will only focus on the case where $A^{(p)}$ and $M^{(q)}$ are mFSTs.

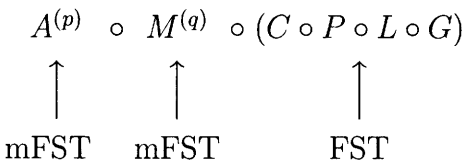


Figure 6-2: Illustration of the multi-stream framework using mFSTs. Notice that both A and M are mFSTs, and FSTs C, P, L, and G are single-tape.

6.3.2 Multi-Stream Acoustic Model mFST $A^{(p)}$

For a single-stream system, the acoustic model FST A associates a sequence of features with a sequence of acoustic models. On each arc of the acoustic model FST, a single feature vector is associated with a single acoustic model, and the weight on the arc represents the acoustic likelihood score of the feature vector for the given acoustic model. The semiring of the single-tape acoustic model FST is defined such that the path weight is equal to the total likelihood of a sequence of feature vectors for the associated sequence of acoustic models.

For a system with F feature streams, the *multi-tape* FST $A^{(p)}$ specifies how the multiple streams of feature vectors are associated with their corresponding acoustic models. On the input side, the mFST also specifies the set of all possible sequence of multi-stream feature vectors. The asynchrony among the features can be encoded in a number of ways. The mFST $A^{(p)}$ can contain all permissible transitions among all the possible sequence of multi-stream feature vectors. Instead, we chose to encode the feature-space asynchrony in two separate components. First, the mFST $A^{(p)}$ is encoded with *all* possible transitions in the F-dimensional feature stream space where all possible asynchrony among the feature streams are permitted. Second, the degree of asynchrony permitted among the feature streams are encoded as part of "time predicts" in the mFST $M^{(q)}$ which will be discussed in detail in the next section. We chose this approach because the mFST in the first component can be represented compactly and the "time predicts" offers high degree of flexibility for specifying the asynchrony among the feature streams.

The mFST $A^{(p)}$ representing *all* possible transitions in the F-dimensional feature

stream space where all possible asynchrony among the feature streams are permitted can be expressed compactly. In Section 2.3.3, we showed how individual feature streams either of a linear sequence type or directed acyclic graph type can be represented with single-tape FSTs. Let A_i be the single-tape FST representing the i^{th} feature stream. Figure 6-3 illustrates example single-tape FSTs for both the linear sequence feature and directed acyclic graph feature. Note that the states in the FSTs are associated with the time stamp of the corresponding feature vectors. Since each feature vector is uniquely identifiable with the corresponding time, each single-tape FST can be thought of as a graph containing potential time transitions. Given the single-tape FSTs A_i used for the individual feature stream, the mFST $A^{(p)}$ encoding with all possible transitions can be thought as the "cross-product" of all the individual feature stream single-tape FSTs. This presentation is very large if it is encoded explicitly. The number of states in this network is equal to the products of the number of states of the individual A_i feature stream FSTs. For a given hyper-state j in this "cross-product" mFST, let $t_i(j)$ be the number of arcs leaving the corresponding the states in the individual A_i feature stream FSTs. The number of arcs leaving the hyper-state j is equal to $\Pi_{i=1}^{F}(t_i(j)+1)-1$ since any combination of one or more (up to F) feature transitions are permitted. Fortunately the "cross-product" can be expressed implicitly linear with the number of feature streams.

The mFST $A^{(p)}$ is simply represented by the set of single-tape FSTs A_i, and we generate the "cross-product" on-the-fly as needed. Since each feature vector is uniquely identifiable with their corresponding time, each mFST state in this "cross-product" mFST can be uniquely identified by a F-tuple $(t^{(1)}, t^{(2)}, \ldots, t^{(F)})$, representing the current time (or state) across all feature streams. We call such a time F-tuple a "*hypertime*" $\mathbf{t}$. With this "cross-product" mFST, the feature streams are allowed to de-synchronize arbitrarily with each other in time. The amount of synchrony can be specified in $M^{(q)}$ by the use of *time predicates*, which we will discuss in more detail in the next subsection. On the output side of the mFST, individual acoustic models are associated with the corresponding feature vectors. The weights on the arcs of the mFST is chosen to be a linear combination of the individual acoustic models scores.

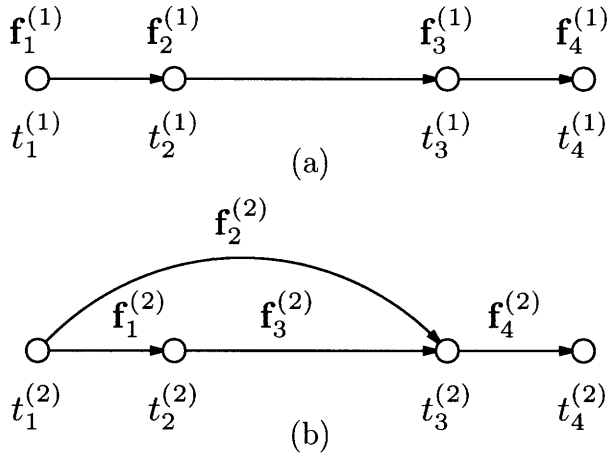


Figure 6-3: Example single-tape FST representing both a linear sequence type feature and a directed acyclic graph type feature. (a) represents an example type of linear sequence features, variable-rate landmark features. Each FST state represents a feature vector $\mathbf{f}_i^{(1)}$ and time $t_i^{(1)}$ associated with each feature. (b) represents an example type of the directed acyclic graph features, the segment features. Each FST arc represents a feature vector $\mathbf{f}_j^{(2)}$ associated with a segment, and each FST state is associated with time $t_k^{(2)}$ representing the starting time of the segment features associated with the leaving arcs from the state.

The linear combination weights for the feature streams are computed by optimizing the recognition performance of the entire system on a development set.

6.3.3 Model Topology mFST $M^{(q)}$

We make use of a multi-tape FST representation for the model topology with $M^{(q)}$. In the single-tape case, the single-tape model topology FST M used for a three-state HMM specifies the allowable HMM state transitions. In the multi-tape case, the multi-tape model topology mFST $M^{(q)}$ also specifies the allowable state transitions. These state transitions in a F-dimension hyper-state space. The synchronization among these hyper-states are also specified with the mFST transition labels.

Each transition label of the mFST is in the form

$$m^{(1)} : m^{(2)} : \ldots : m^{(F)} : p : o \,/\, w \,. \tag{6.1}$$

The input side of $M^{(q)}$ is multi-tape, where $q = F + 1$. The first F tapes encode the acoustic models used for the feature streams. The i^{th} tape of the multi-tape acoustic

model corresponds to the i^{th} feature stream. These first F tapes encode allowable transitions through the F feature streams space.

The $F + 1^{th}$ tape, represented by the symbol p, is used for time predicates. The output side of $M^{(q)}$ is single-tape which specifies the phone-level units, o. w represents the model transition weights (e.g., $-$log probabilities). Each $m^{(f)}$ represents a model identifier for feature stream f, or ϵ if there is none. p identifies a predicate to be applied to the hypertime $\mathbf{t}$, controlling the degree of asynchrony (in time, as opposed to in states) between the feature streams at any given point in the search, or it too can be ϵ for no predicate.[2]

6.3.4 Search

Since the output symbols of $M^{(q)}$ are single-tape, and C, P, L, and G are also single tape, we can compute the FST cascade $MCPLG = M^{(q)} \circ C \circ P \circ L \circ G$ the same way as if they are all single-tape. The recognition problem for multi-stream feature vectors is equivalent to the problem of searching for the *best path* in the generalized composition between the mFST $A^{(p)}$ and the mFST $MCPLG$. Here, the best path is defined to the path with the best associated score. The score is equal to the sum of the linear combination score of acoustic models used for the feature stream, the transition weights in the model topology, and the corresponding language model score. In Section 5.3, we had developed a novel Viterbi beam search for the generalized composition of two mFSTs. In this section, we will discuss how to apply this algorithm for this specific case of $A^{(p)} \circ MCPLG$.

