Incorporating Pitch Features for Tone Modeling in Automatic Recognition of Mandarin Chinese

by

Karen Lingyun Chu

S.B. EECS, M.I.T., 2008

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

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

Author ...... [illegible] ....
Department of Electrical Engineering and Computer Science
August 21, 2009

Certified by ....
Wade Shen
VI-A Company Thesis Supervisor

Certified by ....
Robert C. Berwick
Professor
M.I.T. Thesis Supervisor

Accepted by .... [illegible] ....
Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Incorporating Pitch Features for Tone Modeling in Automatic Recognition of Mandarin Chinese

by

Karen Lingyun Chu

Submitted to the Department of Electrical Engineering and Computer Science
on August 21, 2009, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Tone plays a fundamental role in Mandarin Chinese, as it plays a lexical role in determining the meanings of words in spoken Mandarin. For example, these two sentences 我爱马(I like horses) and 我爱骂(I like to scold) differ only in the tone carried by the last syllable. Thus, the inclusion of tone-related information through analysis of pitch data should improve the performance of automatic speech recognition (ASR) systems on Mandarin Chinese.

The focus of this thesis is to improve the performance of a non-tonal automatic speech recognition (ASR) system on a Mandarin Chinese corpus by implementing modifications to the system code to incorporate pitch features. We compile and format a Mandarin Chinese broadcast new corpus for use with the ASR system, and implement a pitch feature extraction algorithm. Additionally, we investigate two algorithms for incorporating pitch features in Mandarin Chinese speech recognition. Firstly, we build and test a baseline tonal ASR system with embedded tone modeling by concatenating the cepstral and pitch feature vectors for use as the input to our phonetic model (a Hidden Markov Model, or HMM). We find that our embedded tone modeling algorithm does improve performance on Mandarin Chinese, showing that including tonal information is in fact contributive for Mandarin Chinese speech recognition. Secondly, we implement and test the effectiveness of HMM-based multi-stream models.

VI-A Company Thesis Supervisor: Wade Shen

M.I.T. Thesis Supervisor: Robert C. Berwick
Title: Professor

Acknowledgments

I would like to thank first and foremost Wade Shen who supervised the work done in this thesis. Without his invaluable guidance and support, this thesis would not have been possible. I would also like to thank MIT Lincoln Laboratory for the opportunity to perform my thesis research as an intern through the MIT VI-A Program. Their support of this wonderful program is greatly appreciated.

I would also like to express my appreciation for Professor Robert Berwick for being a wonderfully understanding advisor during the past four years. His good advice, both academic and otherwise, was a valuable addition to my experience at MIT.

I want to thank my parents, Chao-Hsien and Fang-Mei Chu, for their unwavering support. The sacrifices they have made in order to finance my years at MIT is very much appreciated, as are their constant words of encouragement. Thank you for allowing me to explore my interests and discover who I am as both a scholar and a person.

I would also like to thank all my friends who have supported me during my time at MIT. My experience would not have been the same without them, and although they are too many to list individually, I would like them all to know how much I value their friendship.

And lastly, I would like to thank Flopper. His constant availability for hugs and his sage though irrelevant advice helped me through many trying times. Baroo.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mandarin Chinese is one of many languages considered to be a tonal language due to the presence of lexical tone [13]. Lexical tone is a distinctive pitch pattern carried by a syllable of a word that is essential to the meaning of the word. In nontonal languages, intonation may be used to communicate higher-level meanings associated with emotion, but do not change the base meaning of words. Words have the same meaning regardless of any tone used by the speaker of those words. In contrast, the meanings of spoken words in tonal languages is heavily dependent on the tones carried each syllable. Changing the tone on just one syllable may change the entire meaning of the sentence. For example, the Mandarin Chinese sentences 我爱马(I like horses) and 我爱骂 (I like to scold) differ only in the tone of the last syllable. Because of the fundamentally important role tone plays in Mandarin Chinese, the inclusion of tone-related information should improve the performance of automatic speech recognition (ASR) systems on Mandarin Chinese.

There have been many other groups pursuing the incorporation of tone into speech recognition for Mandarin Chinese. Some have focused solely on tone recognition, such as in [31]. However, most previous work has taken a two-step approach of implementing tone recognition as separate from traditional speech recognition, followed by incorporating the results of the tone recognizer to augment the results of a traditional ASR system [23, 34, 25, 24, 18]. Other work, including [26, 33, 12, 11] incorporate the tonal aspect as part of the overall ASR process, without a separate step for tone recognition.

It is generally agreed that pitch is a good indicator of tone, and thus tone-related features are comprised of information derived from the pitch track. For the most part, pitch features are extracted in a similar fashion by most researchers. The pitch track is extracted via a speech processing software, then interpolated and smoothed in some manner. However, [24] uses a tone feature extraction approach based on multi-layer perceptrons that provides positive results for improving the performance of their recognition system. A slightly different method is also used by [34], as they use both a tone model and a duration model to gain accuracy in their word recognition experiments.

There is more variation in the method chosen for incorporating the pitch features in the recognition system. For systems based on Hidden Markov Models (HMMs) for phonetic modeling, when tone recognition is explicitly performed, the results are generally incorporated by using the recognized tones to augment the lattices output by the phone recognizer [23]. However, embedded tone modeling is also used, in which case the extracted pitch features are appended to the cepstral feature vector used in phone recognition [24, 11]. Lei and colleagues in fact use a combination of explicit and embedded tone modeling to obtain further improvement than either used alone [26].

A subsyllabic modeling approach is used by [18, 12, 11] in which phone recognition was performed not on entire syllables, but on separate initials and finals. They chose to associate only the syllable finals with a tone label, leaving the syllable initials without tone. While [18] chose to recognize tones separately and use the results in combination with a non-tonal syllable recognizer, [12, 11] perform joint recognition of tones and syllables in their systems.

A multi-stream approach is used successfully by Mak and Tam [29] in which they use HMM composition to recombine sub-band HMMs at the state level. They find that a 1-state asynchrony in sub-bands gives improvement over synchronous recombination, although further asynchrony results in worse performance.

Outside of HMM based recognition systems, some interesting work taking a multi-stream approach has also been done with Dynamic Bayesian Networks. Livescu and colleagues [27] demonstrate that multiple linguistic feature streams can be used successfully in speech recognition, and show that asynchrony between the streams can lead to improvements in performance. Lei and colleagues [25] propose to model the cepstral features and the pitch features in separate streams rather than treating the two as a single combined input stream. With this approach, they also explored the effect of asynchrony between the two streams and find that the use of a multi-stream model results in significant improvements over single-stream recognizers.

1.1 Thesis Outline

The focus of this thesis is to improve the performance of a typical non-tonal ASR system on a Mandarin Chinese corpus by implementing modifications to the system code to incorporate pitch features. This work makes a number of significant modifications to the existing ASR system as well as the Mandarin Chinese broadcast new corpus for use with the ASR system. In addition to building a baseline tonal ASR system with embedded tone modeling, a main focus of this work is in implementing and testing the effectiveness of an HMM-based multi-stream model.

The major contributions of this paper are:

1. Compiling and formatting a Mandarin Chinese corpus for use with our ASR system, and enabling Unicode UTF-8 and GB2312 character encodings as input and output formats for our system.

2. Implementing a pitch feature extraction algorithm resulting in features compatible with our HTK-based phonetic models.

3. Building and testing an ASR system with embedded tone modeling based on the structure and codebase of the original ASR system.

4. Building and testing an ASR system with HMM-based multi-stream models based on the structure and codebase of the original ASR system.

The rest of the thesis is as follows. In Chapter 2, we present a brief introduction to tones in Mandarin Chinese and an overview of our ASR system. Chapter 3 discusses the contribution of this work, and presents the results of our experiments alongside a description of our implementations. Finally, we summarize our work and findings in Chapter 4 and discuss future extensions of the work presented in this thesis.

Chapter 2

Background

This chapter provides an introduction to tones in Mandarin Chinese, as well as an overview of the existing ASR system.

2.1 Tones in Mandarin Chinese

Mandarin Chinese is often described as having five tones differentiated by distinct pitch patterns. As described in [13], they are: Tone 1 - high level tone, Tone 2 - rising tone, Tone 3 - low or dipping tone, Tone 4 - falling tone, and Tone 5 - neutral tone. Graphical representations of the Mandarin tones are presented in Figure 2-1. Tone 1, the high level tone, is characterized by a steady sound sustained at a pitch higher than the speaker's average pitch level. The second, or rising, tone is characterized by the pitch rising from medium to a high within the speaker's pitch range, similar to question intonation in English. The third tone, low or dipping, is more amorphous than the first two, but in general can be characterized as a low pitch accompanied by a subsequent rise to a mid-level pitch. When it occurs without the accompanying rise, it is called a half-third tone. Tone 4, the falling tone, is characterized by a sharp drop in pitch from high to low. This tone is similar to the intonation used on curt assertions or commands in English. The fifth tone is the most difficult to characterize, because its pitch is almost entirely dependent on the tone of the preceding syllable. It often occurs at the ends of words and phrases, and is usually carried by a short unstressed syllable. Linguistically, it may be considered the effect of the preceding tone expanding onto a subsequent untoned syllable.

