INTEGRATE TEMPLATE MATCHING AND STATISTICAL MODELING FOR CONTINUOUS SPEECH RECOGNITION

A Dissertation

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

XIE SUN

Dr. Yunxin Zhao, Dissertation Supervisor

DECEMBER 2011

The undersigned, appointed by the dean of the Graduate School, have examined the dissertation entitled

INTEGRATING TEMPLATE MATCHING AND STATISTICAL MODELING FOR CONTINUOUS SPEECH RECOGNITION

presented by Xie Sun, a candidate for the degree of doctor of philosophy, and hereby certify that, in their opinion, it is worthy of acceptance.

Professor Yunxin Zhao

Professor Dong Xu

Professor Ye Duan

Professor Nancy Flournoy

Professor Xinhua Zhuang

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dr. Yunxin Zhao, who guided and helped me a lot in my Ph.D study. I am so lucky to have such a nice and great advisor during my Ph.D study. Without her knowledge, perceptiveness and guidance, I would never have obtained my Ph.D degree.

Also, I would like to thank Dr. Ye Duan, Dr. Flournoy Nancy, Dr. Dong Xu, and Dr. Xinhua Zhuang, for their help in my graduate study as my committee members. Thank them for reviewing my dissertation and giving inspirational comments.

Thank all the officemates in the Laboratory of Spoken Language and Information Processing, especially Xin Chen, Yi Zhang, and Tuo Zhao, for their great help and useful discussion both in study and daily life. Working with them has been a great experience in my life.

Many thanks to my wife, Hanshuo Zhuang, for her constant support, encouragement, and endless love. She is my motivation to overcome all difficulties during my Ph.D study.

Finally, thanks my parents and my sister, their love always gives me courage to continue going further.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

In this dissertation, a novel approach of integrating template matching with statistical modeling is proposed to improve continuous speech recognition. Hidden Markov Modeling (HMMs) has been the dominant approach in statistical speech recognition since it provides a principled way of jointly modeling speech spectral variations and time dynamics. However, HMMs have the shortcoming of assuming the observations being independent within each state, which makes it ineffective in modeling the details of speech temporal evolutions that are important for characterizing nonstationary speech sounds. Template-based methods make comparisons between a test pattern and the templates derived from training data, and therefore they are able to capture speech dynamics and time correlation of speech frames better than HMM based methods. However, template matching requires large memory space and computational time since feature vectors of training data need to be stored in computer memory for access at the recognition stage, which is difficult in large vocabulary continuous speech recognition (LVCSR). Our proposed approach takes advantages of both statistical modeling and template matching, which overcomes the weakness of conventional template-based method and is feasible for LVCSR.

We use multiple Gaussian Mixture Model (GMM) indices to represent each frame of speech templates, and define the template unit to be context-dependent phone segments (triphone context). We also use phonetic decision trees borrowed from those commonly used in HMMs to tie triphone templates and predict triphones unseen in training data.

Two local distances, log likelihood ratio (LLR) and Kullback-Leibler (KL) divergence, are proposed for dynamic time warping (DTW) based template matching. In order to reduce computational complexity and storage space, we propose methods of minimum distance template selection (MDTS) and maximum log-likelihood template selection (MLTS), and investigate a template compression method on top of template selection to further improve recognition performance.

The template based methods were used to rescore lattices generated by baseline HMMs on the tasks of TIMIT continuous phone recognition and teleheath LVCSR and experimental results demonstrated that the proposed approach of integrating template matching with statistical modeling significantly improved recognition performances over the HMM baselines. The template selection methods also provided significant recognition accuracy improvements over the HMM baseline while largely reducing the computation and storage complexities. When all templates or MDTS were used, using the LLR local distance obtained better recognition performance than the KL divergence local distance. For MLTS and template compression, KL divergence local distance provided better performance than the LLR local distance, and the template compression method made further improvements over KL based MLTS.

Since the templates were constructed based on the GMM indices extracted from HMM baselines, we also validated the effectiveness of the proposed template methods based on enhanced HMM baselines. Experimental results showed that LLR based all template method was able to consistently improve TIMIT phone recognition accuracies based on four enhanced HMM baselines.

Prosodic features such as duration, energy, and pitch can reflect longer span information of speech than conventional single frame vectors but they have commonly been ignored by HMMs. Template based methods provide possibilities to conveniently integrate prosodic features into speech recognition, which has not been well studied in the past. In this dissertation, we investigate combining template based methods with the speech prosodic features of duration, energy and pitch to further improve speech recognition accuracy. The scores of prosodic information were computed by a GMM based method and a non-parametric method, and the prosodic scores were combined with the acoustic scores in triphone template matching. Experimental results obtained on the telehealth task showed that prosodic information had positive effects on vowel sound recognition.

Chapter 1

Introduction

1.1 Background

Automatic speech recognition is a technology that allows a computer to recognize human speech. For human beings, the way of most direct communication is speech. It has been a long-time dream for humans to make computers recognize human being's speech [1]. Speech interface for computers has advantages in speed and convenience compared with today's popular user interfaces such as mouse, keyboard, or touch-screen, etc. Therefore, speech recognition has lots of potentially useful applications. One example is that people can utilize the speech-input ability of a computer to significantly speed up document writing, email sending, web searching, and other operations with a computer. Some of these applications are already used by us and some of them are going to become available, which can bring great convenience to our lives. In addition, automatic speech recognition system can also greatly benefit disabled people who can use speech as the input to control computers in case they have difficulty to use keyboard or mouse with their hands. People who have hearing problems can also use automatic speech recognition technology to transcribe speech from other people to text, which can make their lives easier. Another advantage of automatic speech recognition is that it can be used for situations when people's hands are already occupied, for example, when driving a car, where people can use speech as input to dial or receive a phone call. Speech enabled GPS navigation is another good example. With the development of machine translation techniques, another exciting application called speech-to-speech translation [2] has emerged, which allows people speaking different languages from all over the world to freely communicate via speech without any professional translator. A specialized application of this kind is already being used in military. Speech technology can also be used in education systems to help improve foreign language learning for young people.

Even though there are many potentially useful applications of automatic speech recognition, a universal application of automatic speech recognition products in human life is still facing great challenges since the complexity of human speech production and perception makes speech recognition a very difficult and complicated task, and automatic speech recognition is deeply involved with multiple disciplines including acoustics, phonetics, linguistics, anatomy, physiology, neuroscience, computer science, electrical engineering, artificial intelligence, and signal processing. However, we believe that under the continuous research efforts of human beings, these difficulties will be overcome and automatic speech recognition technology will be integrated into different products and bring more benefits to human life.

1.2 Motivation

In speech recognition, Hidden Markov Models (HMMs) have been the dominant approach since they provide a principled way of jointly modeling speech spectral variations and time dynamics. However, one of the HMM shortcomings is that it assumes within each state the observations are independent, which makes it ineffective in modeling the fine details of speech temporal evolutions that are important in characterizing nonstationary speech sounds [3]. Some technologies are already used to overcome the weakness of HMMs. For example, time derivatives of cepstral coefficients are widely used to capture dynamic information from the neighbor frame vectors in speech feature distributions in HMM states. Trajectory model [4] is another example that introduces time-varying covariance modeling to capture temporal evolutions of speech features. Additionally, approaches like segment model [5] and long-contextual-span model of resonance dynamics [6] have been proposed for similar purposes. The second problem of HMMs is that it doesn't conveniently model prosodic information such as duration, pitch and energy. Prosody has long been studied as an important knowledge source for speech understanding. In recent years there has been a large amount of computational work aimed at prosodic modeling for automatic speech recognition and understanding [7, 8, 9].

Template-based methods make comparisons between a test pattern and templates of training data, and therefore they are able to capture the speech dynamics and time correlation of speech frames better than HMM based methods. For the template-based approach, no explicit assumption about the data needs to be made and it commonly uses DTW to compare a test pattern against a template or template sequences. Template-based methods were originally used to recognize isolated words or connected digits with good performance [1]. When using template-based methods, feature vectors of training set need to be stored in computer memory, and until recently, it has been practically impossible to apply template-based methods to large tasks of speech recognition. With today's rapid advance in computing power and memory, template-based methods have attracted new efforts and the reported results are promising [10, 11, 12]. In addition, template-based methods can utilize the prosodic information conveniently [9]. Some newly proposed methods like template pruning and filtering [13] and template-like dimension reduction of speech observations [14] are helping to address the memory usage problem. However, there is always a compromise between the costs in computation and space and the recognition accuracy.

Combining statistical modeling and template-based approach can overcome the weakness of HMMs while significantly saving computational and storage complexities. It has been shown that by combining HMM-based and template-based recognition at the system level, accuracy performance can be improved over either system alone [15, 16]. However, this kind of combination is not so simple to be applied into large vocabulary continuous speech recognition (LVCSR). In this dissertation, we propose a within system combination of statistical modeling and template matching. This novel integration approach includes new methods for template construction, template tying, and template matching, etc. We also propose template clustering methods to reduce computing time and storage space which allow our approach to be applied to LVCSR.

As we mentioned above, the traditional HMM based method ignores prosodic information where the information is discarded during the process of feature extraction. There has been some research efforts on first modeling prosodic information separately and then combine the prosodic model with the conventional HMM based system. However, using template based methods to integrate prosodic information has not been well studied. Since our template methods are different from the traditional ones [10, 11, 12], the integration of our template based methods and prosodic information is a new problem, and this topic is investigated in this dissertation.

1.3 Contribution of the Current Work

In our current work, we formulate a novel approach of integrating template matching with statistical modeling. The main contributions of the current work are:

1) Instead of using the conventional speech frame feature vectors to represent the templates, we propose to use Gaussian Mixture Model (GMM) indices to represent speech frame vectors for templates. The GMMs are obtained from the well trained baseline HMMs, and the templates are defined as the context-dependent phone segments (triphone context). The compact representation of GMM based templates converts the high dimensional float type frame vectors to integer GMM indices, which can save lots of storage space compared with the traditional frame-vector based templates. By using the Dynamic Time Warping (DTW) based template matching for the new template representation, the shortcoming of HMMs, which is unable to model the fine details of speech temporal evolutions, is overcome. Therefore, the newly proposed method for template representation offers the advantages of both statistical modeling and template matching.
2) Since the templates are constructed and represent by GMM indices, the traditional local distances such as Euclidean distance and Mahalanobis distance are not suitable any more. New local distances are needed in order to performance template matching for the template representations. We explore the local distances of log-likelihood ratio and Kullback-Leibler (KL) divergence for the proposed statistical modeling based template matching method.
3) Since the template based methods are implemented by using the GMM indices extracted from baseline HMMs, we also want to know if further recognition improvements can be obtained when better HMM baselines are used. We verified that the proposed template matching method consistently improved HMM based

system performance on the task of TIMIT phone recognition by using different HMM baselines generated from the four methods investigated in [17]: 1) Discriminative Training (DT) of Minimum Phone Error (MPE), 2) MFCC concatenated with ensemble Multiple Layer Perceptron (MFCC+EMLP) features, 3) DT combined with the MFCC+EMLP features, and 4) data sampling based ensemble acoustic models integrated with DT and MFCC+EMLP features.

4) Traditional template matching methods were used in small scale speech recognition tasks in which the test units were fully covered by the training data. However, if template-based methods are used in LVCSR, allophones which are present in training data may not cover all allophones in test data. In general, how to deal with unseen allophones in test speech data is an issue needed to be solved. In our template-based method, we borrow the idea from HMM state Phonetic Decision Trees (PDT) to assign an unseen triphone into a known cluster to solve the unseen triphone template problem.