Figure 6-4 outlines the pseudo-code for this algorithm. To make a transition within the mFST $A^{(p)}$, equivalently in hypertime space from $\mathbf{t}_1 \rightarrow \mathbf{t}_2$, the presence of the $m^{(f)}$ model identifiers constrains the possible hypertime transitions. If $m^{(f)} \neq \epsilon$ (i.e., model present for stream f), then feature space f must make a transition: $t_2^{(f)} > t_1^{(f)}$. Otherwise, there will be no transition: $t_2^{(f)} = t_1^{(f)}$. In addition, the predicate p on the transition must evaluate to true for the destination hypertime: $p(\mathbf{t}_2) = 1$ if a joint FST and feature space transition is to take place. The predicate could be generalized

[2]Figures in this section have been simplified and do not show the o and w components on FSTs.

```
1   /* First initialize */
2   PriorityQueue ← initial state of A^(p) or hypertime (0, 0, ..., 0)
3   DPNodes(initial state of A^(p)) ← [initial state of M^(q)]
4
5   /* Inner Loop */
6   foreach t_2 ← pop(PriorityQueue)
7       foreach t_1 ← [statesBackwardReachableFrom(t_2)]
8           arc_A ← arc in A^(p) connecting t_1 with t_2
9           foreach DPNode ← DPNodes(t_1)
10              foreach arc_B ← arcsLeavingFrom(DPNode.state)
11                  if ((hypertime and model space transitions match)
12                          and (t_2 satisfies timePredict(arc_B)))
13                      tmpDPNode.weight = DPNode.weight ⊗ w[arc_A] ⊗ w[arc_B]
14                      if tmpDPNode passes score- and count-based pruning for DPNodes(t_2)
15                          add tmpDPNode to DPNodes(t_2)
16                      endif
17                  endif
18              end
19          end
20      end
21      propagate ε for all DPNodes(t_2)
22      PriorityQueue ← [statesForwardReachableFrom(t_2)]
23  end
24
25  /* Backtrace */
26  return BestPath ← backtrace start at DPNodes(final states of A^(p))
```

Figure 6-4: Pseudo-code for the Viterbi search for the multi-stream recognition framework.

to model the probability of a given degree of asynchrony.

When an FST transition is taken, the score is updated by linear combination of the log probabilities provided by the individual feature classifiers. Note that this is a substantial difference from Bourlard et al.'s HMM recombination framework [4], in which the stream scores are integrated only at synchronization states. Earlier integration has the potential advantage that different streams can contribute to beam pruning earlier in the search, though it does limit the possible forms of feature score combination.

The dynamic-programming search finds the best path through the generalized composition of the mFST $A^{(p)}$ and the mFST $MCPLG$. The search starts at the

initial hypertime and proceeds to the final hypertime. The hypertimes are traversed in lexicographically sorted order. Other orderings of the hypertimes can also be used. We chose the lexicographically sorted order due to its simple implementation. Beam pruning is performed at every *DPNode* as described in Figure 6-4. The beam pruning is both count-based and score-based. To further prune the search space, we can also perform beam pruning across all *DPNode* with the same $t^{(1)}$ in hypertime. The first dimension of the hypertime, $t^{(1)}$, typically stores the feature with the finest time resolution.

The acoustic models for the individual features streams are trained separately. Both the acoustic scores and transition weights of individual acoustic models are linearly combined for the multi-stream models. The linear combinations are weighted and the weights are optimized on a development set. In the future, we plan to investigate training the acoustic models jointly.

6.4 Examples with the mFST Framework

6.4.1 Landmarks and Segments

In Section 2.2.1, we have described the usage of landmark and segment features for segment-based speech recognition. The landmark feature framework is motivated by the belief that acoustic cues important for phonetic classification are located at acoustic landmarks corresponding to oral closure (or release) or other points of maximal constriction (or opening) in the vocal tract [65]. The segment features are computed from the portion of the speech waveform belonging to a hypothesized phonetic segment, and the landmark features are computed from fixed-size waveform intervals centered at landmarks. As described in Section 2.2.2, an over-generated phonetic segmentation network facilitates computation of the landmark and segment features. Both of these features are derived from the fixed-rate (5ms) MFCC features. The landmark feature stream is a variable-rate sequence of fixed-length feature vectors. This is similar to the features used in variable-rate HMMs [5]. Figure 6-5(a) shows

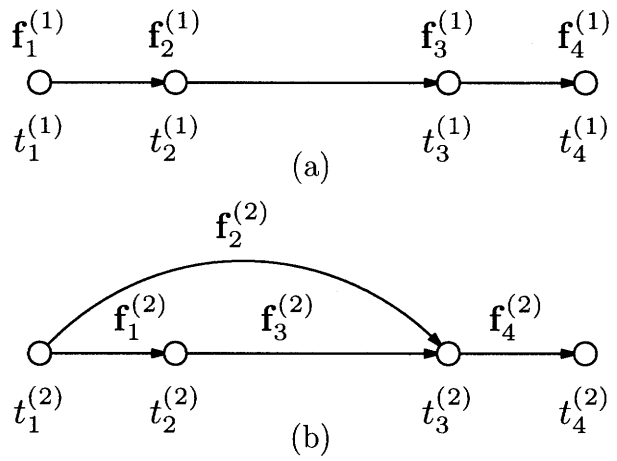


Figure 6-5: 2-stream feature space combining landmark and segment features. Stream 1 in (a) represents variable-rate landmark features, with a feature vector $\mathbf{f}_i^{(1)}$ and time $t_i^{(1)}$ associated with each landmark i. Stream 2 in (b) represents the segment features. Each segment connecting pairs of landmarks, with a feature vector $\mathbf{f}_j^{(2)}$ associated with each segment j and time $t_k^{(2)}$ associated with each segment boundary k.

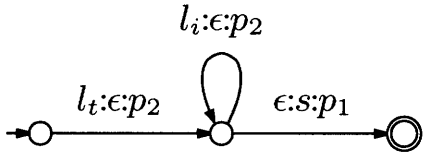


Figure 6-6: Model Topology $M^{(q)}$ for 2-stream landmark/segment phonetic model using mFST. The single stream landmark model is a 2-state HMM, l_t is the transition model, and l_i is the internal model. The single stream segment model is a 1-state whole-segment model s.

the FST representation of a landmark feature sequence. The segment feature cannot be represented by a single linear sequence, but it can be represented by a directed acyclic graph. Figure 6-5(b) shows an FST representation of the segment features. The FST states in Figure 6-5 are also augmented with time stamps of the corresponding feature vectors. The time stamps are used by the time predicates in the model topology mFST.

The single stream landmark model is a 2-state HMM; l_t is the transition model, and l_i is the internal model. The internal model l_i can be skipped or used multiple times. The single stream segment model is a 1-state whole-segment model s. As described in Section 2.2.4, an "antiphone" model is used for the segment model to account for the different segment observation sequences along the different segmentation

paths. The landmark models (transition and internal) and segment models are all context-dependent. At phone boundaries, the landmark and segment feature streams are fully synchronized in time (i.e., $t^{(1)} = t^{(2)}$). Our segmental speech recognizer [27] can use both the landmark and segmental acoustic feature streams jointly. Unlike the new multi-stream framework with mFST, it does not flexibly integrate other types of feature vectors. In this section, we present the mFST version of combining these two features.

Figure 6-6 shows the model topology $M^{(q)}$ for the 2-stream landmark/segment phonetic model using mFST. Tape 1 in the mFST represents models for the landmark feature stream, tape 2 represents the models for the segment feature stream, and tape 3 contains the time predicates used for specifying asynchrony between the two features.

Predicate $p_1(\mathbf{t})$ enforces the degree of asynchrony between the landmark and segment features permitted at phone boundaries. Here, the time predicate $p_1(\mathbf{t})$ is in the form of $|t^{(1)} - t^{(2)}| \leq \tau$. In the baseline system, the landmark and segment features are required to be exactly synchronized, i.e., $\tau = 0$. In this multi-stream framework p_1 allows us to relax this synchrony constraint if needed. Predicate p_2 is used to prevent feature stream 1 from being explored too far ahead of stream 2 unnecessarily. The time predicate $p2(\mathbf{t})$ is in the form of $t^{(1)} \leq \max\text{reachable}(t^{(2)}) + \tau$, where "max reachable" holds the maximum finishing time of any segments starting at $t^{(2)}$. Use of time predicate $p2$ improves the efficiency of the search by eliminating dead ends earlier.