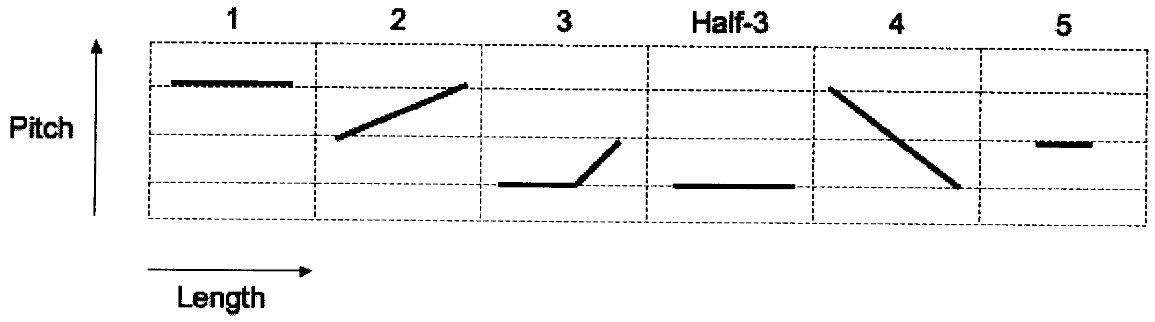


Figure 2-1: Tones in Mandarin Chinese.

There are a variety of conventions for transcribing tones, including the use of diacritical marks in both pinyin and IPA, as well as simple numerals referring to the tones. This paper will represent toned phonemes by simply appending the number of the tone to the end of the phoneme transcription. For example,*wo3* would represent the phoneme *wo* carrying a third tone.

All five tones can be carried by any phoneme within the Mandarin dialect, but must also follow some additional rules governing specific sequences of tones, known as tone sandhi rules [13]. Thus, spoken words will sometimes vary from their defined tone sequences due to the effects of tone sandhi. Examples can be found in Table 2.1.

Rule 1. When there are two adjacent third tones, the first syllable becomes a second tone and the second syllable becomes a half-third tone.

Rule 2. When there are three adjacent third tones:

i. If the first word is two syllables followed by a one-syllable word, the first two syllables become second tones, and the last remains unchanged.

ii. If the first word has one syllable followed by a two-syllable word, the first syllable becomes a half-third tone, the second syllable becomes a second tone, and the last syllable remains unchanged.

Rule 3. When a third tone is followed by any other tone, it usually becomes a half-third tone.

Rule 1	lao3shu3	lao2shu3
Rule 2.i	lao3shu3 pao3	lao2shu2 pao3
Rule 2.ii	xiao3 lao3shu3	xiao3 lao2shu3

Table 2.1: Tone sandhi examples

2.2 ASR System Overview

The existing automatic speech recognition (ASR) system is one optimized for non-tonal languages, such as English, and makes use of signal processing and modeling technologies shared by ASR systems at many institutions, both commercial and academic. The system features a front-end feature extractor that uses perceptual linear prediction (PLP) to extract feature vectors from the raw input speech audio. Using the extracted feature vectors, we then train a phonetic model to represent the speech sounds. The phonetic model in our system is a Hidden Markov Model (HMM) based on the Hidden Markov Model Toolkit (HTK) from Cambridge University [36]. In addition, we train an n-gram language model, intended to capture higher-level language structures such as word patterns within a language. Once these models are trained, the system can then use them to output transcriptions of other speech recordings through a decoding process.

2.2.1 PLP Feature Extraction

Perceptual linear prediction (PLP) is one of several feature extraction methods used in speech processing applications, along with mel-cepstral analysis. These methods are good for speech processing applications due to their simulation of physical aspects of human hearing, as well as their speaker independence and general robustness [16].

The PLP front-end processor in our ASR system consists of the following basic steps, as outlined in 1990 by Hermansky [17]:

1. Frame Blocking: In this step, the signal $s(n)$ from the audio file is blocked into N-sample frames with adjacent frames separated by M samples. Take $M = (1/3)N$ as illustrated in Figure 2-2. The first frame extends for the first N samples. The second frame begins M samples after the first frame, overlapping by $N - M$ samples. The third frame begins $2M$ samples after the first frame and overlaps it by $N - 2M$ samples. If $M \leq N$, then adjacent frames will overlap by $N - M$ samples, with overlapping samples correlating from frame to frame. Thus, for $M \ll N$, the results of PLP will be smooth, due to a large amount of overlapping. However, if $M > N$, there will be no overlap, with some speech signal not appearing in any of the frames, resulting in noise whose magnitude increases with M. Because this situation is detrimental in speech recognition, ASR systems will use only parameters for which $M \leq N$. If we let $x_l(n)$ represent the l^{th} frame of audio data, and there are L frames total, then

$$x_l(n) = s(Ml + n), n = 0, 1, ..., N - 1, l = 0, 1, ..., L - 1. \tag{2.1}$$

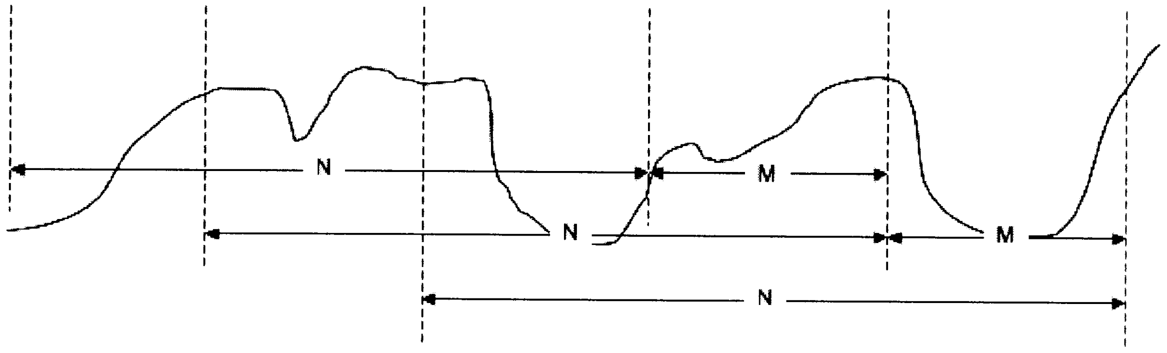


Figure 2-2: Frame blocking illustrated with $M = (1/3)N$.

2. Windowing: This step applies a window to each frame to minimize discontinuities at the beginning and end of the frame. The goal is to taper the signal to zero at the beginning and end of each frame. A typically used window for this step is the Hamming window of the form

$$w(n) = 0.54 - 0.46 \cos \frac{2\pi n}{N - 1}, 0 \leq n \leq N - 1. \tag{2.2}$$

After applying windowing, the signal becomes

$$\tilde{x}_l(n) = x_l(n)w(n), 0 \leq n \leq N-1. \tag{2.3}$$

3. Power Spectrum Calculation: The windowed speech is subsequently transformed into the spectral domain using a fast Fourier transform (FFT), resulting in the speech spectrum $X(\omega)$. The power spectrum is then calculated by computing the squared magnitude of the speech spectrum:

$$P(\omega) = \Re[X(\omega)]^2 + \Im[X(\omega)]^2 \tag{2.4}$$

where $\Re$ is the real part of the speech spectrum, and $\Im$ the imaginary part.

4. Critical-band Integration and Resampling: In order to reduce the spectral resolution of the signal, particularly at high frequencies, to reflect the lower resolution of human hearing at high frequencies, the power spectrum $P(\omega)$ is then warped along the frequency axis ω into the Bark frequency Ω by

$$\Omega(\omega) = 6\ln\left\{\frac{\omega}{1200\pi} + \left[\left(\frac{\omega}{1200\pi}\right)^2 + 1\right]^{0.5}\right\} \tag{2.5}$$

where ω is the angular frequency in radians/second. The warped power spectrum is then convolved with a simulated critical-band masking curve $\Psi(\Omega)$ from Fletcher [15]. This simulated critical-band masking curve crudely approximates what is known about the shape of auditory filters. The convolution of $\Psi(\Omega)$ with $P(\omega)$ results in samples of the critical-band power spectrum

$$\Theta(\Omega_i) = \sum_{\Omega=-1.3}^{2.5} P(\Omega - \Omega_i)\Psi(\Omega) \tag{2.6}$$

The signal is then resampled at approximately 1-Bark intervals.