5) Even though the proposed novel statistical modeling based template method can significantly reduce computation time and storage space compared with the traditional template based approaches, the extra cost of the method over HMM baseline systems is still high when it is used for large vocabulary continuous speech recognition. So we further propose methods of template selection and compression based on the local distances mentioned in 2). The proposed template selection algorithm significantly reduced computation and storage complexities, and the compressed templates produced further performance improvement based on the selected template representatives. The template selection and compression

methods were effective in removing low performance templates, keeping and generating good quality template representatives, and therefore obtaining relatively high recognition performances compared with using all templates.

6) We explored using prosodic information of duration, energy and pitch to improve template matching based LVCSR. The prosodic feature scores were calculated using two different methods: GMM based method and non-parametric method. The integration of prosodic information into template based methods improved recognition accuracies for LVCSR.

1.4 Statistical Speech Recognition

Automatic speech recognition (ASR) is the task of converting a speech input signal into text by a computer. A speech utterance produced by a speaker is represented in the form of sound waves. Microphones capture the sound waves and convert them to electrical signals. Speech features are extracted from the electrical signals and stored in computer memory. Current speech recognition system searches over a large time-state space to find the word string hypothesis with the highest probability of generating the speech utterance. In order to do this, three steps are implemented. First, speech signals are analyzed to obtain frame feature vectors in which necessary information is retained for speech sound discrimination. Second, statistical language model and acoustic model are estimated from training data. Finally, in order to pick the sentence hypothesis with the highest probability, fast and memory-efficient search algorithms are needed. Figure 1.1 [1] shows a block diagram for speech recognition.

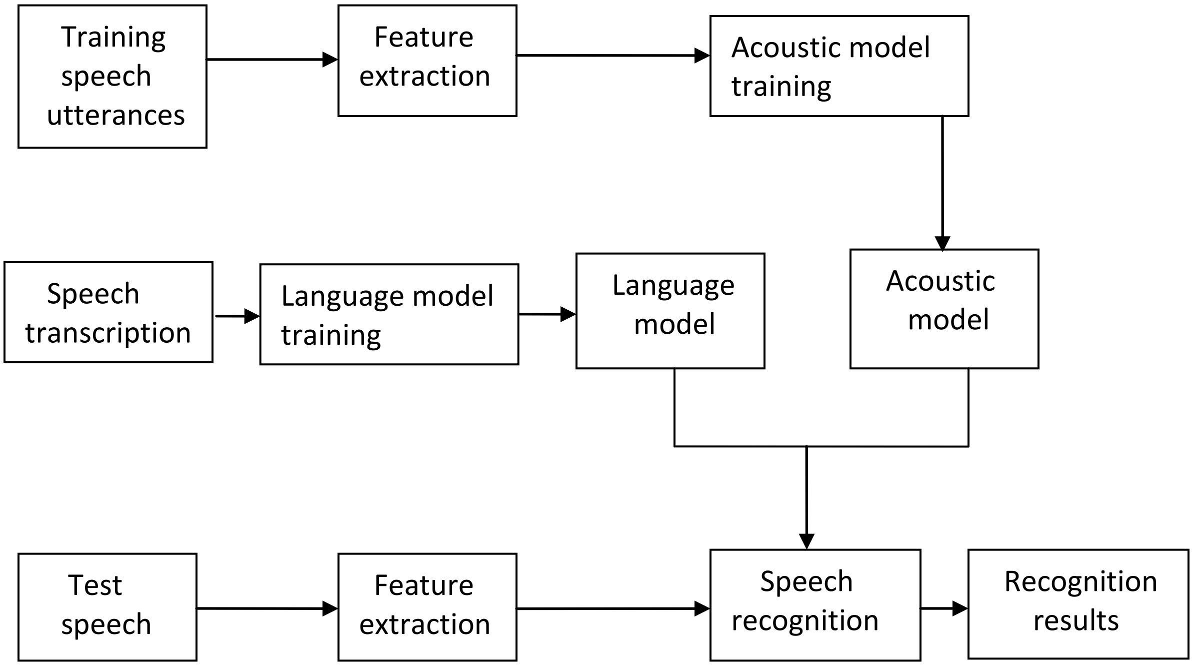

Fig 1.1 Diagram of Speech Recognition System

A speech recognizer maps a sequence of observation vectors of speech into its underlying word sequence, i.e., to find a word string $\widehat{W}$ corresponding to the acoustic observation O=$o_1,o_2,\ldots,\ o_T$, where $\widehat{W}$ is the hypothesis of the highest probability that best matches the words spoken in the speech. We can also use Bayesian decision theory to formulate the speech recognition problem as the following [1]:

$$\widehat{W} = \underset{W}{\operatorname{argmax}} P(W|O) = \underset{W}{\operatorname{argmax}} \frac{P(O|W)P(W)}{P(O)} = \underset{W}{\operatorname{argmax}} P(O|W)P(W) \qquad (1.1)$$

where the observation likelihood $P(O|W)$ is the probability that the speaker produces the acoustic feature vector sequence O under the condition of the word sequence W, and $P(O|W)$ is evaluated based on an acoustic model; $P(W)$ is the prior probability of the word sequence W and is determined by a language model; $P(O)$ is the prior probability of observation O, which can be neglected because it is the same for all hypotheses W and does not affect the decision.

The estimation of $P(O|W)$ is also called acoustic modeling and it typically consists of two parts. The first part is to describe the representation of a word sequence by sub-word units, which is also known as pronunciation modeling. The second part is to map from each sub-word units to acoustic observations [18]. Algorithms used in acoustic modeling involve hidden Markov models (HMM) and phonetic decision trees (PDT) which will be explained in Section 1.6 and in Section 1.7, respectively.

The goal of language modeling is to estimate the probability $P(W)$. Statistical *n*-gram is most commonly used in language modeling and it uses the previous history words to predict the current word, i.e., the probability of the current word is conditional on the previous *n-1* words. More details about the language model will be introduced in Section 1.8.

1.5 Pre-processing of Speech

Since an effective representation of speech signals is required for speech recognition, speech feature extraction is a very important step in pre-processing of speech. The raw data as input to an ASR system is the speech waveform sampled at a rate between 8 kHz (for telephone speech) and 20 kHz. This data are pre-processed to generate feature vectors which are usually computed from overlapped sliding windows of 20 to 30ms in duration at a 10ms frame rate. There are two well-known feature extraction algorithms as the following [1]:

1. Mel Frequency Cepstral Coefficients (MFCC) - the cepstrum resulted from first warping the log energy spectrum according to the Mel frequency scale and then taking the cosine transform [1].

2. Perceptual Linear Prediction (PLP) - a variation of linear prediction by taking into account of human auditory perceptions [19].

Both MFCC and PLP are considered to be short-term locally stationary features and they can not cover the temporal dynamics in speech. In order to overcome the shortcoming, first-order and second-order time-derivatives of the static features are commonly used to capture temporal dynamics for speech recognition in practical use. Algorithms such as principal components analysis (PCA) [20], linear discriminant analysis (LDA or HLDA [21]), vocal tract length normalization (VTLN) [22], and independent component analysis (ICA) [23] are used to further transform the extracted features in order to improve ASR system performance. The ultimate goal of speech pre-processing is to produce as robust and discriminative features as possible to bridge the gap between the performance of ASR systems and that of human beings. More efforts need to be made in the ASR field to achieve this goal.

1.6 Statistical Acoustic Modeling

An acoustic model is used to describe the acoustic-phonetic characteristics of speech signals. Hidden Markov Models (HMMs) have been the dominant approach since they provide a principled way of jointly modeling speech spectral variations and time dynamics. In HMM, the speech production mechanism is treated as a stochastic process which generates the observed speech signals in a series of state transitions. If the probability of moving to the next state only depends on the identity of the current state, a first-order Markov process can be used to model the stochastic process. In speech recognition, a HMM is a stochastic finite state machine. Fig. 1.2 shows an example of HMM [1], where for each time frame, it has two options: either remains at the same state or changes to the next state. When a state *j* comes in at time *t*, the emitting probability distribution $b_j(O_t)$ generates an observation vector O_t. There are two special states in a HMM: an entry state S_0 and an exit state S_4. They are reached before the speech vector generation process begins and when the generation process terminates, respectively, and both states are reached only once. State S_0 and state S_4 do not have emitting probability densities since they do not generate any observation.

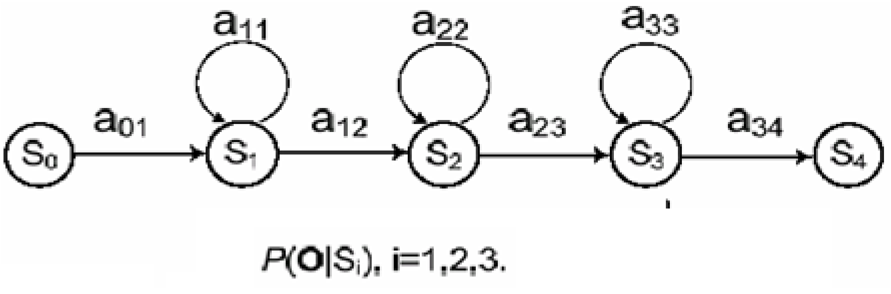

Fig. 1.2 An example of HMM for a phoneme [1]

The transition probability in a hidden Markov model a_{ij} is defined as the probability of entering state *j* given the previous state *i* [1], i.e.,

$$a_{ij} = P_r(s(t) = j)|s(t-1) = i) \tag{1.2}$$

where *s*(*t*) is the state index at time *t*.

The emitting probability density $b_j(o)$ defines the distribution of the observation vectors at the state *j*. Emitting probability density function in continuous density HMM (CDHMM) is often taken as a Gaussian Mixture Model (GMM) [1]:

$$b_j(o) = \sum_{n=1}^{M} w_{j,n} N(o; \mu_{jn}, \Sigma_{jn})$$

$$\sum_{n=1}^{M} w_{j,n} = 1 \text{ and } w_{j,n} \geq 0 \tag{1.3}$$

where $N(O; \mu_{jn}, \Sigma_{jn}) = \frac{1}{(2\pi)^{D/2} |\Sigma_{jn}|^{1/2}} e^{-\frac{1}{2}(o-\mu_{jn})^T \Sigma_{jn}{}^{-1}(o-\mu_{jn})}$ is a multivariate Gaussian density, D is the dimension of a feature vector, and w_{jn}, μ_{jn}, and Σ_{jn} are the weight, mean and covariance of the *n*-th Gaussian component of the GMM at state *j*.

As we discussed in Section 1.4, $P(O|W)$ represents the likelihood of an observation sequence O given word sequence W. Given $\{a_{ij}\}$ and $b_j(o)$, i =1~N, j = 1~N, it is computed as [1]:

$$P(O|W) = \sum_S P(O, S|W) \tag{1.4}$$

where $S = s_1, s_2, \ldots, s_T$ is the hidden Markov model state sequence that generates the observation vector sequence $O = o_1, o_2, \ldots, o_T$, and the joint probability of O and the state sequence S given W, $P(O, S|W)$, which is a product of the transition probabilities and the emitting probabilities, is defined as [1]:

$$P(O, S|W) = \prod_{t=1}^{T} b_{s_t}(O_t) a_{s_t s_{t+1}} \tag{1.5}$$

where s_{T+1} is the non-emitting exit state.

In LVCSR systems, sub-word units, such as syllables and phonemes, are used as the basic units for acoustic model training and recognition test since it is impractical to build a HMM for each word or word sequence. The model of a word string is constructed by concatenating the corresponding basic unit HMMs.