6.4.2 Frames and Landmarks

Figure 6-7 shows the mFST representing the model topology for a 2-stream speech recognition system combining a 3-state HMM and a 2-state landmark model. Here, we interleaved the two models so that the partial-phone frame scores and landmark scores can be integrated as early as possible. Early integration of the scores enables more effective beam pruning during the search. The configuration shown is not the only one that is suitable. For example, a full Cartesian product of the individual

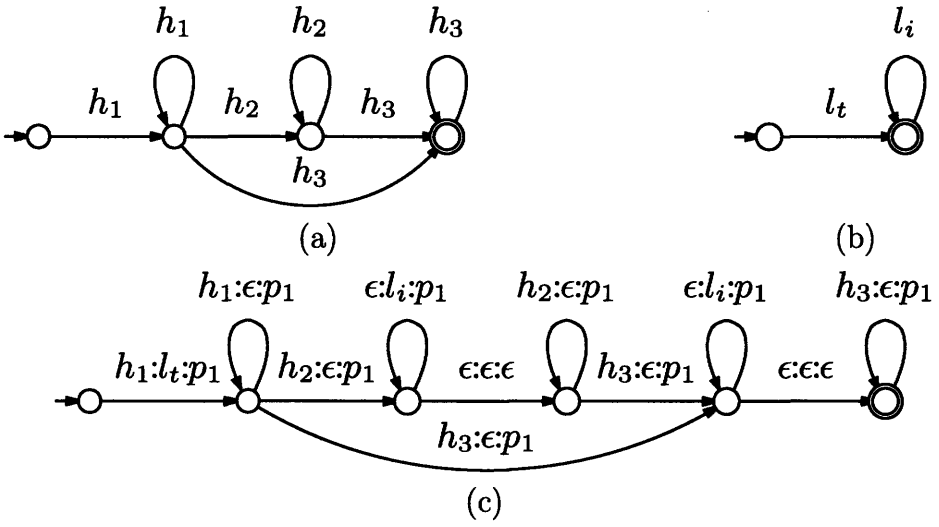


Figure 6-7: (a) Model topology a single-stream HMM. (b) Model topology a single-stream landmark model. (c) Model Topology $M^{(q)}$ for the 2-stream landmark and HMM system using mFST. It is a combination of the single-stream FSTs shown in (a) and (b). Other configurations of the mFST are possible, including a full Cartesian product. (c) shows the topology used in our experiments.

stream FSTs can also be used. Since there is no additional constraint needed at the individual landmark model and HMM states, the full Cartesian-product configuration did not offer any additional advantage with our multi-stream framework.

The time predicate $p_1(\mathbf{t})$ is in the form of $|t^{(1)} - t^{(2)}| \leq \tau$. This time predicate permits the frame and landmark streams to be out of sync by up to τ, both between and within phones.

6.5 Experiments

We have experimented with the multi-stream framework on two different tasks. One is the Wall Street Journal (WSJ) speech recognition task, and the other is an audio visual speech recognition task on the AV-TIMIT corpus [31].

6.5.1 The WSJ Task

The WSJ corpus consists of read speech of sentences from the *Wall Street Journal* newspaper. We chose to do the standard H2-C2 task on the Eval'92 test set [40, 52].

For training, we used the WSJ SI84 corpus. The training set contains 14 hours of speech with 7,138 sentences. The language model is a bigram with a decoding vocabulary of 5,000 words. The Eval'92 test set has 330 sentences with 5,353 words with 0.29% OOV rate.

Two baseline systems were used for this WSJ task. The first baseline system is a standard HMM system. The 42-dimensional feature vector consists of 13-dimensional MFCCs and the energy, the deltas, and the delta-deltas, and they are computed with a fixed 10ms frame shift. The 3-state HMM acoustic models have 3,347 clustered triphone models with 26,742 Gaussians. The word error rate (WER) for the baseline HMM system is 8.8%. The second baseline system uses a 50-dimensional landmark feature vector. They are computed at hypothesized landmark locations. The feature vector can be thought as a variable rate sampling of the acoustic waveform. The average landmark spacing is approximately 30ms. The baseline landmark acoustic models had 993 clustered diphone models with 13,496 Gaussians. The WER using the landmark models is 10.4%. Two separate reasons probably contribute to the higher WER of the landmark models. First, the landmark models are diphone models which have less number of parameters than the triphone HMMs. Second, the acoustic segment network in some cases do not contain the "optimal" segmentation since using a fully connected segment network improves recognition performance with increased computation cost. "Segmentation by recognition" suggested by Chang has demonstrated improved recognition performance [6]. The goal here is not to construct the best baseline system possible, but to test whether the novel multi-stream framework can effectively integrate the feature streams in the baseline systems.

The multi-stream decoder provides a flexible framework to combine these two baseline feature streams. The multi-tape FST representation of the phone model used for these two streams is the same one illustrated in Figure 6-7(a).

Table 6.2 summarizes the results in terms of WERs of the baseline landmark models, HMM models, and their combined models. The combined acoustic models achieved a WER error rate of 8.0%, which improves from either baseline configurations alone. A development set was used to optimize the weighting of the landmark and

Acoustic Models	Test WER
Landmark Models	10.4%
HMM Models	8.8%
Landmark + HMM Models	8.0%

Table 6.2: Word error rates (WER) for variable-rate landmark models, fixed-frame-rate HMMs, and their combined models.

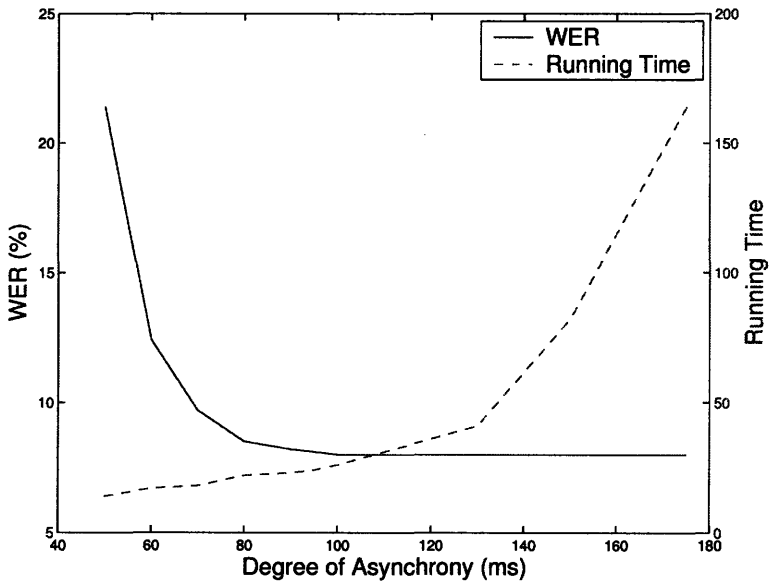


Figure 6-8: WERs and decode time vs. degree of asynchrony.

HMM scores.

The degree of asynchrony allowed in the time predicates has a significant impact on performance in terms of WER and computation time. In general the landmarks and the HMM features are not aligned in time, and a strict requirement of all the phonetic boundaries are synchronized at the same locations will most likely not produce any complete hypothesis. Figure 6-8 shows how the WER and the computation time changes as the degree of asynchrony varies. When the asynchrony between the two streams is at least 95ms ($\tau \geq 95$ms), the WER does not improve and the computation time increases. The two feature stream need a minimum amount of asynchrony so a compatible hypothesis can be considered. The computation time increases with increasing degree of asynchrony because the size of search space increases due to an increasing number of hypertimes visited.

6.5.2 The AV-TIMIT Task

The multi-stream framework can also be applied to other recognition tasks. Here we show its use for audio-visual speech recognition on the AV-TIMIT corpus. The AV-TIMIT corpus is a collection of audio-visual speech data of many speakers reading phonetically rich TIMIT sentences. Along with the speech waveform, the facial movement of the speakers were also captured in video. The training set consists of 3,608 utterances from 185 speakers, and the test set contains 285 utterances from other 19 speakers [31].

Two baseline systems were used for this task. The first baseline system was a segment-based system using both landmark and segment features from audio data only. The second baseline system was a frame-based 3-state HMM system modeling only the visual features. The multi-stream system modeled these three feature streams together.

Table 6.3 summarizes the results in terms of WERs of the baseline speech landmark and segment models, visual HMM models, and their combined models. The visual HMM models alone performed poorly. This is not surprising since the visual information of the mouth movement contains only partial information of the linguistic

Acoustic Models	Test WER
Speech Landmark & Segment	2.27%
Visual HMM Models	96.3%
Speech Landmark & Segment + Visual HMM Models	0.91%

Table 6.3: WERs for speech landmark and segment models, visual HMMs, and their combined models.

message. The WER of the combined models is the same as reported in [29], where a custom designed decoder was used for combining the same 3 feature streams. The decoding speeds of the two decoders were approximately the same.