5. Pre-emphasis: The integrated and downsampled power spectrum $\Theta[\Omega(\omega)]$ is then pre-emphasized to approximate the unequal sensitivity of human hearing

at different frequencies

$$\Xi[\Omega(\omega)] = E(\omega)\Theta[\Omega(\omega)] \tag{2.7}$$

This is implemented as an explicit weighting of elements in the critical band spectrum, where $E(\omega)$ is defined as

$$E(\omega) = \frac{(\omega^2 + 56.8e6)\omega^4}{(\omega^2 + 6.3e6)^2(\omega^2 + 0.38e9)} \tag{2.8}$$

6. Power Law of Hearing: The spectral amplitudes are then compressed to reflect the power law of hearing. In human hearing, the perceived loudness of a signal is related to the cube root of its intensity. Thus, we take the cube root of the signal

   $$\Phi(\Omega) = \Xi(\Omega)^{0.33} \tag{2.9}$$

   In addition to simulating human hearing, this step also reduces amplitude variations in the critical band spectrum to allow for all-pole modeling by a relatively low model order.

7. Autoregressive Modeling: An autoregressive model is then used to smooth the compressed critical band spectrum. We first calculate the inverse discrete Fourier transform (DFT) of our compressed spectrum, resulting in an autocorrelation function dual to $\Phi(\Omega)$. Using this autocorrelation function, we can solve a system of linear equations using Durbin's method to derive the autoregressive model.

8. Conversion to Cepstral Coefficients: In this step, the autoregressive coefficients are then converted to cepstral coefficients, and liftering is performed to adjust the sensitivity of these features to the amplitude of resonance peaks in the spectrum.

9. Temporal Feature Derivative: The PLP representation of a signal provides a good representation of the spectral properties of the signal in the given frame. However, an improved representation can be obtained by including temporal

information using the first and second derivatives of the PLP features. A first or second order difference is an inappropriate approximation of the derivative, as it is very noisy. The derivative $\frac{\delta c_n(t)}{\delta t}$ can be better approximated by a least-squares estimate over a finite length window:

$$\frac{\delta c_n(t)}{\delta t} = \Delta c_n(t) \approx \mu \sum_{k=-K}^{K} k c_n(t+k) \tag{2.10}$$

where μ is a normalization constant and $(2K+1)$ is the number of frames over which the computation is performed. The final PLP feature vector consists of the cepstral coefficients, with an appended vector of the cepstral time derivatives, and if calculated, an appended vector of the second-order time derivatives $\Delta^2 c_n(t)$.

2.2.2 Phonetic Modeling with HMM

HMMs are a common component of contemporary speech recognition systems. The basic theory of HMMs was published by Baum and colleagues ([5, 6, 7, 8, 9]) during the late 1960s to early 1970s. They were implemented for speech processing by Baker [3] at CMU and by Jelinek et al. at IBM ([1, 2, 4, 19, 20, 21]) in the 1970s. In this section, we will provide an overview of HMMs in the context of speech recognition. For a more mathematically rigorous derivation, please consult [22] and [32].

Hidden Markov Models

An HMM is a finite state machine with a set of states $Q = (q_0 q_1 q_2 ... q_n)$, with a set of transition probabilities $A = a_{01} a_{02} ... a_{n1} ... a_{nn}$ representing the probability of transitioning from one state to another. Graphically, the states can be pictured as nodes, and the transition probabilities as directed edges between the nodes. An edge exists only if there is a nonzero transition probability between the two nodes, and a node may have multiple exiting edges as long as the total probability along all exiting edges sums to 1.0. Self-loops are also allowed for modeling variable state durations.

The observation sequence that is used as input for the HMM is $O = (o_1 o_2 o_3 ... o_t)$. The aspect of an HMM that makes it "hidden" is that the observations are probabilistic functions of the states. Therefore, an HMM also has a set of observation likelihoods $B = b_i(o_t)$ representing the probability of an observation o_t being generated by state i. Additionally, our system specifies special start and end states that do not output any observations. A general HMM is illustrated in 2-3.

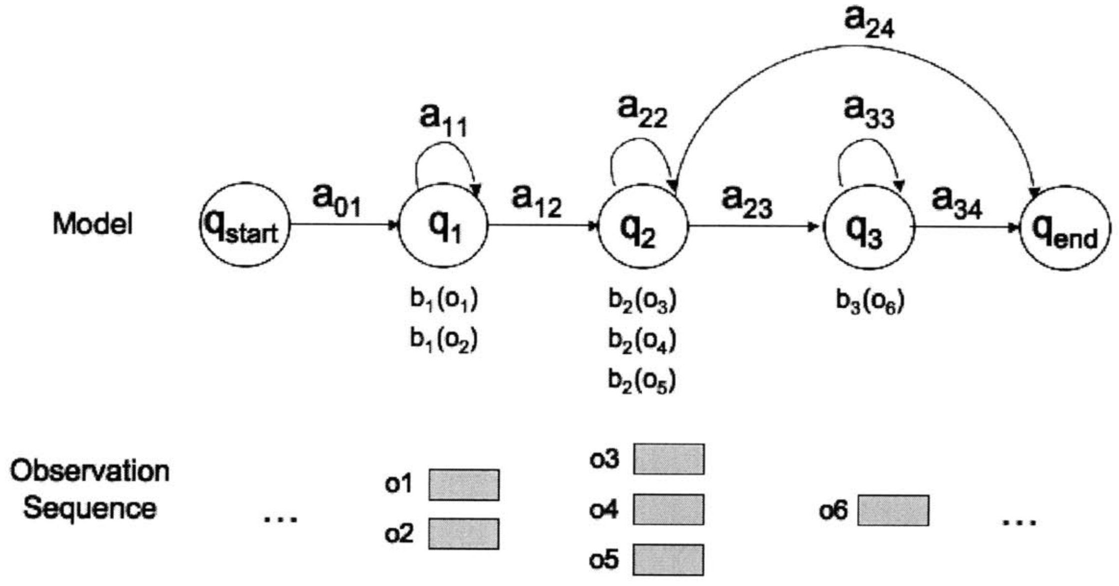


Figure 2-3: An HMM, illustrating states, transition probabilities, an observation sequence, and observation likelihoods.

In speech recognition, the observation sequences consist of speech data. In our ASR system this is in the form of a set of PLP features for each frame of a frame-blocked speech signal. If we take as an example the isolated word recognition problem, each spoken word can be represented by an observation sequence O, and each individual observation o_t would represent the PLP feature vector associated with time t. The goal of isolated word recognition, then, is to calculate

$$\arg\max_i \{P(w_i|O)\} \tag{2.11}$$

where w_i is the i^{th} word in the vocabulary of the system.

Although this probability is not directly computable, using Bayes' Rule results in

$$P(w_i|O) = \frac{P(O|w_i)P(w_i)}{P(O)} \tag{2.12}$$

which means given a set of prior probabilities $P(w_i)$, calculating the most probable word depends only on $P(O|w_i)$. If the word is modeled by an HMM, the computation of $P(O|w_i)$ can be replaced by estimating the HMM model parameters.

Given a model M, the joint probability of observation sequence O being generated by M moving through a state sequence X is simply a product of the transition probabilities and the observation likelihoods:

$$P(O, X|M) = a_{12}b_2(o_1)a_{22}b_2(o_2)a_{23}b_3(o_3)... \tag{2.13}$$

However, in an HMM, the sequence of states X is hidden. Therefore, the likelihood $P(O|M)$ can be computed by summing over all possible state sequences $X = x(1), x(2), x(3), ..., x(T)$

$$P(O|M) = \sum_X a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(o_t)a_{x(t)x(t+1)} \tag{2.14}$$

The Baum-Welch re-estimation method is then used to estimate the HMM parameters A and B, which additionally results in a value for $P(O|M)$. For isolated word recognition, we could therefore also use the Baum-Welch algorithm to find the model with the highest value of $P(O|M)$. However, this cannot be extended to the continuous speech recognition case, and therefore a separate decoding procedure as discussed in further detail in Section 2.2.4 is used.

Triphones, Model Clustering, and Tied State Models

A feature of our ASR system that is used throughout all experiments run in this study is the ability to train triphone phonetic models. Triphones model a phone in the context of the phone immediately preceding it (left context) and the phone immediately after it (right context). This allows our models to capture any pronunciation effects on a phone due to other phones near it. For example, vowels have a tendency to become nasalized when preceded by or followed by a nasal consonant. Basic triphone models are created by cloning the set of monophone models and reestimating their parameter values based on triphone transcriptions. The resulting models can then be made more robust through model clustering and state tying, as described in [35].