1.7 Phonetic Decision Tree

Speech is a very complex signal and its production can be affected by many variation factors. One of the most common variation factor is Co-articulation which means that the pronunciation of a phoneme can be affected by the articulations of neighboring phonemes. Context-dependent (CD) phonemic HMM is usually used to describe the co-articulation phenomena in continuous speech recognition and triphone CD HMMs have been used in speech recognition system. A triphone has a monophone as its center phone with only one left phone and one right phone as the context. Different triphones with the same center phone are called allophones [18]. For instance, *aw-iy+th* (the left context is *aw* and the right context is *th*), *m-iy+t, aw-iy+nx,* ……, are called allophones since they have the same center phone *iy*. Since the left and right neighbors in triphones are different combinations of other phones in the phone set, the total number of triphones is much larger than the commonly used 40 monophones in English. Therefore, for reliable parameter estimations of HMM models, training data always seem insufficient, especially when Gaussian mixture models (GMM) are used as HMM's output models. Phonetic decision tree (PDT) based clustering [22] is one of the most popular clustering algorithms used to reduce the number of physical triphones by tying physical triphone models for a large portion of logical triphone units (such as triphone state) .

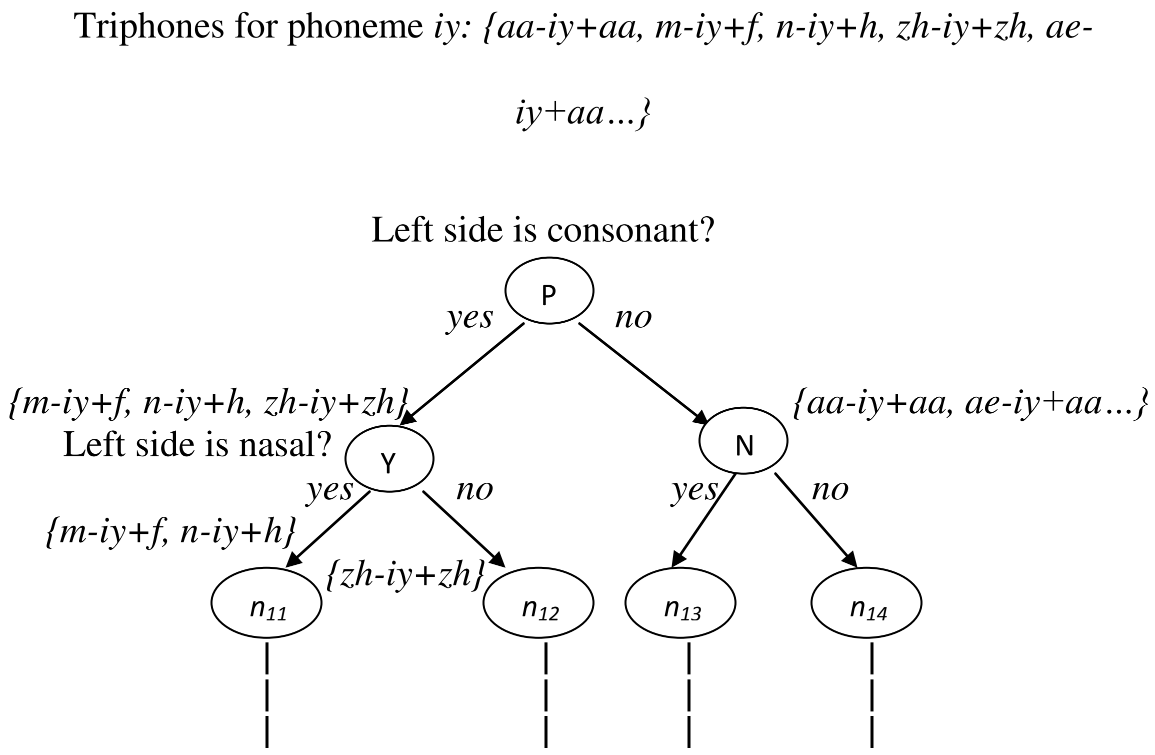

Fig. 1.3 An example of PDT for phoneme *iy*

In Fig. 1.3, an example of PDT is illustrated. From Fig 1.3, we notice that a phonetic decision tree is a binary tree with a yes/no phonetic question attached to each node. In PDT based state tying, from the context-dependent data of that phone, a decision tree is built for each phone state. All allophones of the phone state are tied and modeled by one Gaussian density at the root node of the tree. By asking a phonetic context question, the allophone set is split into two subsets at the node. The quality of each question can be measured by the likelihood increment due to the split, and the node split is determined by selecting the question that leads to the maximum likelihood gain [22]. This node split procedure is iterative. There are two criteria for stopping the iterative process. One is that the data count at a node falls under a predefined threshold, which ensures that all leaf nodes have enough data to estimate reliable GMMs. The other one is that the likelihood gain becomes smaller than a predefined threshold. In addition, leaf nodes with different parents could be merged if the likelihood loss due to the merging is less than the predefined likelihood gain threshold. This procedure of PDT construction is carried out in a top-down order until one of the two termination criteria as described above is met. Compared with other clustering methods such as k-means, PDT can greatly reduce the number of triphone models while effectively incorporating acoustic phonetic knowledge into the clustered models. In addition, it can also predict unseen triphone units which do not occur in training data.

1.8 Language Model

Given a sequence of previously spoken words, Language Model (LM) is used to provide the probability that the word w will be spoken next. There are different ways to solve this problem in the literature, such as Context-Free-Grammar (CFG) [24] and *N*-gram model [25]. CFG methods use knowledge-based rules to define the production of a sentence in words while *N*-gram LM uses counting-based occurrence probabilities to predict the occurrence of the next word. The advantage of the first method is that it is closer to grammar rules, which seems more reasonable than n-gram. However, its disadvantage is also very obvious, that is, in CFG, the knowledge-based rules are too complex to be represented by a grammar model. In fact, the second method is much more successful in practical use since *N*-gram based LM can be easily obtained and consistently integrated with acoustic model in a stochastic framework. An *N*-gram Language Model can be simply represented by $P(w_n|w_{n-N+1},\dots,w_{n-1})$, where $w_{n-N+1},\dots,w_{n-1}$ are *N*-1 words that appeared immediately before the current word w_n. The most commonly used *N*-grams are bigram (*N*=2) and trigram (*N*=3), respectively. If higher order *N*-grams such as

4-gram needs to be used in a speech research system [26], much more training data are required to estimate a reliable LM. Usually, speech transcriptions are not sufficient to estimate a reliable LM. Therefore, an *N*-gram based LM is often estimated using a large text corpus which should include words in the same domain as speech transcriptions. However, for those words which do not appear frequently through the whole corpus, there are smoothing techniques that can help predict the probabilities for their occurrences [27, 28]. One of the most commonly used smoothing techniques is Backing-off model [29] which uses lower-order *N*-grams to approximate the probabilities of those words which rarely appear in a training corpus. In the Backing-off model, an *N*-gram probability is expressed as [1]:

$$P(w_n|w_{n-N+1},\dots,w_{n-1}) = \begin{cases} \frac{Count(w_{n-N+1},\dots,w_{n-1},\ w_n)}{Count(w_{n-N+1},\dots,w_{n-1})} & (a) \\ \alpha(w_{n-N+1},\dots,w_{n-1})P(w_n|w_{n-N+2},\dots,w_{n-1}) & (b) \end{cases} \tag{1.6}$$

where (a) is used if $w_{n-N+1},\dots,w_{n-1},w_n$ has occurred enough times to estimate a reliable probability for the *N*-gram, otherwise, (b) is used. In addition, $\alpha(w_{n-N+1},\dots,w_{n-1})$ is the back-off coefficient of *N*-1-gram $(w_n|w_{n-N+2},\dots,w_{n-1})$, which makes the total probability mass of the *N*-gram equal to 1 (which needs to discount the large counts).

1.9 Viterbi Algorithm

The goal of speech recognition is to obtain the word sequence *W* for a feature sequence *O* so that the posteriori probability *P(W|O)* is maximized. Decoding engine, which is also called speech recognizer, is able to combine the statistical models or knowledge sources including acoustic models, language models, pronunciation models, dictionary, and decoding algorithms and use them during the search process to obtain the overall recognition results in a speech recognition system.

In all decoding algorithms, the most commonly used algorithm is Viterbi algorithm. Viterbi algorithm is based on the Dynamic Programming (DP) principle [30] which decomposes a problem into some sequential, independent sub-problems and obtains the solution for the original problem in a bottom-up manner by solving those sub-problems recursively. A decoding procedure using Viterbi algorithm can be divided into two steps: forward-extension and backtrace [1]. Suppose we have a speech utterance including *T* acoustic frame vectors. In the forward-extension step, all possible paths are extended from time 0 to time *T-1*. In order to speed up the decoding, different heuristic pruning [30] and look-ahead methods may be implemented in this step to cut off those search paths with low probability scores. By combining the acoustic scores and language scores of all acoustic vectors till the current frame, the path scores are accumulated. At each time, when a new word is created, each path records its best previous word. In the backtrace step, a best path is selected with the highest probability score once the last frame at *T-1* has been processed. A backtrack is implemented by recursively obtaining the best previous word recorded in the forward-extension step for the current word.

1.10 Lattice Generation

A word lattice is a compact, intermediate representation of alternative hypotheses for recognizing a speech utterance. A lattice can be generated by a HMM baseline and it is a directed acyclic graph which contains many paths in the search space. In a lattice, nodes are connected by arcs, and each arc is labeled with a word which is hypothesized between the time marks of its nodes as well as the likelihood that the word is uttered in that particular interval. Lattices generated with HTK [22] include time information associated with the start and end nodes of each arc. Therefore, lattices provide word boundary information. The likelihoods or arcs are obtained during the recognition process as a combination of the acoustic and language model probabilities. One method for efficiently constructing word lattices is to use lexical trees [31]. The use of word lattices has become very popular in large vocabulary speech recognition. The main advantage of using word lattices is to provide alternative word hypotheses in a constrained space for speech signals in order to allow using more elaborate knowledge sources to improve recognition accuracy without repeating the whole decoding process. One example is to apply a more complex language model to implement lattice rescoring based on lattices generated by a simpler language model and acoustic model.

1.11 Template based Speech Recognition

Template-based methods make comparisons between a test pattern and templates of training data, and therefore they are able to capture the speech dynamics and time correlation of speech frames better than HMM based methods. Template-based methods commonly use DTW to compare a test pattern against a template or template sequences. They were originally used to recognize isolated words or connected digits with good performance [1]. When using template-based methods, feature vectors of training set need to be stored in computer memory, and until recently, it has been practically impossible to apply template-based methods to large tasks of speech recognition. With today's rapid advance in computing power and memory, template-based methods have attracted new efforts and the reported results are promising [32, 33]. It has also been shown that by combining HMM-based and template-based recognition at the system level, accuracy performance can be improved over either system alone [16].

1.11.1 Dynamic Time Warping

Dynamic time warping (DTW) is used to calculate the shortest distance between two speech frame vector sequences based on some given constraints [34].