6.6 Discussion and Future Work

In this chapter, we presented a new framework for multi-stream speech recognition. The framework takes advantage of the multi-tape FST representation for specifying the various constraints among the individual feature streams. Asynchrony among the various feature streams can also be specified with the novel use of time predicates. Because the degree of asynchrony permitted can significantly change the size of the search space, good design of the time predications themselves is important to achieving good decoding performance. The new framework can accommodate a greater variety of feature types, including fixed-rate and variable-rate sequences of frames, as well as directed acyclic graphs. In the single-stream mode, both traditional frame-based HMMs and segment-based systems can be represented. In the multi-stream model, a great variety of feature types can be integrated. The additional constraint among the various types of features can potentially improve the overall recognition performance. We carried out two experiments using the multi-stream speech recognition framework in this chapter. One experiment combined a traditional frame-based HMM with segment-based landmark features for the Wall Street Journal speech recognition task, while the other experiment combined a landmark model, a segment model, and a visual HMM for the AV-TIMIT task. Both experiments demonstrated improved recognition performance over their single-stream baseline experiments.

Many types of features have been shown to be useful for speech recognition, such

as sub-band features and articulatory features [12, 39, 45, 47]. In the future, we plan to experiment with other types of feature streams with this new multi-stream framework. In general, however, the size of the search space is exponential in the number of feature streams. If the decoding is computationally too demanding, a multi-pass decoding approach may be needed. In this multi-pass decoding approach, a first decoding pass with a subset of the feature streams generates a phone or word lattice. Using all the feature streams and the phone or word lattice from the first pass, a second decoding pass searches for the hypothesis with the best score. Thus far we have focused on synchronization at the phonetic level. We plan to investigate synchronization at other levels, such as the syllable level.

Chapter 7

Conclusions

7.1 Summary

In this thesis, we have focused on improving the acoustic modeling of speech recognition systems to increase the overall recognition performance by machines. We have formulated a multi-stream speech recognition framework using multi-tape FSTs. This multi-stream framework is novel in the following ways:

- The multi-dimensional input labels of the multi-tape FST transitions specify the acoustic models to be used for the individual feature streams. The topology of the multi-tape FSTs are used to specify various constraints among the feature streams.

- In many cases, the asynchrony among the various feature streams needed to be modeled specifically. An additional auxiliary field as part of the multi-tape FST transition is used to model the degree of asynchrony among the feature streams.

- The individual feature streams can be linear sequences such as fixed-frame-rate features in traditional HMM systems, and the feature streams can also be directed acyclic graphs such as segment features in segment-based systems.

- A Viterbi-based search algorithm has been developed for this multi-stream speech recognition framework. This algorithm can accommodate any model topology that can be specified with the multi-tape FSTs.

- In a single-tape mode, this multi-stream framework also unifies the frame-based HMM and the segment-based approach. Both EM-based training algorithm and Viterbi-based search algorithm have been developed for this single-tape mode.

In Chapter 6, we used the multi-stream speech recognition framework for an audio-only and an audio-visual speech recognition task. On the Wall Street Journal speech recognition task, the multi-stream framework combined a traditional frame-based HMM with segment-based landmark features. The system achieved word error rate (WER) of 8.0%, improved from both the WER of 8.8% of the baseline HMM-only system and the WER of 10.4% of the landmark-only system. On the AV-TIMIT audio-visual speech recognition task, the multi-stream framework combined a landmark model, a segment model, and a visual HMM. The system achieved a WER of 0.9%, which also improved from both the WER of 2.27% of a combined audio-only landmark and segment baseline system and the WER of 96.3% of a visual-only HMM system. These results demonstrate the feasibility and versatility of the multi-stream speech recognition framework.

7.2 Future Directions

7.2.1 EM Training of FST Weights

In Chapter 3, we presented a novel method to train FSTs directly via the EM algorithm. The method operates on any generic FST, even those with ϵ transitions. We applied the EM training of FST weights for the pronunciation weighting problem. The application of this algorithm for the problem of pronunciation weight training is the first successful use of this type of algorithm. The FST EM algorithm can have many applications other than pronunciation weight learning. It has also been used in

an FST-based speech synthesis system [61]. To facilitate wide use of this algorithm, we have included this as part of an open source FST toolkit [33].

7.2.2 Frame-based and Segment-based Speech Recognition

In Chapter 4, we have extended the EM training algorithm for FST weights to EM training for acoustic models that can be represented by FSTs. Since we can represent both frame-based and segment-based acoustic models as FSTs, this training algorithm completes the common framework for both frame-based and segment-based speech recognition systems. With this common framework, one can use the same generalized algorithms for training and decoding of frame-based or segment-based speech recognizers. This common framework enables a direct comparison of the frame-based and the segment-based approaches. We have preliminarily explored the effect of the segmentation network on the overall systems performance. It suggested that the "standard" SUMMIT acoustic segmentation algorithm for generating the segmentation network is too restrictive. While the resulting segmentation network improves the decoding time, it is doing so at the expense of recognition error rate. Improvements for the computation of segmentation network has been recently [59]. Further comparison between the frame-based and segment-based approaches are needed using this framework.

7.2.3 Multi-Stream Speech Recognition Framework

In this thesis, we have focused on the framework for multi-stream speech recognition. The framework can accommodate a large class of multiple feature streams by allowing the feature streams to be either linear sequences or directed acyclic graphs and by the use of time predicates on the multi-tape FST transitions. We have demonstrated that the use of framework through two experimental systems using the framework. As noted in Chapter 1, the types of feature streams used for speech recognition can be quite diverse. Much works remain to be done for experimenting with combinations of these feature streams. So far, we have focused on phone-level and subphone-level

features, syllable-level and whole-word-level features should also be considered. As we experiment with more feature streams, different model structure and different asynchrony constraints may be needed to achieve optimal performance.

In this thesis, we have not focused on the computational complexity of using multiple feature streams. In general, however, the size of the search space is exponential in the number of feature streams. If the decoding is computationally too demanding, a multi-pass decoding approach may be needed. In this multi-pass decoding approach, a first decoding pass with a subset of the feature streams generates a phone or word lattice. Using all the feature streams and the phone or word lattice from the first pass, a second decoding pass searches for the hypothesis with the best score.

The multi-stream framework presented in this thesis provides a flexible way to integrate the feature stream during the search. This integration method is typically referred to as "early integration". It would be instructive to compare this framework with other approaches, such as ROVER, a "late integration" approach. [22] We did not perform this comparision on the landmark and HMM combination because ROVER without word confidences requires at least three seperate recognition outputs. The performance of combining two recognition outputs with word confidences using ROVER depends on the quality of the word confidences, the usage of word confidences can complicate the intepretation of the results. We plan to perform this comparision when at least three seperate recognition outputs are available.

In Chapter 3, we presented a novel method to train FSTs directly via the EM algorithm. In Chapter 4, we extended the EM training algorithm for FST weights to EM training for acoustic models that can be represented by FSTs. Within the multi-stream speech recognition framework, the acoustic models for the individual feature streams are trained separately. It should be possible to extend the algorithms in Chapters 3 and 4 for FSTs to the case of multi-tape FSTs. By doing so, the acoustic models can be jointly trained. Intuitively, the jointly trained models should outperform the separately trained models if enough training data is available.

7.3 Conclusions

The primary contributions of this thesis are detailed below:

- We formulated a multi-stream recognition framework with a multi-tape finite-state transducer. This multi-stream framework accommodates multiple streams of features which can be a mixture of sequential and graph features, and it also allows controllable asynchrony across the feature streams. We demonstrated the capabilities on the WSJ task with HMM frame-based features and segment-based landmark features and on a audio-visual recognition task with HMM frame-based features and segment-based landmark and segment features.

- We introduced a single-stream recognition framework based on the finite-state transducer cascade with support for both sequential and graph features. With the existing beam search and newly developed EM-based training for this framework, it freed the dependency on initialization models for the framework and enabled direct comparison among various kinds of recognition systems (e.g., frame-based and segment-based) supported by the framework.

- We developed a novel EM-based weight training algorithm for learning FSTs weights from data. We applied this algorithm for the problem of learning pronunciation weights for the FSTs inside the FST cascade, we showed improved recognition performance with learned pronunciation weights over the unweighted baseline system.

The most significant contribution of thesis is the unified framework for multi-stream speech recognition. While there have been previous efforts to use multiple streams together for recognition, this framework uses the multi-tape FST for flexible specification of the model topology and asynchrony among the feature streams, and it can accommodate both linear sequence and direct acyclic graph features. From two experiment systems, we demonstrated the flexibility and versatility of the multi-stream speech recognition framework. We have not done extensive experimentation to optimize the combination of feature streams for specific speech recognition task.