The idea behind model clustering is to use more than a single HMM to characterize the input data. This allows the model to become more robust to noise in the production of the input, such as accents or different speeds of talking by separating inhomogeneous variabilities. Separating inconsistent training data to create more homogeneous subgroups can decrease the complexity of the models, and increase accuracy by making use of context dependency. For example, we may consider grouping phones into triphone units. In English, there are on the order of 10,000 such units, with many of them functionally identical. The problem to solve is determining which units can be merged with low enough loss of information to not negatively impact the system's performance. There are a number of general clustering algorithms suitable for this purpose, including k-means clustering, generalized Lloyd algorithm, the greedy growing algorithm used in set partition or decision trees. Our ASR system performs clustering using a decision tree algorithm.

Additional robustness can be obtained through state tying. Because speech is very complex to model with numerous phones, and in the case of Mandarin Chinese, variations of those phones with different associated tones, the number of parameters in the model becomes large very quickly. A variation of HMM structure that is used in our ASR system to reduce the number of independent parameters and thus make parameter estimation simpler and more reliable is the idea of tying. The basic concept is to set up equivalence relation between HMM parameters in different states. This can be done when the observation probability density is known to be the same in two or more states. This can be especially effective when there is insufficient training data to reliably estimate a large number of parameters.

2.2.3 n-Gram Language Models

The purpose of training a language model in ASR is to provide higher-level linguistic information on top of that captured in the phonetic models. Especially in the case of tonal languages where the meaning of a word can be changed drastically by a difference in tone carried by the exact same phoneme, using a language model can improve the performance of the recognition system. An n-gram language model is a probabilistic model of word sequences in a language. They can be used to compute probabilities of entire sentences, or for providing a probabilistic prediction of a specific word in a sequence.

The simplest possible language model would simply allow any word to fall in any spot of any sequence. Essentially, there would be no constraints on word ordering, and all words would have equal probability of following any other word. A slightly more complex model would still allow free ordering on words, but the probability of a word following any other word would be related to its frequency of occurrence. For example, in English, a word like *the* has a higher frequency than a word such as *blanket*. Therefore, in this model, both *the* and *blanket* would be allowed to occur after any other word, but *the* would have a higher probability of doing so than *blanket*.

Both of those models are clearly not good models for actually capturing word patterns in a language. A more reasonable concept would be to look at the conditional probability of a word given the previous words, rather than purely at individual frequencies of words. Suppose you have the sequence *"Alice sat on the"*. The probability of seeing *blanket* following that sequence is higher than the probability of *blanket* occurring based purely on its relative frequency in the language. We can express this idea mathematically by representing a complete string of correctly-placed words as w_1^k and calculating

$$P(w_1^k) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)...P(w_k|w_1^{k-1}) \tag{2.15}$$

$$= \prod_{i=1}^{k} P(w_i|w_1^{i-1}) \tag{2.16}$$

Because calculating probabilities like $P(w_k|w_1^{k-1})$ is difficult, we approximate this probability by using n-gram models with small values of n. A bigram model, for example, approximates that probability by the conditional probability of the previous word. Similarly, an n-gram model approximates that probability by using conditional probabilities for the n previous words in the sequence. The general equation for an n-gram approximation is

$$P(w_k|w_1 k - 1) \approx P(w_k|w_{k-n+1}^{k-1}) \tag{2.17}$$

meaning that the probability of a word w_k given all previous words in the sequence can be approximated by the probability given only the previous n words, where n is more computationally tractable than k.

2.2.4 Decoding

The process of decoding calculates the sequence of states in an HMM that is most likely to have generated the observation sequence provided as input to the ASR system. Because word boundaries are not clearly delineated in continuous speech, decoding is much more difficult in this case than when the input is known to be a single word. While the individual phones may be very well modeled with the HMM, it is still necessary to segment, or find word boundaries, in the stream of phones. In our ASR system, this is done using n-gram language models and the Viterbi algorithm. Because the Viterbi algorithm is able to find the optimal state sequence given a set of observations and a single model, it is necessary to combine the phone models to form larger cross-phone and cross-word models. We do so by adding transition probabilities between words, which can be calculated using n-gram language models. Our ASR system only performs Viterbi decoding using bigrams in order to be more efficient. Therefore, our system does a first-pass decode using a bigram and the Viterbi algorithm, which generates a list of N-best sentences for a given input along with their likelihood score. We can then use a trigram or 4-gram language model to rescore the N-best sentences by assigning a new prior probability to each of these

N-best sentences, which can be combined with the bigram likelihood score to generate a posterior probability for each sentence.

Chapter 3

Tone Modeling with Pitch Features

As previously mentioned, the existence of lexical tone in Mandarin Chinese makes it a good candidate for applying pitch features in ASR. In this chapter, we discuss the complications involved in working with a Mandarin Chinese corpus, a procedure for extracting pitch features, and a basic method of incorporating pitch features in the phonetic models.

3.1 Working with a Mandarin Chinese Corpus

3.1.1 Building the Corpus

The training and evaluation data were drawn from two different collections of broadcast news speech available from the Linguistic Data Consortium. The training set consisted of thirty hours of speech data and transcripts from the 1997 HUB4 Mandarin broadcast news speech set [14]. For evaluation, we used Mandarin broadcast news data from DARPA EARS eval-04 dataset.

Both data sets included a collection of audio files, transcripts of the speech with individually tagged sentences and speakers, and a list of start and end times associated with each sentence. As a graphically represented language, Chinese is not usually written with spaces denoting separate words, as is the case in alphabet-based languages. Hence, the raw transcripts required a segmentation process to parse the sentences into words. Fortunately, this corpus had been previously used for machine translation applications, and the segmentation had already been performed.

Because the ASR system requires very specific formatting for input transcripts, the raw transcripts needed to be reformatted in order to be parsable by the system. Specifically, the sentence-tagging scheme required calculating the duration of each sentence, and rewriting the tag to include not only the sentence number, but also the specific audio file it occurs in, the speaker, and the duration of the sentence. The transcriptions themselves also required some editing to replace symbols being used as special characters by the ASR system, and to remove unnecessary punctuation marks.

Additionally, the ASR system requires a number of supporting files to process the audio and transcript files, including a listing of speakers and corresponding genders, a list of speakers with corresponding utterance start and end times and audio filenames, and a file listing for each speaker all audio files where they have an utterance. The original datasets did not include these lists. Therefore we had to construct these using information parsed from the original sentence tags and the original list of sentence start and end times.

Another vital component of the corpus was the pronunciation dictionary, which we built by combining several different dictionaries. An entry in the dictionary consisted of a word composed of one or more Chinese characters followed by alphabetical transcriptions of how the word is pronounced. Alternate pronunciations were separated by spaces, and each entry occupied its own line in the file. In addition to Chinese words, the dictionary also contained place-holding definitions for special sounds such as coughs, breaths, laughs, lipsmacks, and silences.

The ASR system also requires supporting files accompanying the dictionary that detail the linguistic properties of phones in the language and define the symbol used to represent each phone. To create these files, we modified existing versions used with the English corpuses by modifying the linguistic descriptions and defining additional symbols to represent sounds in Mandarin Chinese that are not found in English.

3.1.2 Chinese Character Encoding Compatibility

Due to the graphical nature of the Chinese writing system, special encoding schemes are needed to represent Chinese characters in computing. The most commonly used schemes include Guobiao (GB), used mainly in mainland China and Singapore, Big5, used in Taiwan, Hong Kong, and Macao, and Unicode. The transcripts in both the training and evaluation datasets were encoded in Unicode UTF-8. The training and decoding processes themselves were compatible with the UTF-8 encoding. However, sclite, the rescoring tool used by the system, was only compatible with GB2312 character encodings. To convert the output of the rescoring tool from UTF-8 to GB2312 encoding, we use the iconv program, followed by a script to manipulate the formatting to fit the requirements of sclite.

3.2 Baseline Pitch Incorporation

Because the five tones in Mandarin Chinese are distinguished by different pitch fluctuation patterns, making use of pitch from the audio data is a simple way to incorporate tone-related information. Not only is the absolute pitch value at any given time useful, but also the variations in pitch over time, as those are what truly differentiate one tone from another. Therefore, we also make use of the first and second derivatives of the pitch data to capture the temporal aspect of tone.

Our baseline pitch-incorporating algorithm was based on the algorithm described in [26] and sought to reproduce similar relative improvements over our original non-pitch system as those observed by Lei, et al. In order to follow their methods as closely as possible, our baseline pitch system consists of three major steps:

1. Extract raw pitch track.
2. Smooth and normalize pitch track.
3. Create new feature vector incorporating pitch data.

3.2.1 Extracting a Pitch Track

Extracting a pitch track consisting of a pitch value for every sample in the audio data is the first step of extracting pitch features. The initial solution was to use the built-in pitch extraction and interpolation tools in Praat [10], a commonly used speech processing software. While this yielded reasonable-looking values, we wanted to have more direct control over the interpolation and smoothing methods used to create a continuous pitch track. Thus, we decided to extract the pitch tracks using jaguarpitch [30].