Suppose we have an input speech vector sequence $\boldsymbol{x}$ = $(x_1, . . . , x_N)$ and a template $\boldsymbol{y}$ = $(y_1, . . . , y_{N_y}\ )$. The distance between the two sequences of vectors is calculated as [35]:

$$D(\boldsymbol{x};\boldsymbol{y}) = \sum_{i=1}^{N_\phi} g(d\left(\boldsymbol{x}_{\phi_x(i)};\boldsymbol{y}_{\phi_y(i)}\right), \phi(i) - \phi(i-1)) \qquad (1.7)$$

In Eq(1.7), an alignment path is determined by $\{\phi(x)$ = $(\phi_x(i), \phi_y(i))$, i = 1 to $N_\phi\}$. At each point of the path, the local distance $d(x_{\phi_x(i)};\ y_{\phi_y(i)})$ is calculated. Some constraints on the path can be [35]:

$$\begin{aligned}&0 < \phi_x(i) \leq N\\&0 < \phi_y(i) \leq N_y\\&0 < N_\phi \leq N + N_y\\&\forall j \neq i: \phi(i) \neq\ \phi(j)\\&\phi_x(i)\geq \phi_x(i\text{-}1),\ \ \phi_y(i)\geq \phi_y(i\text{-}1)\end{aligned} \qquad (1.8)$$

Many additional local constraints have been proposed and used [34], and here we list two very important constraints. The first one is symmetric constraint [35]:

$$0 \leq \phi_x(i) - \phi_x(i-1)\ \leq 1 \text{ and } 0 \leq \phi_y(i) - \leq \phi_y(i-1)\ \leq 1 \qquad (1.9)$$

and the second one is Itakura constraint [35]:

$$\phi_x(i) - \phi_x(i-1) = 1$$
$$0 \le \phi_y(i) - \phi_y(i-1) \le 1 \text{ or } \phi_y(i) - \phi_y(i-2) = 2 \qquad (1.10)$$

The symmetric constraint does not allow skipping for both reference and test frames while the Itakura constraint allows skipping of one reference frame and stalling in any particular reference frame for two consecutive input frame vectors. Both of them can generate a proper alignment between a test segment and a reference template even though they have different lengths.

1.11.2 Local Distance

A speech template is a sequence of *N* acoustic frame vectors. For a template having a sequence of *N* *M*-dimensional feature vectors y_i, and it can be represented by $y^N = (y_1, y_2, \dots, y_N)$. The dissimilarity between two acoustic frame vectors is commonly calculated by the Euclidean distance [35]:

$$d(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})' I_M (\boldsymbol{x} - \boldsymbol{y}) = \sum_{i=1}^{M} (x_i - y_i)^2 \qquad (1.11)$$

with I_M the $M \times M$ identity matrix, and the Mahalanobis distance [35]

$$d_{mah}(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})' \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{y}) \qquad (1.12)$$

with $\boldsymbol{\Sigma}$ the covariance matrix of the distribution of the data.

1.12 Prosodic Information

Prosodic information of speech involves variations in phone or syllable length, loudness, and pitch of speech sounds. Loudness can also refer to the sound energy. Details of a language's prosody depend upon its phonology. For instance, in a language including vowel and consonant sounds, vowel sounds usually have longer duration than consonant sounds. In a similar manner, prosodic pitch must not obscure tones in a tonal language if speech is to be intelligible. Prosodic information is useful for automatic speech recognition (ASR) because it plays an important role in the comprehension of spoken language by human beings: it helps in recognizing spoken words, in resolving global and local ambiguities, and in processing discourse structure [36]. The role of prosody is particularly important in spontaneous speech. For example, acoustic differences between stressed and unstressed syllables are greater in spontaneous speech than in reading speech [37]. Since spontaneous speech contains lots of prosodic information [38], it has been hypothesized that prosodic information could be useful to improve speech recognition accuracy. However, prosody in speech recognition has not been studied well since prosodic information is ignored during the HMM based feature extraction process. Some promising work [39, 40] used prosodic information for improved duration modeling to control the search space. Vowel sound duration modeling was used to assess non-native speaker's English [41]. In addition, prosodic information was also used for cross-word context models [42] and language modeling [43, 44]. Finally, prosody features have been widely used in speech understanding and many applications have been investigated [45, 46].

The integration of prosodic information with ASR has been investigated in [47, 48, 49]. There are two ways to do so. The first one is to incorporate prosodic features as another stream at the segment level [49], which has the advantage that spectral and prosodic features are jointly modeled. However, the disadvantage of the method is that phenomena beyond the segment level cannot be captured. The second way is to build prosodic feature models which are independent of the ASR acoustic and language models. This method has the advantage that the models can be built at arbitrary linguistic levels and combined with the ASR hypotheses by lattice rescoring. In addition, there is no modification for the conventional ASR in order to include prosodic information. The method can also be extended to the combination of template matching and prosodic features. Therefore, in this dissertation, we take the same strategy as the second method to integrate template matching scores with prosodic scores.

Prosodic information has been ignored in the HMM based system and this kind of information is discarded during feature extraction. However, for template based approaches, prosodic information derived scores can be integrated with template matching scores which are based on frame spectral features, and this integration offers a possibility of obtaining a better recognition accuracy. There are also applications in integrating a template-based approach and prosody information [9] for connected digit recognition with positive results. Recently, the integration of template based methods and prosody information for large vocabulary speech recognition tasks has been explored in [50]. Since our template based methods have been successfully used for LVCSR [51], we investigate adding the prosody information to our template methods in order to further improve speech recognition accuracy.

Chapter 2

Template Construction, Matching, Clustering and Lattice Rescoring

2.1 Viterbi Alignment

In Section 1.9, we have introduced Viterbi algorithm which is based on Dynamic Programming. To generate the time boundaries for templates, we use the Viterbi algorithm to align speech utterances with their word transcriptions. The acoustic model used to do the alignment is from the HMM baseline model. The aligned results include the start frame, the end frame, triphone unit and log-likelihood alignment scores, which are illustrated below for a sentence fragment.

Start Frame	End Frame	Phone Unit	Alignment Score
0	2	sil	-168.038223
2	5	sil-k+ay	-245.223969
5	11	k-ay+n	-389.628296
11	15	ay-n+d	-235.218353
15	21	n-d+ah	-310.358765
21	22	sp	-69.052261
22	26	d-ah+v	-259.734985
26	31	ah-v+m	-356.604858
31	100	sp	-4271.944824
100	106	v-m+ay	-399.708344
106	116	m-ay+jh	-738.841736
116	117	sp	-80.015892
117	128	ay-jh+ao	-738.621521
128	146	jh-ao+z	-1201.498779
146	153	ao-z+aa	-483.028595
153	154	sp	-82.471916
154	158	z-aa+r	-245.659470
158	161	aa-r+g	-194.141525
⋮	⋮	⋮	⋮
281	283	sil	-133.543701

With the template boundaries obtained from the Viterbi alignment, we can proceed to construct templates.

2.2 Template Construction

We choose the template unit as context-dependent phone segments (triphone context). As discussed in Section 2.1, forced alignments of training speech data with their transcriptions are first carried out to obtain phone boundaries which define the context-dependent phone templates. We then use the GMM codebook which consists of the GMMs $\{m_1, m_2, \ldots, m_N\}$ of the phonetic decision tree tied triphone states in the baseline HMMs to label the template frames in the templates. To do so, we compute the likelihood scores of a frame x_t of a phone template by all GMMs and the GMMs that give the top n likelihood scores, $p(x_t|m_{1(t)}) \geq p(x_t|m_{2(t)}) \geq \cdots \geq p(x_t|m_{n(t)}) \geq \cdots$, are used to label x_t. Each GMM index is also associated with a weight w_i that is proportional to the likelihood score $p(x_t|m_{i(t)})$. A frame which is labeled by GMMs is therefore represented as:

$$x_t \rightarrow \left\{ \begin{bmatrix} m_{1(t)} \\ \vdots \\ m_{n(t)} \end{bmatrix} \begin{bmatrix} w_{1(t)} \\ \vdots \\ w_{n(t)} \end{bmatrix} \right\}$$

$$w_{i(t)} = \frac{p(x_t|m_{i(t)})}{\sum_{i=j}^{n} p(x_t|m_{j(t)})}, \qquad \sum_{i=1}^{n} w_{i(t)} = 1 \tag{2.1}$$

For a template X having n frame vectors, it is represented by $X = \{x_1, x_2, \ldots, x_n\}$ with each x_i $(i = 1, \ldots, n)$ a frame vector. Compared with the traditional real frame vector based templates, the templates constructed here are represented with multiple GMM indices associated with the corresponding weights. In conventional speech recognition, speech fames are represented by high-dimensional spectral feature vectors (usually 39-dimensional vectors are used) with the float data type. Our novel method of using the GMM indices to represent the templates converts the 39-dimensional float vectors to integer numbers, which greatly reduces the memory space requirement and makes the template based method possible to be extended to LVCSR.

2.3 Template Matching

DTW is widely used for measuring the dissimilarity between two speech utterances to cope with variations in speaking speed. DTW has different local path constraints as we introduced in Section 1.11.1. For template matching, we adopt a symmetric constraint defined as:

$$D(i,j) = d(i,j) + min\{D(i-1,j), D(i-1,j-1), D(i,j-1)\} \tag{2.2}$$

The advantage of the symmetric constraint is that it does not set any limitation on the lengths of two matching objects, which can guarantee a test segment to match with a template with any length. The distance between two feature vector sequences x and y is calculated as:

$$D(x,y) = \frac{L}{N} min_{\emptyset} \sum_{k=1}^{N} d(\emptyset_x(k), \emptyset_y(k)) \tag{2.3}$$

where *d* is the local distance between any two frame vectors in the two sequences,$\emptyset_x$ and $\emptyset_y$ are the functions that map *x* and *y* to the common time axis, *N* is the warping path length, and *L* is the length of the test feature vector sequence. It is noted that the defined *D(x, y)* equals to a product of the average frame distance and *L*, and it is therefore proportional to the test segment length.

2.4 Template Clustering

In HMMs, Phonetic Decision Tree (PDT) is commonly used for triphone state tying. Although this method can be extended for tying whole triphone templates in our task, it is also plausible to utilize the PDT tying structures of the states of phone HMMs directly for triphone template tying, since the tying structure of a phone state indicates partial similarities among triphone segments. Specifically, for the triphone templates of each monophone, we keep three tying structures defined by the three emitting states of the corresponding phone HMMs, and the multiple tying results are jointly used in template matching.

In matching a test speech segment with a triphone unit, we select the *n*-best templates from the corresponding tied triphone cluster that are closest to the test segment and use their average score as the match score. More details about how to select the *n*-best templates will be explained in Chapter 5.

2.5 Lattice Rescoring

Lattices can be generated by a HMM baseline system. A lattice contains many hypotheses. By setting different numbers of tokens [22], different sized lattices can be generated. Usually, a larger lattice can include more correct hypotheses but at the same time it also brings more confusion. An example of word lattice is shown in Fig 2.1. As we already introduced in Section 1.10, in a lattice, there is a starting node and an ending node which represents the beginning and the ending of an utterance, respectively. In addition, nodes in a lattice provide word boundary information and arcs between nodes represent words with the associated scores (Scores are not shown in the figure) which are the combined scores from the acoustic and language models. In Fig.2.1, the utterance starts at frame 0 and ends at frame 51. The highlighted path will be picked up as the speech recognition output since the sum of word hypothesis scores along the path is the highest. The advantage of using lattice rescoring is that lattices can provide hypothesized phone boundary information and template matching can be directly performed on the phone arcs in each lattice without the re-decoding process. When using template matching for lattice rescoring, the scores changed by rescoring are only the acoustic scores and the scores from the language model remain unchanged.

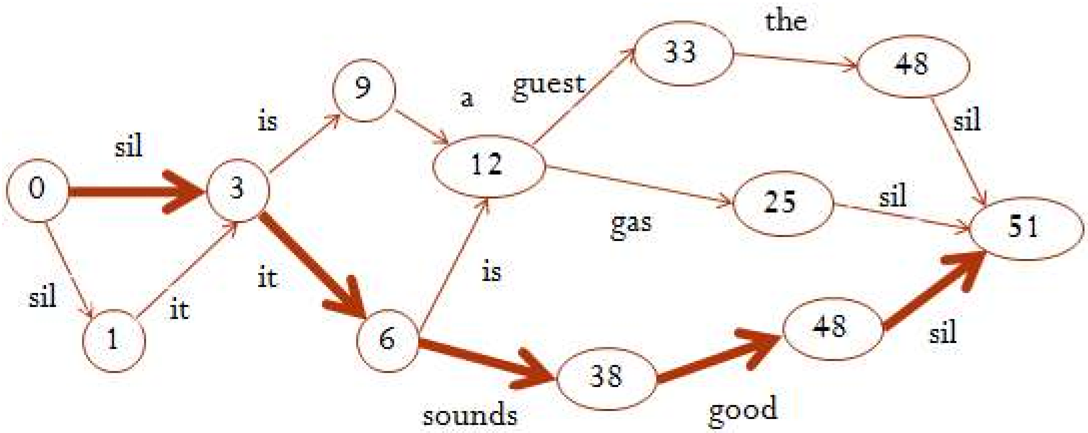

Fig. 2.1 An example of a word lattice

Chapter 3

Local Distances

3.1 Euclidean Local Distance

The commonly used local distance in DTW for matching templates and test segments is Euclidean distance [10] since it is easily used to calculate the distance between two high dimensional frame vectors. We use the Euclidian distance as the local distance in DTW as a baseline of template matching in order to compare the recognition performance between the traditional template matching method and the proposed template matching method in this dissertation.