With this framework, we have constructed a platform for others to experiment with multi-stream recognition. We hope that the multi-stream framework will encourage more researchers to investigate in this direction.

Appendix A

Phonetic Alphabet

label	description	example	transcription
[iy]	high front tense	sweet	[s w iy t]
[ih]	high front lax	Bill	[b ih l]
[ey]	middle front tense	ate	[ey t]
[eh]	middle front lax	head	[h eh d]
[ae]	low front lax	after	[ae f t er]
[er]	high central lax r-colored (stressed)	bird	[b er d]
[axr]	high central lax r-colored (unstressed)	creature	[k r iy ch axr]
[uh]	middle central lax (stressed)	butter	[b uh dx axr]
[ax]	middle central lax (unstressed)	about	[ax b aw t]
[ay]	low central tense diphthong	kite	[k ay t]
[aw]	low central lax	flower	[f l aw er]
[aa]	low back lax	hot	[h aa t]
[ow]	middle back tense rounded	goat	[g ow t]
[oy]	middle back tense rounded diphthong	toy	[t oy]
[ao]	middle back lax rounded	bought	[b ao t]
[uw]	high back tense rounded	smooth	[s m uw dh]
[uh]	high back lax rounded	wood	[w uh d]

Table A.1: *The vowels of the ARPABET phonetic alphabet, with descriptions and examples. Based on http://www.billnet.org/phon/arpabet.php*

label	description	example	transcription
[p]	voiceless bilabial stop	put	[p uh t]
[t]	voiceless alveolar stop	top	[t aa p]
[k]	voiceless velar stop	crazy	[k r ey z iy]
[b]	voiced bilabial stop	buy	[b ay]
[d]	voiced alveolar stop	dull	[d uh l]
[g]	voiced velar stop	bug	[b uh g]
[m]	voiced bilabial nasal	mouth	[m aw th]
[n]	voised alveolar nasal	night	[n ay t]
[ng]	voiced velar nasal	sing	[s ih ng]
[f]	voiceless labiao dental fricative	find	[f ay n d]
[v]	voiced labio dental fricative	vine	[v ay n]
[th]	voiceless dental fricative	cloth	[k l aa th]
[dh]	voiced dental fricative	clothe	[k l ow dh]
[s]	voiceless alveolar fricative	see	[s iy]
[z]	voiced alveolar fricative	zoo	[z uw]
[sh]	voiceless palato-alveolar fricative	cash	[k ae sh]
[zh]	voiced palato-alveolar fricative	leisure	[l iy zh axr]
[ch]	voiceless palato-alveolar affricate	chicken	[ch ih k ih n]
[jh]	voiced palato-a;veolar affricate	judge	[jh uh jh]
[l]	voiced alveolar lateral	liquid	[l ih k w ih d]
[w]	voiced bilabial approximant	water	[w ah dx axr]
[r]	voiced alveolar approximate	round	[r aw n d]
[y]	voiced velar approximant	year	[y iy r]
[h]	voiceless glottal fricative	happy	[h ae p iy]
[q]	voiceless glottal stop	kitten	[k ih q n]
[dx]	voiceless tap (allophone of /t/)	latter	[l ae dx er]

Table A.2: *The consonants of the ARPABET phonetic alphabet, with descriptions and examples. Based on http://www.billnet.org/phon/arpabet.php.*

Appendix B

Pronunciation Rules

Each rule is in the form of

$$\{\lambda_1, \ldots, \lambda_m\}\ \phi\ \{\rho_1, \ldots, \rho_n\} \Rightarrow \psi.$$