Jaguarpitch takes a variety of audio formats as input and can perform a variety of manipulations, including pitch extraction. The raw output from jaguarpitch consisted of a text file with a line for each sample stating the time of the sample (within the audio file), a value for extracted pitch, and a third value between 0 and 1.0 representing voicedness of the sample. In Mandarin Chinese, as in almost all languages, there are sounds that can be pronounced without vibration of the vocal cords, termed unvoiced phones. Examples in English include *s*, *p*, and *k* (contrast these with their voiced counterparts *z*, *b*, and *g*). The lack of vocal vibrations during these phones means pitch is undefined for these phones, and therefore the values provided by any pitch extractor will not be accurate. Thus, we set a minimum voicedness threshold of 0.9, and replaced the pitch value with a placeholder value of -1 for any sample with voicedness less than 0.9, with the intention of interpolating pitch for these samples in a subsequent step.

3.2.2 Interpolating Undefined Pitch Areas

Because the raw pitch tracks from jaguarpitch contained undefined values during unvoiced phones (see Figure 3-1(a)), we needed to interpolate values for these samples to obtain a continuous pitch track for the entire audio file.

For the sake of simplicity, the first interpolation scheme we implemented replaced the undefined pitches with the mean of all the defined pitch values within the track. As can be seen in Figure 3-1(b), this is an improvement over the raw pitch track. However, this interpolation scheme does a poor job of modeling realistic pitch behavior, resulting in many points with sudden jumps in pitch values.

To improve on the first scheme, we tried a method of having interpolated values exponentially approach the mean of the defined pitch values. Interpolated pitch values were calculated using the following equation:

$$p(i) = \mu + (p_o - \mu)e^{-0.00475n} \tag{3.1}$$

where $p(i)$ is the interpolated pitch value for sample i, μ is the mean of all defined pitch values in the pitch track, p_o is the most recent defined pitch value preceding sample i, and n is the number of undefined pitch values between i and the sample number of p_o.

While this interpolation scheme provides more reasonable behavior at the beginning and end of the audio file, for the most part it still results in nearly as many sudden jumps in pitch value as the first interpolation scheme, with the addition of some very unnatural exponential behavior in extended areas of interpolated pitch, as can be seen in Figure 3-1(c).

Because the simple schemes involving replacement of undefined pitch with the mean pitch value give less than acceptable results, we tried a simple linear interpolation scheme in which the interpolated value at sample i, if originally undefined, was based on the preceding pitch value (original if defined, or interpolated if undefined) and the following pitch value or the mean if sample $i+1$ has undefined pitch:

$$p(i) = \begin{cases} \mu & i = 1 \\ p(i-1) + \frac{\mu - p(i-1)}{2} & i = n \text{ or } p(i+1) \text{ undefined} \\ p(i-1) + \frac{p(i+1) - p(i-1)}{2} & \text{otherwise} \end{cases} \tag{3.2}$$

where $p(i)$ is the interpolated pitch value for sample i, μ is the mean of all defined pitch values, and n is the total number of samples.

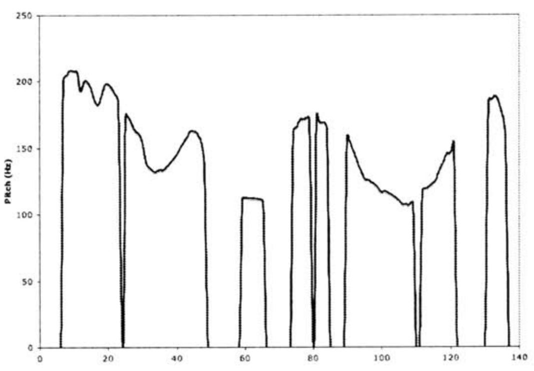
As can be seen in Figure 3-1(d), linear interpolation results in far fewer unnatural behaviors than either simply inserting the mean pitch value, or interpolating by exponential approach to the mean pitch value. However, there are still points where sudden jumps in pitch value occur.

Although linear interpolation is simple and gives some decent results, we decided to try a polynomial interpolation scheme by calculating a piecewise cubic Hermite interpolating polynomial (PCHIP) for each pitch track. This is the method used in [26], whose algorithm we were trying to reproduce. The previous interpolation schemes had all been simple enough to be manually included in our own code, but to do PCHIP interpolation, we chose to use the *pchip* function in MATLAB. This interpolation method gives some very nice results, as seen in Figure 3-1(e). It emulates natural-looking pitch behavior at the beginning and end of the pitch track, and also eliminates any sudden jumps in pitch that were still observable in the linear interpolation result.

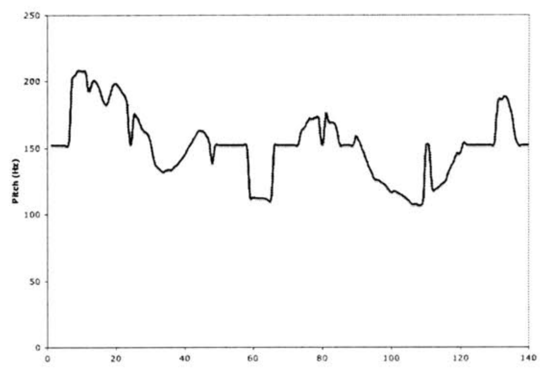
Using MATLAB's *pchip* function in batch mode does add some computation time to the overall process. However, the interpolation and pitch extraction only needs to be performed once for an audio file. Therefore, after the first time, this process does not need to be repeated for subsequent experiments using the same set of audio data. Because it gave the best results while still being relatively easy to implement, we chose to use PCHIP as our final interpolation method.

3.2.3 Normalizing and Denoising Pitch Track

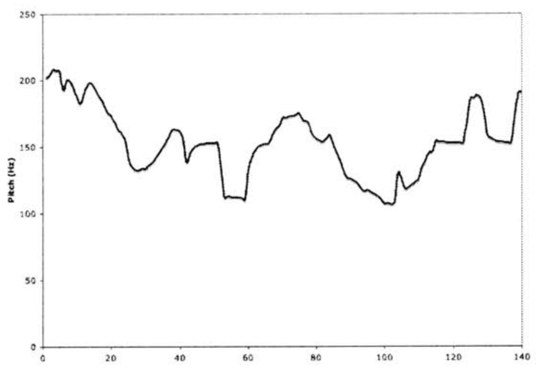
Pitch variation patterns on the syllable timescale are key to distinguishing different tones in Mandarin Chinese. However, the existence of lexical tone does eliminate the presence of phrase-level intonation used to carry emotion-related information, as in non-tonal languages. For example, if the speaker is asking a question, not only will there be the small scale pitch fluctuations with each syllable, but there will also be an overall upward trend in the pitch track, carrying the question inflection. In order to compensate for phrase-level intonation, we normalized each pitch track after interpolation. Additionally, we applied smoothing to compensate for any irregularities



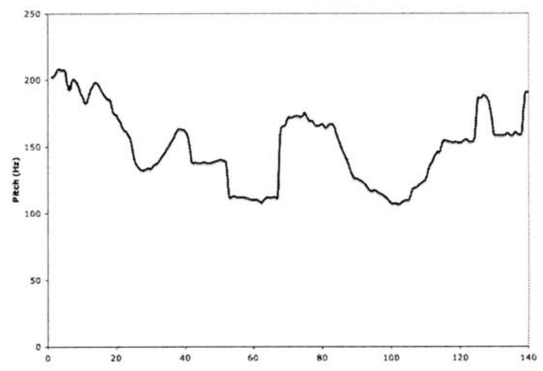
(a) Raw pitch track



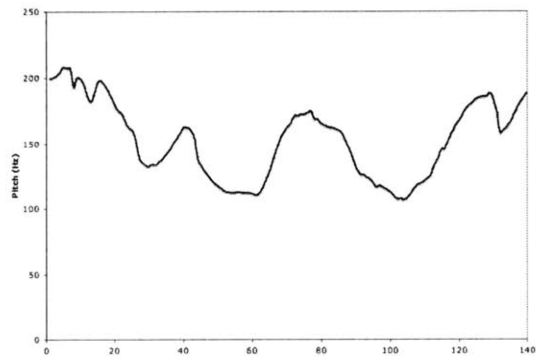
(b) Fill with mean



(c) Exponential approach to mean



(d) Linear interpolation



(e) PCHIP interpolation

Figure 3-1: Results of various pitch interpolation schemes.

caused by outlying pitches or pitch extraction and interpolation errors, and general noise in the pitch data.