3.2 Negative Log-likelihood Local Distance

In the proposed template matching method, templates are represented by GMM indices where the traditional local distance for two vectors such as the Euclidean distance can not be used any more. So here we first use negated log likelihood score to calculate the local distance between two frame vectors.

Let each speech frame be labeled by its 1-best GMM index. Suppose we have a test phone segment $n_1\{ f_1: m_1 , f_2: m_2 , f_3: m_3\}$ consisting of three frame vectors f_1, f_2, and f_3 that are indexed by m_1, m_2, and m_3, respectively, and a phone template $n_2\{ k_1: m_4, k_2: m_5 , k_3: m_6\}$ consisting of three frame vectors k_1, k_2, and k_3 indexed by m_4, m_5, and m_6, respectively. Assuming that the local distance $d(f_1, k_1)$ needs to be calculated. The negative log likelihood local distance between f_1 and k_1 is defined as:

$$d(f_1, k_1) = -\log p(f_1|m_4) \tag{3.1}$$

It is worth noting that the negative log likelihood distance directly changes the similarity measure of log likelihood into a dissimilarity measure.

3.3 Log-likelihood Ratio based Local Distance

In the negative log-likelihood local distance, we only use the test frame vectors as the information from the test segments. Since test frame vectors are also labeled by GMM indices as we do for template construction, we can also include GMM indices which are used to label the test frame vectors in the local distance measurement. So here we propose a novel log likelihood ratio measure to calculate the local distance between two frame vectors. Suppose that a test phone segment $n_1\{ f_1: m_1 , f_2: m_2 , f_3: m_3\}$ consists of three frame vectors f_1, f_2, and f_3 that are indexed by m_1, m_2, and m_3, respectively, and a phone template $n_2\{ k_1: m_4, k_2: m_5 , k_3: m_6\}$ consists of three frame vectors k_1, k_2, and k_3 indexed by m_4, m_5, and m_6, respectively. The log-likelihood local distance $d(f_1, k_1)$ between f_1 and k_1 is:

$$d(f_1, k_1) = log\frac{p(f_1|m_1)}{p(f_1|m_4)} \tag{3.2}$$

The log likelihood ratio measure contrasts the fit scores of a test frame vector with its best model against its fit scores with the best model of the template vector, and therefore compares the two frame vectors indirectly through the models. The log likelihood ratio measure is nonnegative when 1-best GMM is used in frame indexing. When using multiple GMM indices for speech frame representation, however, this property is occasionally violated when a test frame vector is very close to the template vector. In the sense of indirectly measuring the frame distance, we simply take the absolute value for the log likelihood ratio. Suppose f_1 is represented by top 2 GMM indices m_1 and m_2 with weights w_{11} and w_{12}, and likewise k_1 is represented by GMM indices m_4 and m_5 with weights w_{21} and w_{22}. The likelihood scores S_1 and S_2 are calculated as:

$$S_1 = w_{11}p(f_1|m_1) + w_{12}p(f_1|m_2)$$

$$S_2 = w_{21}p(f_1|m_4) + w_{22}p(f_1|m_5) \tag{3.3}$$

where

$$w_{11} = \frac{p(f_1|m_1)}{p(f_1|m_1)+p(f_1|m_2)}\,, \qquad w_{12} = 1 - w_{11}$$

$$w_{21} = \frac{p(k_1|m_4)}{p(k_1|m_4)+p(k_1|m_5)}\,, \qquad w_{22} = 1 - w_{21} \tag{3.4}$$

As we discussed in Section 3.2, the negative log likelihood local distance between f_1 and k_1 is defined as

$$d(f_1, k_1) = -\,log\,S_2 \tag{3.5}$$

and the log likelihood ratio local distance between f_1 and k_1 is defined as:

$$d(f_1, k_1) = \left|log\,\frac{S_1}{S_2}\right| \tag{3.6}$$

3.4 Kullback–Leibler Divergence based Local Distance

In the negative log-likelihood and log-likelihood ratio local distances, except for using the GMM information for the templates and test segments, the real frame vectors are also used in both distances to compute the likelihood scores. Since both templates and test segments are labeled by GMM indices, we can also consider measuring the distance between GMMs without using the real frame vectors. Kullback–Leibler (KL) Divergence is an effective way to measure the distance between two GMMs [52]. So we can use the KL distance between GMMs to measure the dissimilarity of two frame vectors. Since there is no closed form expression for KL distance of GMMs, we use the Monte Carlo sampling method of [52] to compute the distance from a GMM m_x to a GMM m_y as:

$$d\left(m_x \,||\, m_y\right) = \frac{1}{n}\sum_{i=1}^{n} log\frac{m_x(x_i)}{m_y(x_i)} \tag{3.7}$$

where x_i's are i.i.d. samples generated from the GMM m_x. Since the KL divergence is asymmetric, we define the KL distance between m_x and $m_y$as:

$$d_{KL}\left(m_x, m_y\right) = \frac{1}{2}\left(d\left(m_x \,||\, m_y\right) + d\left(m_y \,||\, m_x\right)\right) \tag{3.8}$$

The local distance between two frame vectors x_t and $y_{t'}$ is calculated as:

$$d(x_t\,, y_{t'}) = \sum_{i=1}^{n_{x_t}} \sum_{j=1}^{n_{y_{t'}}} \left(w_{i(t)} w_{j(t')} d_{KL}\left(m_{i(t)}, m_{j(t')}\right)\right) \tag{3.9}$$

where $n_{x_t}$and $n_{y_{t'}}$ are the numbers of GMMs used to label x_t and $y_{t'}$, respectively.

We already introduced and discussed the necessary concepts and components for the method of integrating statistical modeling with template matching. In Fig. 3.1, a block diagram of the overall method is given.

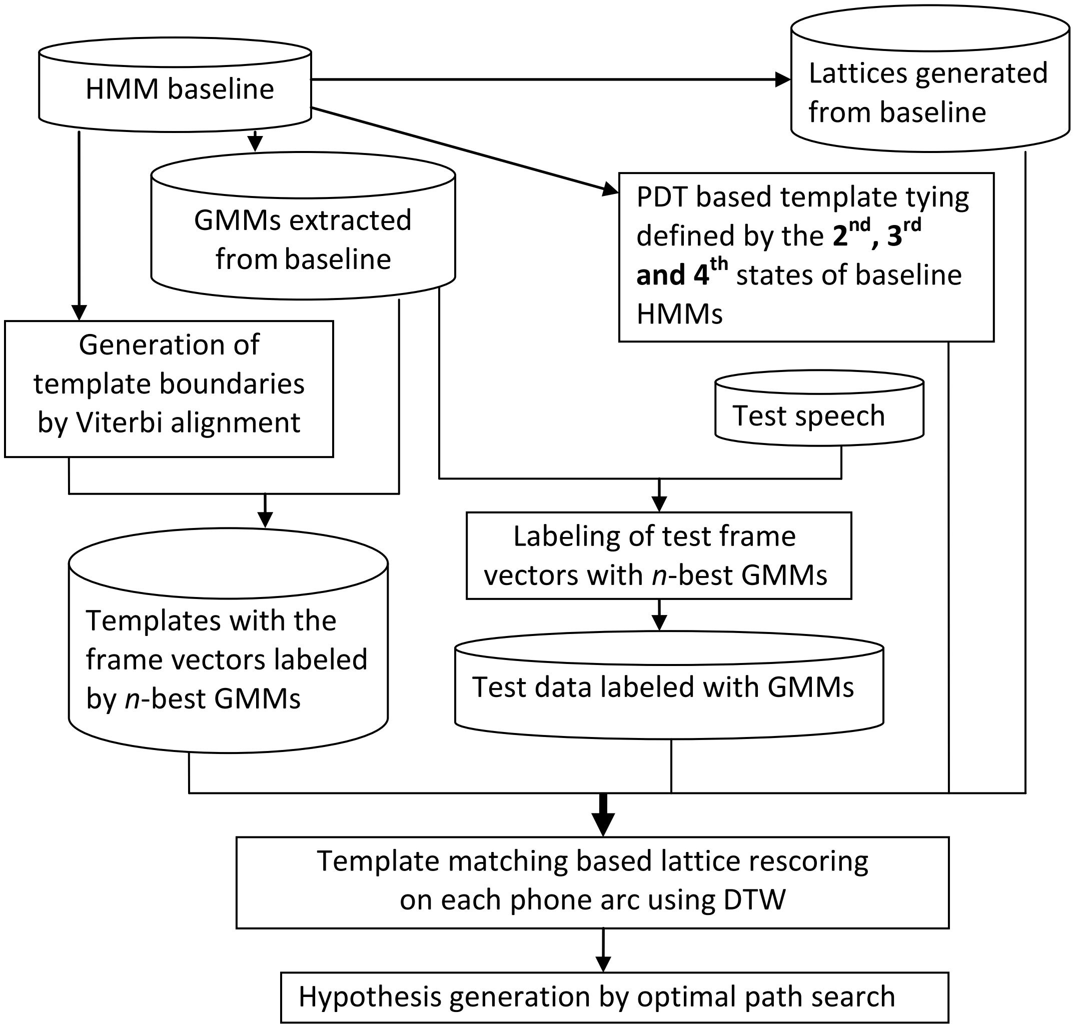
HMM baseline
Lattices generated from baseline
GMMs extracted from baseline
PDT based template tying defined by the **2nd, 3rd and 4th** states of baseline HMMs
Generation of template boundaries by Viterbi alignment
Test speech
Labeling of test frame vectors with *n*-best GMMs
Templates with the frame vectors labeled by *n*-best GMMs
Test data labeled with GMMs
Template matching based lattice rescoring on each phone arc using DTW
Hypothesis generation by optimal path search

Fig. 3.1: Block diagram of integrating statistical modeling with template matching

Chapter 4

Template Selection and Compression

Even though using GMMs to label frame vectors can save computation time and storage space significantly compared with the traditional template matching method, when the method is used for large vocabulary speech recognition, the cost of computation and storage is still too high. In order to further reduce recognition time and storage space, we propose template selection and compression algorithms. Before we get into the details of these algorithms, the commonly used hierarchical agglomerative clustering algorithm [20] which is used to agglomerate templates into different clusters is first described.

4.1 Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering is a bottom-up algorithm which at the beginning treats each template as a singleton cluster and then successively merge (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all templates. Given a distance function $D(C_i, C_j)$ for two clusters, the following procedure describes the clustering algorithm for *m* templates $\{x_1, x_2, \ldots, x_m\}$:

1. Initialize the template set $Z_1=\{\{x_1\}, \{x_2\},\ldots,\{x_m\}\}$ with each template being a cluster.
2. For $n = 2,\ldots,m$:

   Obtain the new set Z_n by merging two clusters C_i and C_j in the set Z_{n-1} with the minimum distance $D(C_i, C_j)$ among all existing distinct cluster pairs. Stop the clustering process if the number of clusters in the set Z_n drops below a threshold.