It states that phoneme ϕ with left context of $\lambda_1, \ldots,$ or λ_m and right context $\rho_1, \ldots,$ or ρ_m can be mapped to a regular expression of phones, ψ. Additionally, ψ has the capability to specify output (surface) context constraints. "<{}" and ">{}" are used for left and right output context constraints respectively. The rules apply to both within-word and cross-word phoneme sequences.

```
/* Define the input alphabet */
alphabet {
- _ aa ae ah ah_fp ao aw ax axr ay b bd ch d dd df dh dr eh el
en er ey f g gd hh ih ix iy jh k k- kd l m n ng nt ow oy p p-
pd r s sh t t- td tf th tq tr uh uw v w y z zh
};

/* Define the initial and final phones */
/* initial {-}; */
/* final {-}; */

/* Define special rule connection symbols. */
connect axr$ $axr;
connect ax$ $ax;
connect en$ $en;
connect syl$ $syl;
connect dx$ $dx;
```

```
/* Define the symbols not taking part in rules. */
ignore {#};

/* Define some broad phonetic classes */
DSTOP = {bd dd gd pd td kd tq};
STOP = {b d g p t k p- t- k- tr dr tf df ch jh};
VOWEL = {aa ae ah ah_fp ao aw ax axr ay eh
         el er ey ih ix iy ow oy uh uw};
VOWEL_NO_R = {aa ae ah ah_fp ao aw ax ay eh
              el ey ih ix iy ow oy uh uw};
VOWEL_NO_Y = {aa ae ah ah_fp ao aw ax axr eh
              el er ih ix ow uh uw};
VOWEL_NO_W = {aa ae ah ah_fp ao ax axr ay eh
              el er ey ih ix iy oy uh};
SEMIVOWEL = {l y w r};
NASAL = {n en m ng};
FRIC = {f th s sh v dh z zh};
// Alveolar & Dental sounds
ALVEOLAR = {en n t t- td tf tr d dd df dr s z th dh};
PALATAL = {sh ch zh jh}; // Palatal sounds
AFFRIC = {ch jh};

/*****************************************************************/

/*** Rules for /_/ ***/
{} _ {} => _ ;

/*** Rules for /-/ ***/
{} - {} => - ;

/*** Rules for /aa/ ***/
{} aa {} => aa ;

/*** Rules for /ae/ ***/
{} ae {} => ae ;

/*** Rules for /ah/ ***/
{} ah {} => ah ;

/*** Rules for /ah_fp/ ***/
{} ah_fp {} => ah_fp ;

/*** Rules for /ao/ ***/
{} ao {} => ao ;
```

```
/*** Rules for /aw/ ***/
{} aw {df dd tf td} => aw [ w {dx}> ] ;
{} aw {VOWEL r l y hh} => aw [w] ;
{} aw {} => aw ;

/*** Rules for /ax/ ***/
{f v} ax {n} => ix | ax | syl$ ;
{s z sh zh th dh} ax {n} => ix | [epi] syl$ ;
{ALVEOLAR} ax {n} => ix | <{tcl dcl tq} syl$ ;
{r} ax {n} => ( ax | ix ) | $axr axr ;
{} ax {n} => ax | ix | <{pcl kcl bcl gcl} syl$ ;
{f v} ax {m} => ax | [epi] syl$ ;
{s z sh zh th dh} ax {m} => ix | ax | [epi] syl$ ;
{ALVEOLAR} ax {m} => ax | ix | <{tcl dcl tq} syl$ ;
{r} ax {m} => ax | $axr axr ;
{} ax {m} => ax | <{pcl kcl bcl gcl} syl$ ;
{ALVEOLAR} ax {l} => ax | ix | syl$ ;
{r} ax {ALVEOLAR PALATAL y} => ( ax | ix ) | $axr axr ;
{ALVEOLAR PALATAL} ax {r} => ax | ix | axr axr$ ;
{y iy ey ay oy} ax {r} => ax | ix | axr axr$ ;
{ALVEOLAR PALATAL} ax {ALVEOLAR PALATAL y} => ix ;
{ y iy ey ay oy} ax {ALVEOLAR PALATAL y} => ix ;
{ALVEOLAR PALATAL} ax {} => ax | ix ;
{y iy ey ay oy} ax {} => ax | ix ;
{r} ax {r} => ax | axr axr$ ;
{r} ax {l} => ax | $axr axr | syl$ ;
{} ax {l} => ax | syl$ ;
{} ax {r} => ax | axr axr$ ;
{r} ax {} => ax | $axr axr ;
{} ax {ALVEOLAR PALATAL y} => ax | ix ;
{} ax {} => ax ;

/*** Rules for /axr/ ***/
{} axr {df dd tf td} => axr [r {dx}>] ;
{} axr {VOWEL l y w hh} => axr [r];
{} axr {} => axr ;

/*** Rules for /ay/ ***/
{} ay {df dd tf td} => ay [y {dx}>] ;
{} ay {VOWEL r l w hh} => ay [y] ;
{} ay {} => ay ;

/*** Rules for /b/ ***/
{VOWEL SEMIVOWEL} b {VOWEL} => bcl [b] ;
```

```
{- _} b {} => b ;
{} b {} => ( bcl | <{m em pcl tcl kcl bcl dcl gcl} ) b;

/*** Rules for /bd/ ***/
{m} bd {} => b | bcl [b] ;
{} bd {} => bcl [b] ;

/*** Rules for /ch/ ***/
{- _} ch {} => ch ;
{} ch {} => ( tcl | <{pcl tcl kcl} ) ch ;

/*** Rules for /d/ ***/
{- _} d {sh zh y} => d | jh ;
{} d {sh zh y} => (dcl | <{n en pcl tcl kcl bcl dcl gcl}) ( d | jh );
{- _} d {} => d ;
{} d {} => ( dcl | <{n en pcl tcl kcl bcl dcl gcl} ) d;

/*** Rules for /dd/ ***/
{VOWEL} dd {VOWEL hh} => dcl [d] | dx ;
{SEMIVOWEL} dd {VOWEL hh} => dcl [d] | $dx dx ;
{VOWEL} dd {y} => dcl [d | jh ] | dx dx$ ;
{SEMIVOWEL} dd {y} => dcl [d | jh ] | $dx dx dx$ ;
{VOWEL} dd {r w l} => dcl [d] | dx dx$ ;
{SEMIVOWEL} dd {r w l} => dcl [d] | $dx dx dx$ ;
{} dd {en} => dcl ( d ax$ | en$ ) ;
{en n} dd {y sh zh} => [dcl] [d | jh] ;
{en n} dd {} => [dcl] [d] ;
{} dd {y sh zh} => ( dcl | <{pcl tcl kcl bcl dcl gcl} ) [ d | jh ];
{} dd {} => ( dcl | <{pcl tcl kcl bcl dcl gcl} ) [d] ;

/*** Rules for /df/ ***/
{VOWEL} df {VOWEL hh} => dcl d | dx;
{SEMIVOWEL} df {VOWEL hh} => dcl d | $dx dx;
{VOWEL} df {y} => dcl ( d | jh ) | dx dx$ ;
{SEMIVOWEL} df {y} => dcl ( d | jh ) | $dx dx dx$ ;
{VOWEL} df {w r l} => dcl d | dx dx$ ;
{SEMIVOWEL} df {w r l} => dcl d | $dx dx dx$ ;
{} df {} => dcl d ;

/*** Rules for /dh/ ***/
{dh} dh {} => dcl dh | [dh] ;
{DSTOP FRIC AFFRIC NASAL} dh {} => [dcl] dh ;
{} dh {} => dh ;

/*** Rules for /dr/ ***/
```

```
{n} dr {} => [dcl] dr ;
{- _} dr {} => dr ;
{} dr {} => ( dcl | <{pcl tcl kcl bcl dcl gcl} ) dr ;

/*** Rules for /eh/ ***/
{} eh {} => eh ;

/*** Rules for /el/ ***/
{} el {df dd tf td} => el [l {dx}>] ;
{} el {VOWEL r y w hh} => el [l];
{} el {} => el ;

/*** Rules for /en/ ***/
{tq td dd} en {} => [$en] en | [$ax] ix n ; // en | ax n
{} en {} => [$en] en | [$ax] ax n ;

/*** Rules for /er/ ***/
{} er {df dd tf td} => er [r {dx}>] ;
{} er {VOWEL l y w hh} => er [r];
{} er {} => er ;

/*** Rules for /ey/ ***/
{} ey {df dd tf td} => ey [y {dx}>] ;
{} ey {VOWEL r l w hh} => ey [y] ;
{} ey {} => ey ;

/*** Rules for /f/ ***/
{} f {} => f | <{f} ;

/*** Rules for /g/ ***/
{ng} g {} => [gcl] g ;
{- _} g {} => g ;
{} g {} => ( gcl | <{pcl tcl kcl bcl dcl gcl} ) g ;

/*** Rules for /gd/ ***/
{ng} gd {} => g | gcl [ g ] ;
{} gd {} => gcl [ g ] ;

/*** Rules for /hh/ ***/
{} hh {} => hh ;

/*** Rules for /ih/ ***/
{} ih {} => ih ;

/*** Rules for /ix/ ***/
```

```
{} ix {} => ix ;

/*** Rules for /iy/ ***/
{} iy {df dd tf td} => iy [y {dx}>] ;
{} iy {VOWEL r l w hh} => iy [y] ;
{} iy {} => iy ;

/*** Rules for /jh/ ***/
{n} jh {} => [dcl] jh ;
{- _} jh {} => jh ;
{} jh {} => dcl jh ;

/*** Rules for /k/ ***/
{- _} k {} => k ;
{} k {} => ( kcl | <{kcl pcl tcl} ) k ;

/*** Rules for /k-/ ***/
{} k- {} => kcl k- ;

/*** Rules for /kd/ ***/
{} kd {} => kcl [ k ] ;

/*** Rules for /l/ ***/
{df dd tf td} l {} => $dx ll | l ;
{ax} l {df dd tf td} => -l | $syl el [l dx$] | ll dx$ ;
{ax} l {VOWEL r y w hh} => $syl el [l] | ll ;
{ax} l {} => $syl el | -l ;
{VOWEL r} l {df dd tf td} => -l | ll dx$ ;
{VOWEL r} l {VOWEL r y w hh} => ll ;
{VOWEL r} l {} => -l;
{} l {} => l ;

/*** Rules for /m/ ***/
{m} m {} => [m] ;
{ax} m {} => $syl em | m ;
{} m {} => m ;

/*** Rules for /n/ ***/
{ax} n {ax ix} => $syl en | n | nx;
{VOWEL SEMIVOWEL} n {ax ix} => n | nx ;
{ax} n {gd g kd k} => $syl en | n | ng ;
{} n {gd g kd k} => n | ng ;
{en n} n {} => [n] ;
{ax} n {} => $syl en | n ;
{} n {} => n ;
```

```
/*** Rules for /ng/ ***/
{} ng {} => ng ;

/*** Rules for /nt/ ***/
{} nt {} => n [tcl t] ;

/*** Rules for /ow/ ***/
{} ow {df dd tf td} => ow [w {dx}>] ;
{} ow {VOWEL r l y hh} => ow [w] ;
{} ow {} => ow ;

/*** Rules for /oy/ ***/
{} oy {df dd tf td} => oy [y {dx}>] ;
{} oy {VOWEL r l w hh} => oy [y] ;
{} oy {} => oy ;

/*** Rules for /p/ ***/
{- _} p {} => p ;
{} p {} => ( pcl | <{pcl tcl kcl} ) p ;

/*** Rules for /p-/ ***/
{} p- {} => pcl p- ;

/*** Rules for /pd/ ***/
{} pd {} => pcl [ p ] ;

/*** Rules for /r/ ***/
{df dd tf td} r {ax} => $dx rr | axr$ | r ;
{df dd tf td} r {} => $dx rr | r;
{ax} r {ax} => rr | $axr [r] | axr$ ;
{ax} r {VOWEL y w l hh} => rr | $axr [r] ;
{ax} r {tf td df dd} => -r | rr dx$ | $axr ;
{ax} r {} => -r | $axr ;
{VOWEL_NO_R l} r {ax} => rr | axr$ ;
{VOWEL_NO_R l} r {VOWEL y w l hh} => rr;
{VOWEL_NO_R l} r {tf td df dd} => -r | rr dx$ ;
{VOWEL_NO_R l} r {} => -r ;
{th} r {ax} => axr [r] | r | axr$ ;
{th} r {} => [axr] r ;
{} r {ax} => r | axr$ ;
{} r {} => r ;

/*** Rules for /s/ ***/
{en n} s {en n m ng w r l el} => [epi | tcl [t]] s [epi] ;
```

```
{en n} s {sh zh} => [epi | tcl [t]] ( s | sh ) ;
{en n} s {y} => [epi | tcl [t]] ( s [epi] | sh ) ;
{en n} s {} => [epi | tcl [t]] s ;
{m ng l el} s {en n m ng w r l el} => [epi] s [epi] ;
{m ng l el} s {sh zh} => [epi] ( s | sh ) ;
{m ng l el} s {y} => [epi] ( s [epi] | sh ) ;
{m ng l el} s {} => [epi] s ;
{} s {en n m ng w y r l el} => ( s | <{s z} ) [epi] ;
{} s {sh zh} => s | sh | <{s z} ;
{} s {y} => ( s | <{s z} ) [epi] | sh;
{} s {} => s | <{s z} ;

/*** Rules for /sh/ ***/
{en n m ng l el} sh {en n m ng w r l el} => [epi] sh [epi] ;
{en n m ng l el} sh {} => [epi] sh ;
{} sh {en n m ng w r l el} => ( sh | <{sh ch} ) [epi] ;
{} sh {} => sh | <{sh ch};

/*** Rules for /t/ ***/
{s} t {ax ix} => [tcl t] ;
{- _} t {y} => ( t | ch ) ;
{- _} t {} => t ;
{} t {sh zh y} => ( tcl | <{pcl tcl kcl} ) ( t | ch ) ;
{} t {} => ( tcl | <{pcl tcl kcl} ) t ;

/*** Rules for /t-/ ***/
{s} t- {ax ix} => [tcl t-] ;
{} t- {} => tcl t- ;

/*** Rules for /td/ ***/
{VOWEL} td {VOWEL hh} => tcl [t] | tq | dx ;
{SEMIVOWEL} td {VOWEL hh} => tcl [t] | tq | $dx dx ;
{VOWEL} td {y sh zh} => tcl [ t | ch ] | tq | dx dx$ ;
{SEMIVOWEL} td {y sh zh} => tcl [ t | ch ] | tq | $dx dx dx$ ;
{VOWEL} td {r w l} => tcl [t] | tq | dx dx$ ;
{SEMIVOWEL} td {r w l} => tcl [t] | tq | $dx dx dx$;
{VOWEL SEMIVOWEL} td {} => tcl [t] | tq ;
{NASAL} td {sh zh y } => tcl [ t | ch ] | tq ;
{NASAL} td {en} => tq en$ | tcl (en$ | t ax$) ;
{NASAL} td {} => tcl [t] | tq ;
{s f} td {sh zh y} => tcl ( t | ch ) | [tcl] ;
{f s} td {} => [ tcl [t] ] ;
{} td {sh zh y} => ( tcl | <{pcl tcl kcl} ) [ t | ch ] ;
{} td {} => ( tcl | <{pcl tcl kcl} ) [t] ;
```

```
/*** Rules for /tf/ ***/
{VOWEL} tf {VOWEL hh} => tcl t | dx ;
{SEMIVOWEL} tf {VOWEL hh} => tcl t | $dx dx ;
{VOWEL} tf {y} => tcl ( t | ch ) | dx dx$ ;
{SEMIVOWEL} tf {y} => tcl ( t | ch ) | $dx dx dx$ ;
{VOWEL} tf {l r w} => tcl t | dx dx$ ;
{SEMIVOWEL} tf {l r w} => tcl t | $dx dx dx$ ;
{- _} tf {} => t ;
{} tf {} => ( tcl | <{pcl tcl kcl} ) t ;

/*** Rules for /th/ ***/
{th} th {} => tcl th | [th] ;
{DSTOP FRIC AFFRIC NASAL} th {} => [tcl] th;
{} th {} => th ;

/*** Rules for /tq/ ***/
{} tq {en} => tq en$ | tcl t ax$ ; // tq {en}> | tcl t {ix}>
{} tq {} => tq ;

/*** Rules for /tr/ ***/
{- _} tr {} => tr ;
{} tr {} => ( tcl | <{pcl tcl kcl} ) tr ;

/*** Rules for /uh/ ***/
{} uh {} => uh ;

/*** Rules for /uw/ ***/
{ALVEOLAR PALATAL y iy ey ay oy} uw {ALVEOLAR PALATAL y} => ux ;
{ALVEOLAR PALATAL y iy ey ay oy} uw {VOWEL r l y hh} => uw [w] | ux;
{ALVEOLAR PALATAL y iy ey ay oy} uw {} => uw | ux ;
{} uw {df dd tf td} => uw [w {dx}>] | ux ;
{} uw {ALVEOLAR PALATAL} => uw | ux ;
{} uw {y} => uw [w] | ux;
{} uw {VOWEL r l hh} => uw [w] ;
{} uw {} => uw ;

/*** Rules for /v/ ***/
{} v {} => v | <{v} ;

/*** Rules for /w/ ***/
{dd df td tf} w {} => w | $dx ww ;
{VOWEL_NO_W r l} w {} => ww ;
{uw} w {} => <{uw} w | <{ux} ww ;
{} w {} => w ;
```

```
/*** Rules for /y/ ***/
{df dd tf td t} y {} => y | $dx yy | <{ch jh} [y] ;
{VOWEL_NO_Y r l} y {} => yy ;
{} y {} => y ;

/*** Rules for /z/ ***/
{en n m ng l el} z {en n m ng w r l el} => [epi] z [epi] ;
{en n m ng l el} z {y sh zh} => [epi] ( z | zh ) ;
{en n m ng l el} z {} => [epi] z ;
{} z {en n m ng w r l el} => ( z | <{z} ) [epi] ;
{} z {y sh zh} => z | zh | <{z} ;
{} z {} => z | <{z} ;

/*** Rules for /zh/ ***/
{en n m ng l el} zh {en n m ng w r l el} => [epi] zh [epi] ;
{en n m ng l el} zh {} => [epi] zh ;
{} zh {en n m ng w r l el} => ( zh | <{zh} ) [epi] ;
{} zh {} => zh | <{zh};
```