Our normalization and smoothing process follows that described in [26]. Thus, or normalization and smoothing algorithm is as follows:

1. Take the log of the pitch track.
2. 100-sample moving window normalization
3. 5-point moving average smoothing

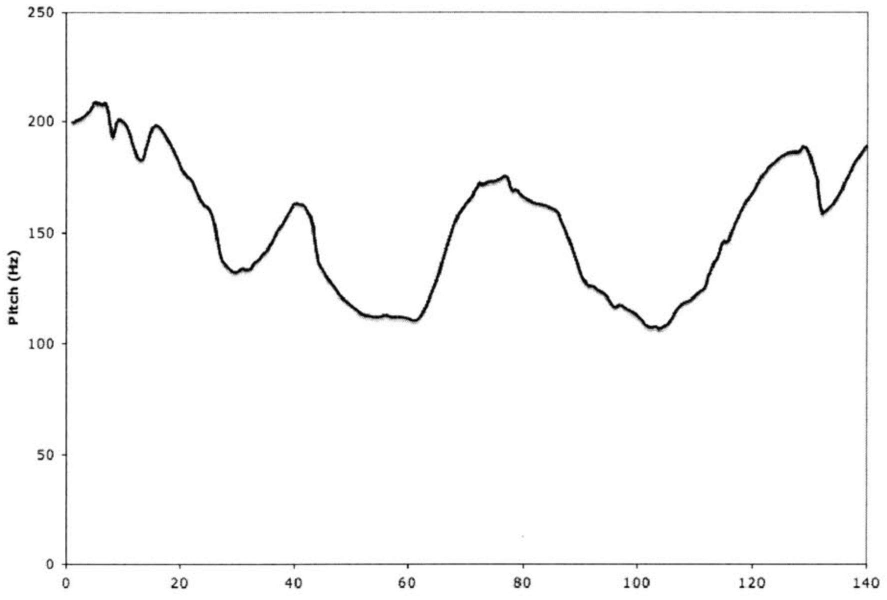
This process not only normalizes the pitch data, but also removes a significant amount of noise, as can be seen in the comparison in Figure 3-2.

3.2.4 Creating a Feature Vector with Pitch

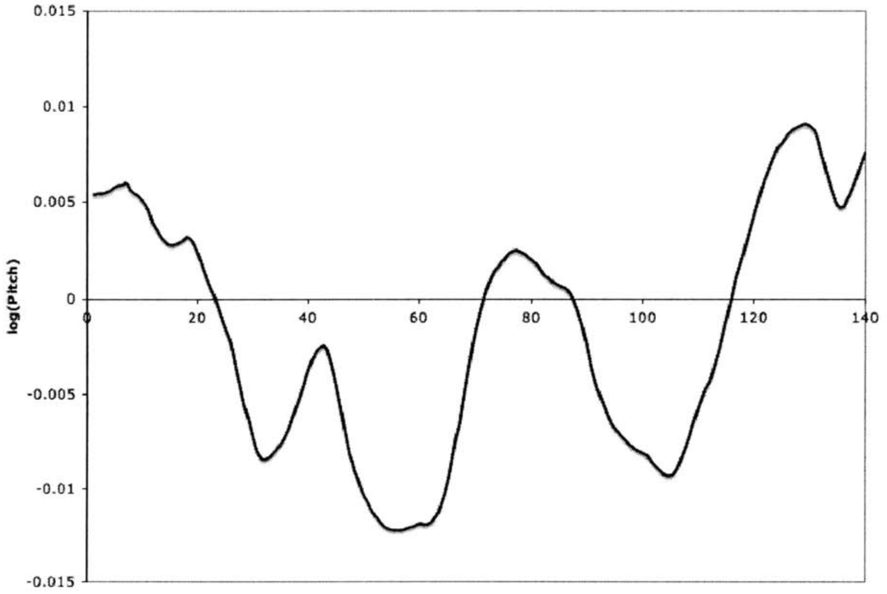
The simplest way to incorporate pitch information is to create a pitch feature vector $[pitch, dpitch, d^2pitch]$ (consisting of pitch and its first and second derivatives, respectively) and concatenate these features to the end of the cepstral feature vector being used by the original ASR system. This method of incorporating pitch was also used by [26], whose results we were trying to reproduce. Thus, we create a new feature vector: $[39\ cepstral\ features, pitch, dpitch, d^2pitch]$ in our system's front-end feature extractor and train our models as we normally would with only cepstral features.

3.2.5 Baseline Experiment Results and Discussion

Experiments testing the effectiveness of the baseline pitch-incorporation algorithm were run with no adaptations during training and decoding, and also with combinations of SAT, MLLR, and VTLN adaptations enabled. The parameters of these experiments were set up to match those used by [26] as closely as possible. Our phonetic models were trained with 32 mixture components per state, and we performed separate runs using bigram, trigram, and 4-gram language models. The results for bigram, trigram and 4-gram language models are presented in Tables 3.1, 3.2, and 3.3, respectively.



(a) PCHIP interpolation



(b) Normalized and smoothed

Figure 3-2: Comparison of interpolated pitch track (a) vs. normalized and smoothed pitch track (b).

	No Adaptation	SAT, MLLR, VTLN
No Pitch	25.1	22.9
Baseline Pitch	24.7	22.0

Table 3.1: Baseline experiment results (word error rate) with bigram language models.

	No Adaptation	SAT, MLLR, VTLN
No Pitch	24.5	22.4
Baseline Pitch	23.9	21.7

Table 3.2: Baseline experiment results (word error rate) with trigram language models.

	No Adaptation	SAT, MLLR, VTLN
No Pitch	24.6	22.4
Baseline Pitch	24.0	21.7

Table 3.3: Baseline experiment results (word error rate) with 4-gram language models.

The improvement from incorporating pitch is most noticeable when using SAT, MLLR, and VTLN with a bigram language model. While our improvements are more modest than those observed by the authors of [26], the consistent improvement across all experiments when incorporating pitch features shows that the additional information can serve to improve recognition of Mandarin Chinese.

3.3 Tone Modeling with Multi-stream HMMs

As described in Chapter 2, model clustering is used by our ASR system to take advantage of contextual information when training our phonetic models. Without pitch features, the clustering process works well for modeling phonemes by context. However, our baseline pitch-incorporating system appends the pitch features to the original PLP feature vector, resulting in an input vector that is no longer purely phone-based, but also includes tone information. When we perform model clustering in the baseline system with pitch information, the homogeneity of our data is now a combination of effects from both the phones and the tones. The criteria for clustering, however, remained the same as in the original ASR system. Therefore, properties of the left and right contexts were described for phones. For example, a question used in the decision tree may be 'R-UnRoundVowel' meaning "is the right context an unrounded vowel." This question, however, does not differentiate between the differing tones carried by the vowels falling in the unrounded category, and therefore does not make effective use of tone contexts. In order to fully take advantage of separate cepstral and pitch context dependencies for model clustering, we propose an approach using multi-stream HMMs. We are optimistic about our approach based on the findings of Mak and Tam on HMM recombination [29] as well as Lei and colleagues in their study with multi-stream DBN models [25]. We choose to implement our algorithm using HMMs because graphical models such as DBNs are very computationally complex, and also for ease of integrating with our existing ASR system.

The HTK Toolkit offers built-in functionalities for model clustering, including the decision-tree based clustering method being used in both the original and the baseline pitch-incorporating ASR system. However, although the HTK Toolkit offers limited support for multi-stream HMMs, there are no functions supporting decision-tree clustering for multi-stream models. Therefore, we implemented an algorithm in which two sets of models, one for cepstral data and one for pitch data, were trained and clustered using the HTK functionalities, and the two models were then tied through an auxiliary script we wrote to generate our own HMM definition file. The major steps of this algorithm are as follows:

1. Train a phone model using cepstral features.
2. Train a tone model using pitch features.
3. Manually generate a multi-stream HMM definition file combining the phone and tone models.
4. Train the multi-stream model.

A conceptual illustration of combining a triphone and tritone model is given in Figure 3-3. After steps 1 and 2 of our algorithm, we will have separate phone and tone models, which can then be combined in a multi-stream HMM.