The cluster distance function *D(C_i, C_j)* is commonly defined by the distance of their elements *D(x_i, x_j)*. The average distance measure [20]

$$D\left(C_i, C_j\right) = \frac{1}{|C_i||C_j|} \sum_{x_i \epsilon C_i} \sum_{x_j \epsilon C_j} D\left(x_i, x_j\right) \tag{4.1}$$

is adopted here. Note that *D(x_i, x_j)* is the DTW distance of two templates defined in Section 2.3, and in this step of template clustering, the local distance d is the Euclidean distance of two speech feature vectors (not represented by GMM indices).

4.2 Minimum Distance based Template Selection

The hierarchical agglomerative clustering algorithm divides the templates in each tied triphone cluster into smaller clusters. For each such smaller cluster, a template representative is selected to represent all templates in the same cluster. Only the template representatives are stored and used in recognition. We choose to use a minimum distance based criterion to produce the template representatives, which calculates the minimum average distance from a template to all other templates in the cluster to select cluster template representatives. We call the template selection algorithm the minimum distance template selection (MDTS) algorithm. The template-to-cluster distance is defined as [20]:

$$D(x_i, C_i) = \frac{1}{|C_i|-1} \sum_{\substack{x_j \epsilon C_i \\ x_i \neq x_j}} D\left(x_i, x_j\right) \tag{4.2}$$

The template x_i^* is selected as the cluster template representative when it satisfies the following condition:

$$D(x_i^*, C_i) = \min_{x_i \epsilon C_i} D(x_i, C_i) \tag{4.3}$$

The frames of the selected template representatives are subsequently indexed by their *n*-best GMMs. Since we use a selected template representative to represent all other templates, only the template representative is used in recognition, which can further save the recognition time and memory space for the template storage.

4.3 Maximum Likelihood based Template Selection

In the algorithm of MDTS, we use the GMMs to label the selected templates as the template representatives which are then used in speech recognition. However, the template representatives generated by the MDTS method produced very poor recognition performance when the KL divergence local distance (described in Section 3.4) was used. In order to match better with the KL divergence local distance, we further propose a maximum likelihood based template selection method, and we call it the maximum likelihood template selection (MLTS) algorithm. In MLTS, based on the cluster center s^* which is selected by MDST, we relabel s^* by using a set of GMMs and the selection of GMMs follows a maximum likelihood criterion. We use DTW to align all other templates in the cluster C_i to the initialized template center s^* and the local distance used is the Euclidean distance. Fig. 4.1 illustrates an alignment result among the sequences $s^*, s_1, \dots, s_N$ in C_i. The frame vectors $x_{t_1}, \dots, x_{t_N}$ from the sequences $s_1, \dots, s_N$, respectively, are aligned to the frame vector x_t of the cluster center s*. The following procedure describes the MLTS algorithm using the aligned frame vectors $X = \{x_t, x_{t_1}, \dots, x_{t_N}\}$:

1. Pool the distinct GMMs which are used to label all the frames in X into a local GMM set M.
2. Use the k-medoids algorithm with the KL distance to partition the GMM set M into l clusters M_i, $i = 1, \dots, l$.

3. For $i = 1, \dots, l$:

Use the maximum likelihood criterion to select a GMM cluster center M_i^* for M_i:

$$M_i^* = \operatorname{argmax}_{M_i^j \epsilon M_i} \left(\sum_{x \epsilon X_{M_i}} lnp\left(x \middle| M_i^j\right) \right) \quad (4.4)$$

where M_i^j is a GMM in M_i, $X_{M_i} \subset X$ includes all frame vectors labeled by the GMMs in M_i.

4. For $i = 1, \dots, l$:

Calculate the weight w_i for each GMM cluster center M_i^*, which is proportional to the total likelihood of X evaluated by M_i^*, i.e.,$p(X|M_i^*)$.

$$w_i = \frac{p(X|M_i^*)}{\sum_{k=1}^{l} p(X|M_k^*)} = \frac{e^{\sum_{x \epsilon X} ln\, p(x|M_i^*)}}{\sum_{k=1}^{l} e^{\sum_{x \epsilon X} ln\, p(x|M_k^*)}} \quad (4.5)$$

The *k*-medoids algorithm is a clustering algorithm which attempts to minimize the distance from all other points in the cluster to the medoid. A medoid is a cluster center whose average distance to all the points in the cluster is minimal, and itself is also a point in the cluster.

The ML algorithm for generating the set of GMMs to relabel the frame vector x_t of s^* is applied to all frame vectors in $s*$. The frame of the template representative corresponding to the aligned frame vectors X is represented by M_i^* and w_i $(i = 1, \dots, l)$, and the representation has the same form as the templates in Section 2.2 with the difference that we use top *n* GMMs to label a frame vector in Section 2.2, and here the aligned multiple frame vectors are used to select and generate a set of GMM indices for the frame labeling. The selected GMMs represent better the frame vectors in each cluster and are thus more informative as the template representatives.

$$\begin{tikzcd}
s_1:\dots\ x_{t_1}: \left\{ \begin{bmatrix} m_{1(t_1)} \\ \vdots \\ m_{n(t_1)} \end{bmatrix} \begin{bmatrix} w_{1(t_1)} \\ \vdots \\ w_{n(t_1)} \end{bmatrix} \right\} \dots \arrow[d, "\vdots"'] \\
\text{Center } s^*:\dots\ x_t: \left\{ \begin{bmatrix} m_{1(t)} \\ \vdots \\ m_{n(t)} \end{bmatrix} \begin{bmatrix} w_{1(t)} \\ \vdots \\ w_{n(t)} \end{bmatrix} \right\} \dots \\
s_N:\dots\ x_{t_N}: \left\{ \begin{bmatrix} m_{1(t_N)} \\ \vdots \\ m_{n(t_N)} \end{bmatrix} \begin{bmatrix} w_{1(t_N)} \\ \vdots \\ w_{n(t_N)} \end{bmatrix} \right\} \dots \arrow[u, "\vdots"']
\end{tikzcd}$$

Fig. 4.1: Align the sequences $s_1, \dots, s_N$ to s^*

4.4 Template Compression

In MLTS, we discard all other GMMs if the cluster center M_i^* is selected. In template compression of this section, instead of keeping a cluster center GMM M_i^* and excluding all other GMMs from a cluster in labeling a template representative frame, we further merge the GMMs in each cluster M_i to include more information of the original templates into a compressed template representative. To remove the effect of outliers, we calculate the mean distance $\bar{d}$ and standard deviation σ between all other GMMs M_i^j to the cluster center M_i^* from the distance d_i^j which is the KL distance between a GMM M_i^j and the cluster center M_i^*. An outlier GMM M_i^j which is *t* times standard deviation away from $\bar{d}$, i.e.,

$$|d_i^j - \bar{d}| > t\sigma \tag{4.6}$$

is removed. Suppose that there are n_G GMMs left in M_i after removing the GMM outliers. We first pool all Gaussian components from the GMMs together, and we then normalize the weight of each Gaussian component with n_G. Two Gaussian components f_1 and f_2 are merged if the entropy increase due to the merge is the smallest. The entropy increase is calculated as [53]:

$$\Delta E(f_1\,,\,f_2) = log|\Sigma| - \frac{w_1}{w_1+w_2} log|\Sigma_1| - \frac{w_2}{w_1+w_2} log|\Sigma_2| \tag{4.7}$$

where w_1 and w_2 are the normalized mixture weights for f_1 and f_2 ,Σ_1 and $\Sigma_2$are the diagonal covariance matrices of f_1 and f_2 , and Σ is the diagonal covariance matrix of the Gaussian density generated by merging f_1 and f_2 . The mean μ, covariance Σ, and mixture weight w of the newly generated Gaussian component are defined as [53]:

$$\Sigma = \frac{w_1}{w_1+w_2}\Sigma_1 + \frac{w_2}{w_1+w_2}\Sigma_2 + \frac{w_1 w_2}{(w_1+w_2)^2}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$\mu = \frac{w_1}{w_1+w_2}\mu_1 + \frac{w_2}{w_1+w_2}\mu_2$$

$$w = w_1 + w_2 \tag{4.8}$$

We continue merging the Gaussian components until the number of Gaussian components in M_i is below a preset threshold. The remaining Gaussian components construct a new GMM, and the new GMM is used as a member of the GMM set to label a frame vector of the template representative.

Chapter 5

Experiments and Analysis

5.1 Experimental Setup

5.1.1 TIMIT Corpus and Experiment Setup

TIMIT is a corpus of phonemically and lexically transcribed speech of American English speakers of different genders and dialects. It was designed to further acoustic-phonetic knowledge and automatic speech recognition systems. The TIMIT training set consisted of 3696 sentences from 462 speakers and the standard test set included 1344 sentences spoken by 168 speakers. For the TIMIT dataset, a set of 39 phones units was obtained by reducing the original phone set of 61 phones, and a phone bi-gram language model (LM) was used. Speech frame feature consisted of 39 components: 13 MFCCs and their 1st and 2nd order time derivatives. The HMM baseline was trained with the GMM mixture sizes of 24. From the baseline HMMs, 1189 GMMs were extracted for template generation and matching. Phone lattices were generated for each test sentence. The total number of triphone templates was 152715. To calculate a KL distance between two GMMs, 10000 Monte Carlo simulation data samples were generated [52]. Crossword triphone models were trained and used in decoding search.

5.1.2 Large Vocabulary Speech Corpus and Experiment Setup

The large vocabulary speech recognition task is based on the telehealth automatic captioning system developed in the Spoken Language and Information Processing Laboratory at the University of Missouri-Columbia. The objective of this project is to develop an online captioning system to help patients with hearing impairments communicate with doctors in teleconferencing. Spontaneous speech data from 5 doctors were included and the vocabulary size was 46K. The challenges in developing such a system include several ways such as data collection, preprocessing of speech data, acoustic and language modeling, etc. A summary of the Telemedicine corpus is given in Table 5.1 [54]. The training and test data sets consist of 5 doctors' speech. Word counts from transcription texts are also listed. A total of 52 acoustic sound units were defined, including 42 speech monophone units, seven filled pause units, one unit for sound artifacts like lip smack and microphone ruffling, as well as one pause and one silence unit. For more details about the system, please refer to [55].

Table 5.1 Datasets used in the telehealth task: speech (min.)/text (no. of words)

	Training set	Test set
Dr. 1	180/39.148	27.8/6421
Dr. 2	250/44,967	12.1/3988
Dr. 3	210/35,348	29.8/5085
Dr. 4	145/28,700	19.3/3248
Dr. 5	200/39.398	14.3/2759
Total	985/187,561	103.3/21501

Baseline acoustic model parameters were trained by using the HTK toolkit [56] where HMM states were tied by phonetic decision trees. Speech features consisted of 39 dimensional frame vectors: 13 MFCCs base features and their 1st and 2nd order time derivatives. Short-time analysis window size was 20 ms and shift was 10 ms. Context-dependent crossword triphone models were used. The average number of triphone templates was 181601 per speaker for the 5 doctors. On average, 1905 GMMs were extracted from the baseline HMMs for each of the 5 doctors for template generation. Word lattices were generated for each test sentence with phone boundaries by using HTK. 10000 Monte Carlo simulation data samples were generated to calculate a KL distance between two GMMs [52].

In order to enlarge vocabulary coverage and improve events estimation, the language model (LM) was trained using the transcripts of the training speech as well as textual data from other domains including public domain data sets of Broadcast News, Switchboard, and Call Home, and medical domain data including telehealth related dataset on dermatology. Trigram LM using the SRI toolkit [57] was trained with Kneser-Ney backoff [58]. Four word trigrams LMs were trained for the out-of-domain datasets, and two class trigram LMs were trained for the in-domain datasets. These six LMs were linearly interpolated by using a ten-fold validation on telehealth training set. Details of language modeling for telehealth captioning are described in [59].