Bibliography

[1] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximizzation technique occurring in teh statistical analysis of probabilistic function of markov chains. *Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[2] J. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report ICSI-TR-97-021, University of Berkeley, 1997.

[3] H. Bourlard and S. Dupont. A new ASR approach based on independent processing and recombination of partial frequency bands. In *Proc. Intl. Conf. on Spoken Language Processing*, volume 1, pages 426–429, Philadelphia, PA, October 1996.

[4] H. Bourlard, S. Dupont, and C. Ris. Multi-stream speech recognition. Technical Report IDIAP-RR 96-07, IDIAP, Martigny, 1996.

[5] Ö. Çetin. *Multi-rate Modeling, Model Inference, and Estimation for Statistical Clusters.* PhD thesis, University of Washington, Seattle, Washington, 2004.

[6] J. W. Chang. *Near-Miss Modeling: A Segment-Based Approach to Speech Recognition.* Ph. D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, June 1998.

[7] J. W. Chang and J. R. Glass. Segmentation and modeling in segment-based recognition. In *Proc. Eurospeech*, pages 1199–1202, Rhodes, Greece, September 1997.

[8] C. Chibelushi, F. Deravi, and J. Mason. A review of speech-based bimodal recognition. *IEEE Trans. on Multimedia*, 4(1):23–37, March 2002.

[9] P. Clarkson and R. Rosenfeld. Statistical language modeling using the cmu-cambridge toolkit. In *Proc. Eurospeech*, pages 2707–2710, Rhodes, Greece, September 1997.

[10] M.H. Cohen. *Phonological Structures for Speech Recognition.* PhD thesis, Computer Science Division, University of California at Berkeley, April 1989.

[11] M. Collins. *Head-Driven Statistical Models for Natural Language Parsing.* PhD thesis, University of Pennsylvania, 1999.

[12] M. P. Cooke, A. C. Morris, and P. D. Green. Missing data techniques for robust speech recognition. In *Proc. ICASSP*, pages 863–866, Munich, Germany, April 1997.

[13] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continously spoken sentences. *IEEE Trans. Acoustics, Speech and Signal Processing*, 28(4):357–366, August 1980.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, June 1977.

[15] V. Digilakis. *Segment-based stochastic models of spectral dynamics for continuous speech recognition.* PhD thesis, Boston University, January 1992.

[16] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, Chichester, Brisbane, Toronto, Singapore, 1973.

[17] S. Dupont and H. Bourlard. Using multiple time scales in a multi-stream speech recognition system. In *Proc. Eurospeech*, pages 3–6, Rhodes, Greece, September 1997.

[18] S. Dupont and J. Luettin. Using the multi-stream approach for continuous audio-visual speech recognition: Experiments on the M2VTS database. In *Proc. ICSLP*, volume 4, pages 1283–1286, November 1998.

[19] S. Dupont and J. Luettin. Audio-visual speech modeling for continuous speech recognition. *IEEE transactions on multimedia*, 2(3):1520–9210, 2000.

[20] J. Eisner. Expectation semirings: Flexible EM for learning finite-state transducers. In *Proc. of the ESSLLI Workshop on Finite-State Methods in NLP*, Helsinki, August 2001.

[21] J. Eisner. Parameter estimation for probabilistic finite-state transducers. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, Philadelphia, July 2002.

[22] J. Fiscus. A post-processing system to yield reduced word error rates: Recogniser output voting error reduction (ROVER). In *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 347–352, Santa Barbara, CA, 1997.