3.3.1 Training a Phone Model

Training a phone model was a simple step, because we needed only to provide a feature vector containing PLP features without pitch, as was used in the original ASR system. This was used as the input vector to the modified ASR system, resulting in a set of

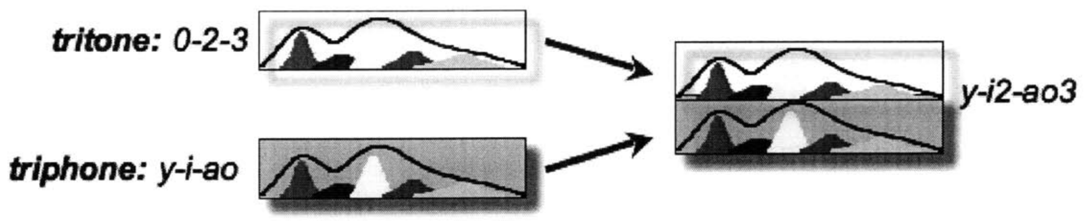


Figure 3-3: Conceptual illustration of multi-stream HMM for independent tone and phone model clustering.

HMMs trained using only cepstral data. Because we need to manually combine the phone and tone models after independent clustering, we are only interested in the models resulting immediately after the model clustering step of the training process. The remainder of the training process will be completed after the phone model is manually incorporated into the multi-stream model.

3.3.2 Training a Tone Model

Conceptually, the process for building a tone model is similar to that for a phone model. We need to operate our modified Chinese-compatible system with an input vector containing only pitch features. However, the dictionaries and auxiliary linguistic data files being used in the Mandarin Chinese corpus contain pronunciation definitions in the form of phones with an attached tone. In order to train a tone model, we needed to construct a new set of dictionary and auxiliary files containing only the tones. We chose to convert the original dictionary by representing toned segments with *t1, t2, t3, t4, t5*, and unvoiced (and therefore untoned) segments were replaced with the null tone *t0*. Examples of this conversion are given in Table 3.4. Additionally, we modified the list of acceptable segments to include only *t0* through *t5* rather than the original phones.

j-y-o4-w	→	t0-t0-t4-t0
i1-g-e4	→	t1-t0-t4
w-e2-y	→	t0-t2-t0

Table 3.4: Examples of conversion to tone dictionary.

Using the new tone-only dictionary and associated auxiliary files, we can then provide as input to our system a vector containing only pitch features, and train a set of tone models in the same manner as our phone models.

3.3.3 Combining Phone and Tone Models

The HMM definition files output by the HTK training functions contain a number of units used to define HMMs:

- Transition matrices: For each phone or tone segment used in the pronunciation dictionary, there is a transition matrix representing the state transition probabilities for that segment. Because the HMMs we used have three states with an additional start state and end state, these are 5 by 5 matrices.

- Mixture components: In the clustering process, each triphone or tritone used in the training data is associated by the decision tree with a specific mixture component, based on the left and right context. Each of these mixture components contains a set of Gaussian models representing each feature in the data vectors. For example, a mixture component in the phone model contains a set of 39 mean values, one for each PLP feature, and 39 corresponding variance values.

- HMM definitions: Each triphone or tritone appearing in the training data has a definition for the HMM modeling the it. The definition specifies the number of features, and optionally the number of streams, number of features per stream, and relative stream weights. The HMM is defined by calling on the mixture component associated with the triphone or tritone, based on the clustering process results. The transition probabilities for the HMM are defined using the transition matrix for the center phone or tone.

To generate our own HMM definition file, we first copied all transition matrices from the phone model definition. Secondly, we copied all the mixture component definitions from both the phone model and the tone model definitions as the set of mixture components to be used in our multi-stream HMMs. To define the HMMs themselves, we specified number of streams to be two, with equal weighting between the cepstral and the pitch streams. To determine which phone and tone mixture components corresponded to the triphone, we wrote an auxiliary script to parse and trace through the clustering decision trees. The cepstral stream was then defined by referencing the appropriate mixture component, and the pitch stream likewise. Our manual generation process resulted in a two-stream HMM with the 39 cepstral features in the first stream, and the three pitch features in the second stream.

3.3.4 Training the Multi-stream Model

In order to implement our multi-stream approach within the framework of the original ASR system, we ran the system up through the completion of model clustering twice: once for building a phone model set, and once for building a tone model set. We then generated a multi-stream HMM definition file combining the clustered tone and phone models. To complete training the multi-stream model, we simply continued running the system starting after the model clustering stage.

3.3.5 Multi-Stream Experiment Results and Discussion

The preliminary results from our multi-stream experiments show unexpectedly bad performance indicative of possible errors or bugs in our implementation (see Table 3.5).

	Word Error Rate
Bigram	87.8
Trigram	88.0
4-gram	88.0

Table 3.5: Preliminary multi-stream experiment results (word error rate) for bigram, trigram, and 4-gram language models.

However, due to time constraints, we were unable to resolve the possible issues (detailed below) for inclusion in this thesis.

Unresolved Implementation Issues

Firstly, our implementation does not make use of stream weights in either training or decoding. In both processes, the cepstral and the pitch streams were weighted equally at 1.0. While Lei and colleagues [25] also do not use stream weights during training, they do make use of stream weights with slight emphasis on the pitch stream during decoding (0.5 for cepstral stream and 0.6 for pitch stream). Therefore, the lack of intelligent stream weighting in our model could be a contributing factor to our unsatisfactory results. We had originally planned to compute optimal stream weights using a linear programming algorithm as detailed in [28]. Unfortunately, this was not implemented due to time constraints. Future modifications should include implementing a procedure for training intelligent weights and including them in the decoding process.

Another source of errors may lie in our method for manually generating a multi-stream method. Our process was based on the assumption that HTK macros originally defined as a shared state distribution could be copied and used as a mixture component. HTK lists these as two separate macro types. However, both are defined for input vectors of length n using a unique macro name, a set of n means, a set of n variances, and a Gaussian weighting constant. Therefore, we assumed it would be possible to relabel the shared state distribution macros resulting from the HTK tree-based model clustering and state-tying method as mixture components in our multi-stream model definitions. However, this may not be the case, and the changed labeling could result in unexpected behavior if the state distributions are treated significantly differently by HTK than the mixture components. This issue could be avoided by directly copying the actual sets of means, variances, and the Gaussian constant into the HMM definitions rather than referring to a macro. This would require a more complex parsing procedure and could take more time to generate the multi-stream definitions file because for each HMM defined, our process would need to find the actual state distribution macro beinf referred to and copy the contents into the HMM definition.

It is already clear that HTK has limited support for multi-stream models, resulting in the need to manually generate a multi-stream HMM defnition file. There may have been further complications during the training process, such as in our model-splitting method, resulting from incomplete support for multi-stream models in HTK methods used for that step. In this case, larger portions of the model generation and training process may need to be implemented either outside of the HTK system or through modifying the HTK codebase for our purposes.

The poor performance could also be a result of errors in our creation of multi-stream models. During one of our first experiments with the multi-stream system, we discovered that we had been parsing the decision trees used for clustering our tone-only and phone-only models incorrectly, reversing the right and left branchings. That problem was fixed for the preliminary results presented in this section, but there may be other errors either in parsing the decision trees, or in generating the contents of the HMM definition file that would result in a set of incorrect models being trained. Due to time constraints, we were unable to fully debug and scrutinize our model generation process for inclusion in this thesis.

Expected Results

With a fully operational multi-stream system, we expect to see results corroborating the findings of [25], in which multi-stream models provided a 3.4% increase in toneme recognition accuracy over a single-stream model. Therefore, we would expect to see results with word error recognition rates of around 20% without any adaptation methods applied.

Further improvements could then be achieved by incorporating SAT, MLLR, and VTLN adaptations as used in the baseline pitch incorporation system. With those adaptations, we would expecte to see word error recognition rates of around 18-19%. The current multi-stream system is incompatible with the SAT and MLLR implementations used in the non-tonal and baseline tonal ASR systems due to lack of multi-stream support in the HTK functions used in those implementations. Therefore, enabling these adaptations would require either an implementation outside of the

HTK framework, or a modification of the HTK codebase.

Chapter 4

Conclusions and Future Directions

In this thesis, we successfully modified an existing non-tonal ASR system to incorporate pitch features for embedded tone modeling in recognition of Mandarin Chinese. We compiled and formatted a training and evaluation Mandarin Chinese corpus for use with the system, and implemented a pitch feature extraction algorithm following that of Lei and colleagues [26]. Our baseline pitch incorporation system concatenates the pitch features onto the PLP feature vector being used for non-tonal ASR for embedded tone modeling. Our baseline experiment results demonstrate improved performance through pitch incorporation, with a decrease in word error rate by abouth 4% relative. This corroborates the findings of previous researchers by demonstrating that pitch feature incorporation is beneficial to performance of ASR systems on Mandarin Chinese.

We implemented and performed preliminary experiments on a further modification incorporating multi-stream HMMs. We proposed using separate streams for the PLP features and the pitch features in order to allow independent model clustering for phone and tone context dependencies. However, due to time constraints, there are still several unresolved implemenatation issues in the multi-stream system, and therefore further work will need to be done before any conclusive results can be presented for the system. With a fully debugged system and adaptation methods enabled, we expect to see word recognition error rates of around 18-19%, which would be about a 3% absolute decrease in word error rate over our baseline single stream HMM system.