5.1.3 Lattice Rescoring

For the phone recognition task, a phone lattice was generated for each test sentence, and for large vocabulary speech recognition, a word lattice was generated for each test sentence with phone boundaries, both by using HTK [56]. Since lattices provided hypothesized phone boundaries, template matching was directly performed on the phone arcs in each lattice. Three tying structures defined by the three emitting states of the corresponding phone HMMs for the triphone templates of each monophone were kept, and the multiple tying results were jointly used in template matching. In matching a test speech segment with a tied triphone template, we selected the n-best templates from the corresponding tied triphone cluster that are closest to the test segment and used. The average matching score is calculated for on the *n*-best templates. The whole process is illustrated in Fig. 5.1 in which the square root of n_i (i=*1, 2,* and *3*) is the number of templates used in the template matching score calculation.

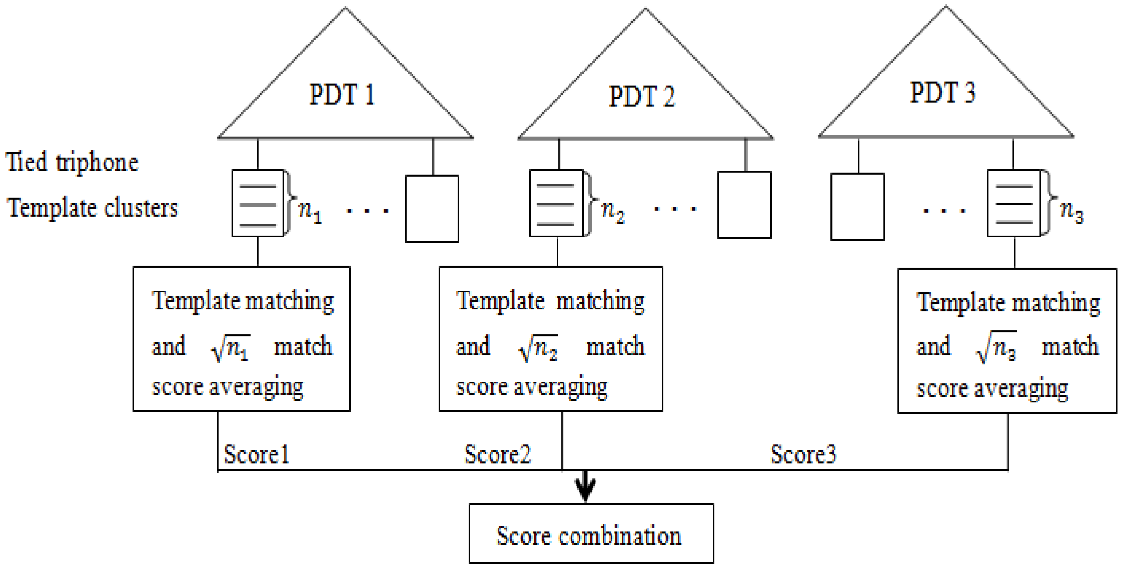
PDT 1
PDT 2
PDT 3
Tied triphone
Template clusters
n_1
n_2
n_3
. . .
Template matching and $\sqrt{n_1}$ match score averaging
Template matching and $\sqrt{n_2}$ match score averaging
Template matching and $\sqrt{n_3}$ match score averaging
Score1
Score2
Score3
Score combination

Fig. 5.1: Using PDT clustering structures inherited from a baseline phone HMM to calculate the template matching score

5.2 TIMIT Phone Recognition

5.2.1 Evaluation on Different Local Distances

We first investigate the recognition performance of different local distances discussed in Chapter 3. In Fig. 5.2, we compare phone recognition performances by using the HMM baseline and the four different local distance measures in DTW: 1) HMM baseline; 2) Euclidean local distance between two frame vectors; 3) negative log likelihood (NLL) local distance; 4) log likelihood ratio (LLR) local distance [60]; 5) KL-divergence local distance. Except for the cases 1) and 2) which did not use GMM indeces, each frame vector for the cases 3) to 5) was labeled by 5 GMM indices. The HMM baseline used 24-component GMMs with the recognition accuracy of 72.72%. The Euclidean and NLL local distances obtained the recognition accuracies of 72.83% and 72.92%, respectively, and they did not make significant improvements over the HMM baseline. However, the LLR and KL divergence local distances achieved 1.79% and 1.54% improvements over the HMM baseline. We conducted student *T* tests on the performance differences between the proposed methods based on the LLR and KL divergence local distances and the HMM baseline method and conclude that our proposed template matching methods based on the LLR and KL divergence local distances improved TIMIT phone recognition accuracy significantly over the HMM baseline at the significant level of α=0.05. We also examined the recognition result without using the language model and found that the LLR local distance based template matching method got a big improvement for semivowels /w/, /l/, /r/, and /y/, and also obtained an obvious improvement for some consonant sound like /b/,/dh/, /p/,/n/ and /k/, and the KL local distance based template method improved vowel sounds such as /ah/, /ey/, and /ow/ significantly, and also got better recognition accuracy for several consonant sounds like /b/, /d/, and /g/. Since the LLR and KL divergence local distances made significant improvements over the HMM baseline, we only used these two local distances in the subsequent experiments.

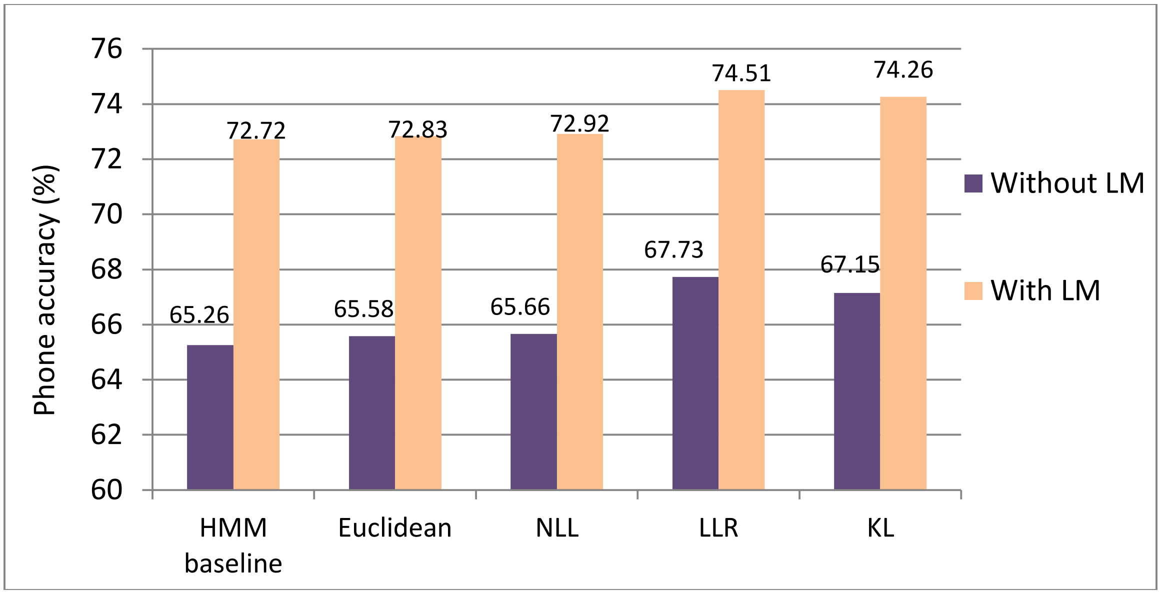

Fig. 5.2: A comparison of phone accuracy (%) by using the HMM baseline and different local distance measures for DTW in lattice rescoring.

5.2.2 Evaluation on Different Numbers of GMM for Labeling a Frame Vector

In this section, we investigate how the number of GMMs for labeling a frame vector affected the recognition performance. The local distances used here were LLR and KL divergence. In Figure 5.3, we compare using different numbers of GMM indices 1, 3, 5, and 7 to label the frame vectors. The HMM baseline accuracy of 72.72% (24-component GMMs) is shown across the four cases for comparison. For both the LLR and the KL divergence local distances, recognition accuracies with or without the language model peaked when the top 5 GMMs were used. The results verified that multiple GMM indices represented frame vectors better than that of a single GMM. We also notice that the LLR local distance gave a better recognition performance than the KL divergence local distance did for all four cases. In the subsequent experiments, we used top 5 GMMs to label each frame vector.

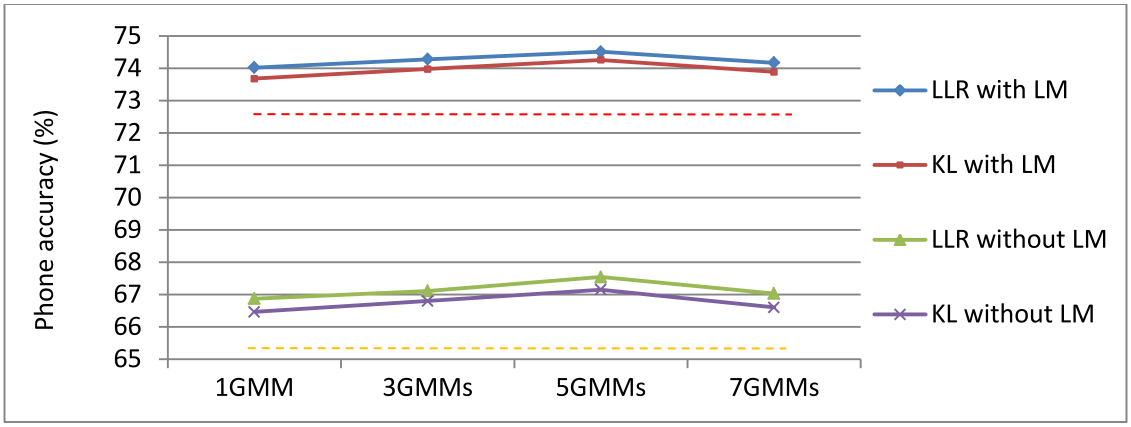

Fig. 5.3: A comparison of phone accuracy (%) by using different numbers of GMM indices for template representation with the LLR and KL divergence local distances in lattice rescoring (The red dashed line is the baseline with LM and the yellow dashed line is the baseline without LM).

5.2.3 Evaluation on the Templates Taken as the *N*-best Templates

As we discussed in Section 5.1.3, only *n*-best templates were selected to calculate the final scores for lattice rescoring even though all templates were used to match with the test segments. Therefore, we need to find the *n*-best templates from each tied triphone class for a test segment. There are different ways to decide *n*. Here we first define *n* as a fixed percentage of the tied triphone class. In Fig.5.4, we demonstrate how the percentages of the selected *n*-best templates affected the recognition performance based on the LLR and KL divergence local distances. When 3% best templates were used for calculating the final scores, the LLR local distance obtained its best performances of 67.48% and 74.38% for the cases of acoustic model alone and acoustic model plus language model, respectively. However, for the KL divergence local distance, the best performances of 67.03% and 74.18% for acoustic model and acoustic model plus language model were achieved respectively when 5% best templates were used for the computation of final scores in lattice rescoring.