[23] G. D. Forney. The Viterbi algorithm. *Proc. IEEE*, 61:268–278, March 1973.

[24] S. Furui. Speaker independent isolated word recognition using dynamic features of speech spectrum. In *IEEE Trans. Acoustics, Speech and Signal Processing*, pages 52–59, 1986.

[25] E. P. Giachin, A. E. Rosenberg, and C. H. Lee. Word juncture modeling using phonological rules for HMM-based continuous speech recognition. *Computer Speech and Language*, 5(2):155–168, April 1991.

[26] H. Gish and K. Ng. Parametric trajectory models for speech recognition. In *Proc. ICSLP*, volume 1, pages 466–469, Philadelphia, PA, USA, October 1996.

[27] J. R. Glass. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17(2):137–152, 2003.

[28] G. Gravier, G. Potamianos, and C. Neti. Asynchrony modeling for audiovisual speech recognition. In *Human Language Technology Conference*, pages 1–6, San Diego, USA, March 2002.

[29] T. J. Hazen. Visual model structures and synchrony constraints for audio-visual speech recognition. *IEEE Trans. Acoustics and Audio Processing*, May 2006. to be published.

[30] T. J. Hazen, I. L. Hetherington, H. Shu, and K. Livescu. Pronunciation modeling using a finite-state transducer representation. *Speech Communication*, 46(2):189–203, June 2002.

[31] T. J. Hazen, K. Saenko, C. La, and J. R. Glass. A segment-based audio-visual speech recognizer: Data collection, development and initial experiments. In *Proc. ICMI*, State College, Pennsylvania, October 2004.

[32] I. L. Hetherington. An efficient implementation of phonological rules using finite-state transducers. In *Proc. Eurospeech*, pages 1599–1602, Aalborg, September 2001.

[33] I. L. Hetherington. The MIT finite-state transducer toolkit for speech and language processing. In *Interspeech*, pages 2609–2612, Jeju Island, Korea, October 2004.

[34] X. D. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[35] M. Johnston and S. Bangalore. Finite-state multimodal parsing and understanding. In *Proceedings of the 18th conference on Computational linguistics*, pages 369–375, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

[36] B. H. Juang and L. R. Rabiner. The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-38(4):1639–1641, 1990.

[37] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[38] G. A. Kiraz. Multitiered nonlinear morphology using multitape finite automata: a case study on syriac and arabic. *Computational Linguistics*, 26(1):77–105, 2000.

[39] K. Kirchhoff, G. A. Fink, and G. Sagerer. Combining acoustic and articulatory feature information for robust speech recognition. *Speech Communication*, 37:303–319, 2000.

[40] F. Kubala, J. Bellegarda, J. Cohen, D. Pallett, D. Paul, M. Phillips, R. Rajasekaran, F. Richardson, M. Riley, R. Rosenfeld, B. Roth, and M. Weintraub. The hub and spoke paradigm for CSR evaluation. In *Proc. ARPA Human Language Technology Workshop*, pages 37–42, Princeton, NJ, USA, March 1994.

[41] W. Kuich and A. Salomaa. *Semirings, Automata, Languages.* Springer-Verlag, 1986.

[42] L. Lamel, J. Gauvain, and G. Adda. Lightly supervised acoustic model training. In *Proc. Automatic Speech Recognition workshop*, volume 1, pages 150–154, Paris, France, September 2000.

[43] S. Lauritzen. *Graphical Models.* Oxford University Press, Oxford, UK, 1996.

[44] R. P. Lippmann. Speech recognition by machines and humans. *Speech Communication*, 22:1–15, July 1997.

[45] K. Livescu. *Feature-Based Pronunciation Modeling for Automatic Speech Recognition.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, August 2005.

[46] K. Livescu and J. R. Glass. Segment-based recognition on the PhoneBook task: Initial results and observations on duration modeling. In *Proc. Eurospeech*, pages 1437–1440, Aalborg, Denmark, September 2001.

[47] K. Livescu and J. R. Glass. Feature-based pronunciation modeling with trainable synchrony probabilities. In *Proc. ICSLP*, Jeju, South Korea, October 2004.

[48] A. Ljolje, J. Hirschberg, and J. P. H. Van Santen. Automatic speech segmentation for concatenative inventory selection. In J. P. H. Van Santen, editor, *Progress in Speech Synthesis*, pages 305–311. Springer, 1997.

[49] E. McDermott. *Discriminative Training for Speech Recognition.* PhD thesis, Waseda University, 1997.

[50] M. Meteer and J.R. Rohlicek. Statistical language modeling combining n-gram and context-free grammars. In *Proc. ICASSP*, pages II–37 – II–40, Minneapolis, MN, April 1993.

[51] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[52] J. Odell. *The Use of Context in Large Vocabulary Speech Recognition.* PhD thesis, Cambridge University, 1995.

[53] M. Ostendorf, V. Digilakis, and O. Kimball. From HMM's to segment models: a unified view of stochastic modelling for speech recognition. *IEEE Trans. Speech and Audio Processing*, 4(5):360–378, 1996.

[54] J. F. Pitrelli, C. Fong, S. H. Wong, J. R. Spitz, and H. C. Leung. Phonebook: A phonetically-rich isolated-word telephone-speech database. In *Proc. ICASSP*, pages 101–104, Detroit, MI, May 1995.

[55] G. Potamianos, C. Neti, G. Gravier, A. Garg, and A. Senior. Recent advances in the automatic recognition of audiovisual speech. *Proceedings of the IEEE*, 91(9):1306–1326, September 2003.

[56] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77:257–286, 1989.

[57] L. R. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition.* PTR Prentice Hall, Englewood Cliffs, NJ, 1993.

[58] K. Saenko, K. Livescu, K. Schutte, J. R. Glass, and T. Darrell. Audio-visual speech recognition with streams of articulatory features. In *Advances in Neural Information Processing Systems*, 2005.

[59] T. Sainath. Acoustic landmark detection and segmentation using the Mcaulay-Quatieri sinusoidal model. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, August 2005.

[60] T. Sainath and T. J. Hazen. A sinusoidal model approach to acoustic landmark detection and segmentation for robust segment-based speech recognition. In *Proc. ICASSP*, Toulouse, France, May 2006.

[61] S. Sakai and H. Shu. A probabilistic approach to unit selection for corpus-based speech synthesis. In *Interspeech*, pages 81–84, Lisbon, Portugal, September 2005.

[62] S. Seneff. Comments on "Towards increasing speech recognition error rates" by H. Bourlard, H. Hermansky, and N. Morgan. *Speech Communication*, 18:253–255, May 1996.

[63] S. Seneff. The use of linguistic hierarchies in speech understanding. In *Proc. Intl. Conf. on Spoken Language Processing*, Sydney, August 1998.

[64] H. Shu and I. Lee Hetherington. EM training of finite-state transducers and its application to pronunciation modeling. In *Proc. Intl. Conf. on Spoken Language Processing*, pages 1293–1296, Denver, CO, USA, September 2002.

[65] K. N. Stevens. Applying phonetic knowledge to lexical access. In *Proc. Eurospeech*, pages 3–11, Madrid, Spain, September 1995.

[66] A. Stolcke. SRILM - an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, Denver, CO, USA, September 2002.

[67] N. Ström, I. L. Hetherington, T. J. Hazen, E. Sandness, and J. R. Glass. Acoustic modeling improvements in a segment-based speech recognizer. In *IEEE Automatic Speech Recognition and Understanding Workshop*, pages 139–142, Snowbird, December 1999.

[68] M. Tang, S. Seneff, and V. W. Zue. Modeling linguistic features in speech recognition. In *Proc. Eurospeech*, Geneva, Switzerland, September 2003.

[69] D. T. Toledano, L. A. Hernández Gómez, and L. V. Grande. Automatic phonetic segmentation. *IEEE Trans. Speech and Audio Processing*, 11(6):617–625, November 2003.

[70] V. Valtchev, J. Odell, P. Woodland, and S. Young. MMIE training of large vocabulary recognition systems. *Speech Communication*, 22:303–314, September 1997.

[71] Y. Wang, M. Mahajan, and X. Huang. A unified context-free grammar and n-gram model for spoken language processing. In *Proc. ICASSP*, pages 1639–1642, Istanbul, Turkey, June 2000.

[72] S. Young, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book*. Cambridge University, Cambridge, UK, 1997.

[73] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen, and I. L. Hetherington. JUPITER: A telephone-based conversational interface for weather information. *IEEE Trans. on Speech and Audio Processing*, 8(1):100–112, 2000.