4.1 Future Directions

4.1.1 Limited-asynchronous Stream Models

While a multi-stream HMM approach allows for independent phone and tone model clustering, it still does not fully represent the tone-phone dynamics of spoken Mandarin. In their study using multi-stream DBN models [25], Lei and colleagues observed that allowing a limited amount of asynchrony between the cepstral and tonal data streams resulted in a further improvement in performance over models with synchronized streams. This corroborates the findings of [29, 27] as well on the use of multiple asynchronous streams in speech recognition. In a synchronized stream model, although cepstral and pitch features can be independently clustered for context dependency, when combined into a multi-stream model, tones and phones are still constrained to undergoing simultaneous state changes. However, real speech is far more flexible, and tone and phone transitions are very likely not perfectly synchronized. Therefore, allowing limited asynchrony between streams may help to capture this flexibility, resulting in improved performance.

Limited-asynchronous modeling could be implemented for our ASR system by modifying the multi-stream process implemented in this thesis. Within the combined phone and tone model, the set of allowable tone states can be expanded to allow slight offsets between tone and phone changes. For example, as illustrated in 4-1, rather than having the change in phone occur with a corresponding change in tone,

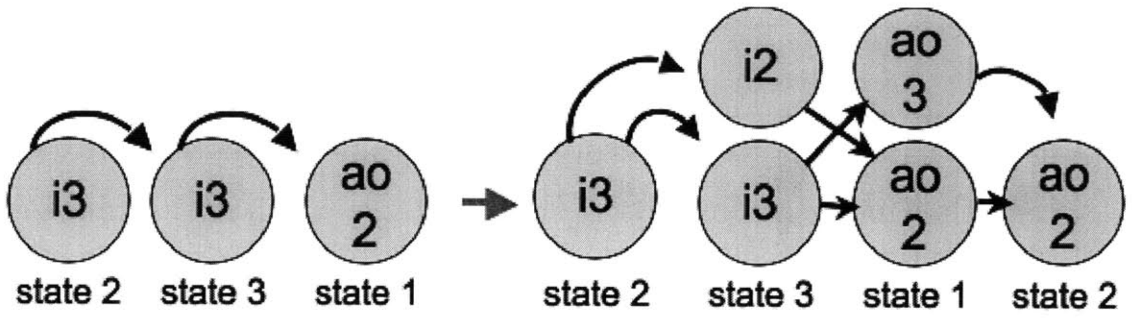


Figure 4-1: Illustration of expanded state sets for limited-asynchronous modeling.

there could be an additional candidate state using the previous tone with the new phone. Similarly, to allow a change in tone before the phone change, we can add an additional candidate state representing the new tone but carried by the previous phone.

Bibliography

[1] L.R. Bahl and F. Jelinek. Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition. *IEEE Trans. Information Theory*, IT-21:404–411, 1975.

[2] L.R. Bahl, F. Jelinek, and R.L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-5:179–190, 1983.

[3] J.K. Baker. The dragon system – an overview. *IEEE Trans. Acoustics, Speech, Signal Proc*, ASSP-23(1):24–29, February 1975.

[4] R. Bakis. Continuous speech word recognition via centisecond acoustic states. *Proc. ASA Meeting (Washington, DC)*, April 1976.

[5] L.E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[6] L.E. Baum and J.A. Egon. An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology. *Bull. Amer. Meteorol. Soc.*, 73:360–363, 1967.

[7] L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Ann. Math. Stat.*, 37:1554–1563, 1966.

[8] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Stat.*, 41(1):164–171, 1970.

[9] L.E. Baum and G.R. Sell. Growth functions for transformations on manifolds. *Pac. J. Math.*, 27(2):211–227, 1968.

[10] Paul Boersma and David Weenink. Praat: doing phonetics by computer (Version 5.1.12) [Computer program]. Retrieved from `http://www.praat.org`, August 2007.

[11] Eric Chang, Jianlai Zhou, Shuo Di, Chao Huang, and Kai-Fu Lee. Large vocabulary mandarin speech recognition with different approaches in modeling tones. In *In Proc. ICSLP 2000*, pages 983–986, 2000.

[12] M. Dell'Amico, S. Martello, Chih-Heng Lin, Chien-Hsing Wu, Pei-Yih Ting, and Hsin-Min Wang. Frameworks for recognition of mandarin syllables with tones using sub-syllabic units. *Speech Communication*, 18:175–190(16), April 1996.

[13] San Duanmu. *The Phonology of Standard Chinese.* Oxford University Press, Oxford, Oxfordshire, 2000.

[14] Jonathan Fiscus et al. 1997 HUB4 Broadcast News Evaluation Non-English Test Material. Linguistic Data Consortium, Philadelphia, 2001.

[15] H. Fletcher. Auditory patterns. *Rev. Mod. Phys.*, 12:47–65, 1940.

[16] Ben Gold and Nelson Morgan. *Speech and Audio Signal Processing: Processing and Perception of Speech and music.* John Wiley & Sons, Inc., New York, NY, 2000.

[17] Hynek Hermansky. Perceptual linear predictive (PLP) analysis of speech. *J. Acoust. Soc. Am.*, 87:1738–1752, 1990.

[18] Hsiao-Wuen Hon, Baosheng Yuan, Yen-Lu Chow, S. Narayan, and Kai-Fu Lee. Towards large vocabulary mandarin chinese speech recognition. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, volume i, pages I/545–I/548 vol.1, Apr 1994.

[19] F. Jelinek. A fast sequential decoding algorithm using a stack. *IBM J. Res. Develop.*, 13:675–685, 1969.

[20] F. Jelinek. Continuous speech recognition by statistical methods. *Proc. IEEE*, 64:532–536, April 1976.

[21] F. Jelinek, L.R. Bahl, and R.L. Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Trans. Information Theory*, IT-21:250–256, 1975.

[22] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Pearson Education, Inc., University of Colorado, Boulder, 2000.

[23] Tan Lee, Wai Lau, Y.W. Wong, and P.C. Ching. Using tone information in Cantonese continuous speech recognition. *ACM Transactions on Asian Language Information Processing*, 1(1):83–102, March 2002.

[24] Xin Lei, Mei-Yuh Hwang, and Mari Ostendorf. Incorporating tone-related MLP posteriors in the feature representation for Mandarin ASR. In *Proc. Interspeech 2005*, pages 2981–2984, September 2005.

[25] Xin Lei, Gang Ji, Tim Ng, Jeff Bilmes, and Mari Ostendorf. DBN multistream models for Mandarin toneme recognition. *Proc. ICASSP '05. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1:349–352, 18-23 March 2005.

[26] Xin Lei, Manhung Siu, Mei-Yuh Hwang, Mari Ostendorf, and Tan Lee. Improved tone modeling for Mandarin broadcast news speech recognition. In *Proc. Interspeech 2006*, pages 1237–1240, September 2006.

[27] Karen Livescu and James Glass. Feature-based pronunciation modeling with trainanable asynchrony probabilities. In *In Proc. ICSLP*, October 2004.

[28] Brian Mak and Benny Ng. Discriminative training by iterative linear programming optimization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 4061–4064, April 2008.

[29] Brian Mak and Yik-Cheung Tam. Asynchrony with trained transition probabilities improves performance in multi-band speech recognition. In *In ICSLP-2000*, volume 4, pages 149–152, 2000.

[30] R.J. McAulay and T.F. Quatieri. Pitch estimation and voicing detection based on a sinusoidal speech model. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, volume 1, pages 249–252, 3-6 April 1990.

[31] Gang Peng and William S.-Y. Wang. Tone recognition of continuous Cantonese speech based on support vector machines. *Speech Communication*, 45(1):49–62, January 2005.

[32] Lawrence Rabiner and Juange Biing-Hwang. *Fundamentals of Speech Recognition.* Prentice Hall, Upper Saddle River, New Jersey, 1993.

[33] Alexander Sorin, Tenkasi Ramabadran, Dan Chazan, Ron Hoory, Michael McLaughlin, David Pearce, Fan CR Wang, and Yaxin Zhang. The ETSI extended distributed speech recognition (DSR) standards: client side processing and tonal language recognition evaluation. *Proc. ICASSP '04. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1:129–132, 17-21 May 2004.

[34] Chao Wang and Stephanie Seneff. A study of tones and tempo in continuous Mandarin digit strings and their application in telephone quality speech recognition. In *Proc. 1998 International Conference on Spoken Language Processing*, pages 695–698, Sydney, November 1998.

[35] S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 307–312, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[36] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. *The HTK Book (for HTK Version 3.4)*. Cambridge University Engineering Department, December 2006.