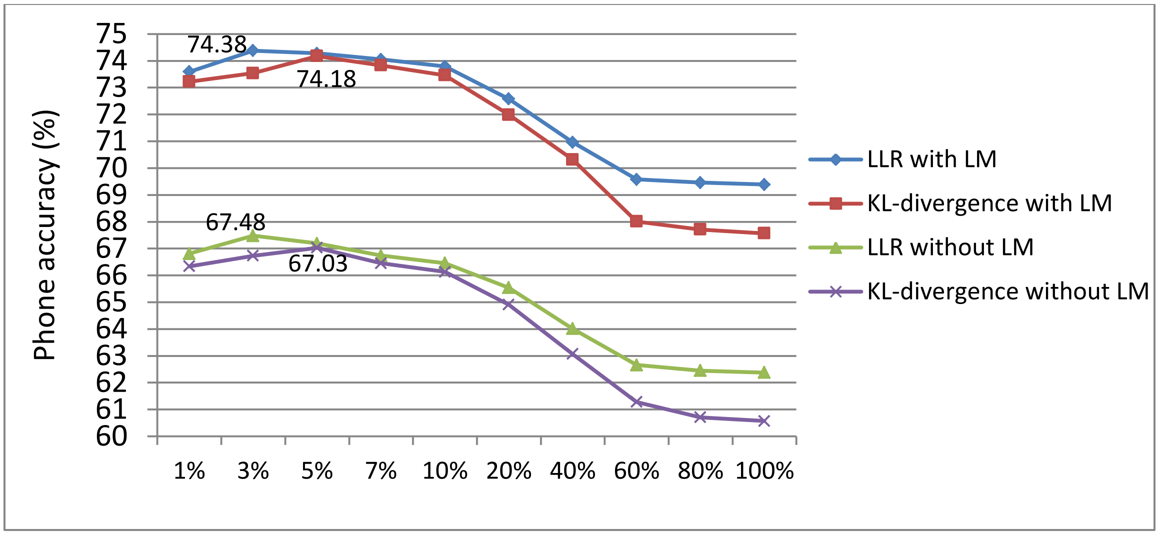

Fig. 5.4: A comparison of phone accuracy (%) by using different percentages of templates for final score calculation in lattice rescoring

We also decided the value of *n* by an empirical method in which *n* was taken as the square root of the number of all templates in each tied triphone class in analogy to the *K*-nearest neighbor (KNN) method of [61] where *K* was decided by the square root of training sample size which helps reduce the variance of the *n*-best numbers due to unbalanced data distribution. By using this method and with the LLR local distance, we obtained the recognition performances of 67.73% and 74.51% for acoustic model alone and for acoustic model plus language model, respectively. For the KL divergence local distance, the recognition performances of 67.15% and 74.26% for acoustic model alone and for acoustic model plus language model were achieved, respectively. Since the empirical method of taking *n* as the square root of all templates in each tied triphone class obtained better recognition performances than the method of taking *n* as a fixed percentage such as 3%, we use the empirical method to decide the value of *n* in all the subsequent experiments.

5.2.4 Evaluation on Template Selection and Compression Algorithms based on Different Local Distances

In this section, we investigate the effects of the template selection and compression algorithms as discussed in Chapter 4. In Fig. 5.5, we compare the recognition performance for the template selection and compression algorithms based on the LLR and KL divergence based local distances. Five cases are shown: 1) HMM baseline; 2) use all templates (without selection) based on the LLR and KL divergence local distances; 3) use the minimum distance template representative selection algorithm based on the LLR and KL divergence local distances; 4) use the maximum likelihood template representative selection algorithm based on the LLR and KL divergence local distances; 5) use the template compression algorithm based on the LLR and KL divergence local distances. The HMM baseline accuracy was 72.72% (24-component GMMs).

When all templates were used (without selection and compression), the LLR based local distance obtained better recognition performance of 74.51% than the KL divergence based local distance did (74.26%). The minimum distance template selection algorithm had the recognition accuracy of 73.79% when the LLR local distance was used, and when the KL local distance was used, it only increased the recognition performance by 0.03% over the HMM baseline. The maximum likelihood template selection algorithm got the recognition accuracy of 74.19% when the KL local distance was used, while when using the LLR local distance, it only improved recognition accuracy by 0.46% over the HMM baseline. Based on the maximum likelihood template selection algorithm, we further conducted an experiment on the template compression method based on the LLR and the KL divergence local distances. The template compression made further improvement

(74.35%) over the MLTS method based on the KL local divergence distance while it decreased the phone recognition accuracy by 0.43% from the MLTS method when LLR local distance was used. The number of GMMs used to label frame vectors for cases 2) to 5) was 5. The threshold *t* in equation (4.6) for removing the GMM outliers was set to 2. The percentages of template representatives taken in MDTS, MLTS and template compression were 10%, 5%, and 5%, respectively (more details about the selected template percentages are shown in Fig. 5.7).

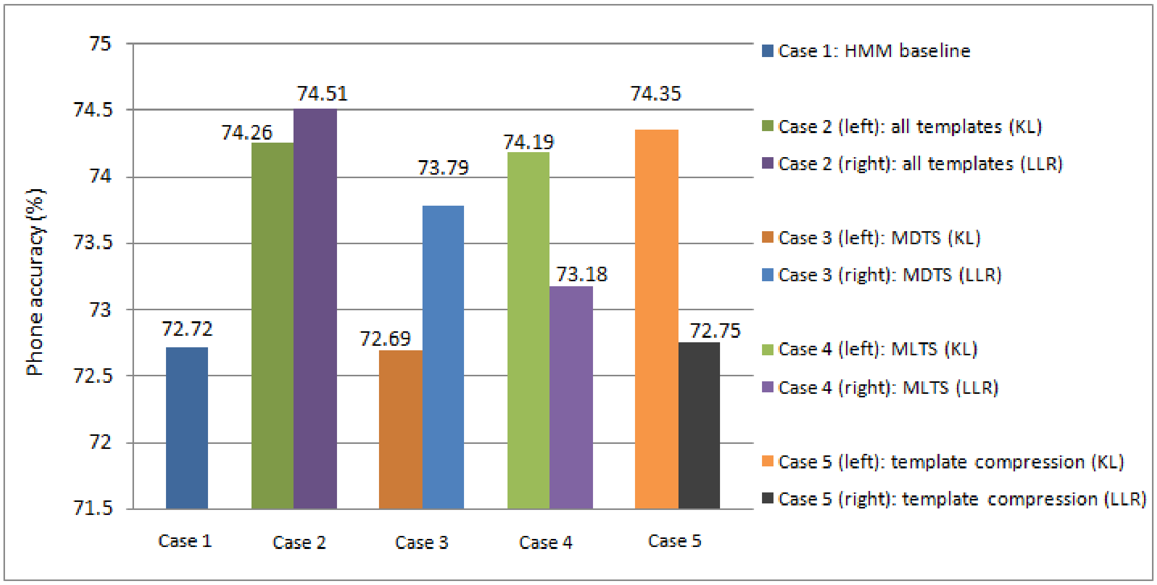

Fig.5.5 Phone accuracies (%) of all template and template representative methods with KL and LLR local distances (mixture size=24)

Several points are worth noting in Fig.5.5. First, when the original templates (without template selection) were used, the LLR distance worked better than the KL distance in DTW. This is due to the fact that the KL divergence measures the distance between GMM distributions and it does not directly measure the distance between frame feature vectors; in contrast, the LLR local distance is computed by plugging the frame feature vectors in the associated GMMs that best fit the vectors, and it therefore more directly measures the distance between two frame vectors.

Second, for the MDTS method, the LLR distance worked better than the KL distance, but for the MLTS method, the KL distance worked better than the LLR distance. In MDTS, the template representatives were selected from the original templates, and therefore LLR distance worked better as discussed above for using all templates. In contrast, in MLTS, each frame vector of a selected template representative was relabeled by GMMs that maximized the likelihood of a cluster of template frame vectors, and therefore the KL distance that measures the distance between GMMs worked better.

Third, relative to the case of using all original templates, MLTS with the KL distance only slightly decreased recognition accuracy, but MDTS with the LLR distance significantly decreased the accuracy. As was discussed above, MDTS simply selects a cluster center as the template representative, but MLTS further refines the GMM indices of template representative frames by maximizing the likelihood of all frame feature vectors in each cluster. Through this procedure, MLTS can generate more informative template representatives than MDTS.

Finally, template compression further improved the performance over MLTS method with the KL distance. This can be attributed to the better representation of the template frame clusters by the new GMMs generated through merging the GMMs of the original frame vectors, which include more information of the frames of the templates in the cluster whereas in MLTS only the labels of GMMs were refined.

In summary, when original templates or MDTS were used, the LLR distance worked well in DTW, and when MLTS or template compression were applied, the KL distance worked well in DTW. Using the respectively compatible local distances, MLTS performed better than MDTS, and template compression further improved MLTS. We also conducted the same significance test on the performance differences between our proposed methods (case 2 to case 5) and the HMM baseline. The proposed methods that improved the recognition accuracy significantly over the baseline at the significance level of $\alpha = 0.05$ are: 1) KL and LLR based all templates in case 2; 2) LLR based MDTS in case 3; 3) KL based MLTS in case 4; and 4) KL based template compression in case 5. For the subsequent experiments, we only used the LLR distance based MDTS method and the KL distance based MLTS and template compression methods.

In Table 5.2, we investigate how the threshold value t of Eq. (4.6) for removing the GMM outliers affected the recognition performance. Since the template compression method is based on the template selection method and only KL divergence local distance based MLTS obtained a significant improvement over the HMM baseline, here we only evaluated the threshold effect suing the KL divergence based MLTS method. When t=2 and 5% template representatives were selected, the KL divergence based template compression method gave the best phone accuracy performance of 74.35%, and when $t = \infty$, all GMMs in a cluster were used to generate compressed templates. For all the subsequent experiments, the threshold value t=2 was used in template compression.

Table 5.2.Phone accuracies (%) from using different threshold values for the compressed template representatives

Threshold tσ	1σ	2σ	3σ	∞
Accuracy (%)	73.99	74.35	73.52	71.02

In Fig. 5.6, we show how the percentages of templates selected from the total templates as the representatives affected recognition accuracies for MLTS with KL distance, and we also show the effect on the template representatives from using different numbers of GMMs in labeling the frames of the original templates. The percentages varied from 100% down to 1% and the number of GMMs used to label each frame vector was 1, 3, and 5. For the 1 GMM case, the number of GMM clusters l in MLTS was set to 5 with the intention of selecting 5 GMMs to label each frame vector for the template representatives. However, when the percentage of templates selected as representatives was high, an aligned frame vector set may have less than 5 GMMs, and in such a case all GMMs in the set were used to label the corresponding frame vector of the template representative. It is observed from Fig. 5.6 that when each frame vector was labeled by 1 GMM, using 100% templates produced a phone accuracy lower than using 80% templates since the latter used MLTS to label each frame vector by more than 1 GMMs (1.3 on average); when the percentage was reduced from 80% to 60%, phone accuracy decreased as there were less template representatives to be used in rescoring, and each frame vector of the template representatives was still labeled by less than 2 GMMs (1.7 on average); when the percentage further decreased from 60% to 5%, phone accuracy increased steadily since on average each aligned frame set had more GMM candidates for selection, and the chance of finding good and sufficient GMMs to label the frame vectors increased (for the cases of 40%, 20% 10% and 5% template representatives, the average number of GMMs used to label each frame of template representatives were 2.6, 5, 5, and 5, respectively) even though the number of template representatives decreased. The best accuracy performance was obtained when only 5% template representatives were selected. When the percentage reduced to 1%, the template representatives became insufficient and the accuracy performance dropped. When 3 GMMs were used to label each frame vector, the trend of the phone accuracy curve is similar to the 1 GMM case except that the peak of the former occurred at 20%. When 5 GMMs were used to label each frame vector, using more template representatives produced higher recognition accuracy. Compared with the 1 GMM case, when 3 GMMs or 5 GMMs were used to label each frame vector, there were more GMM candidates to be selected to form l (l=5) clusters, but the confusion among the selected templates also increased since there were larger overlaps of GMMs in the original frame vector labels.

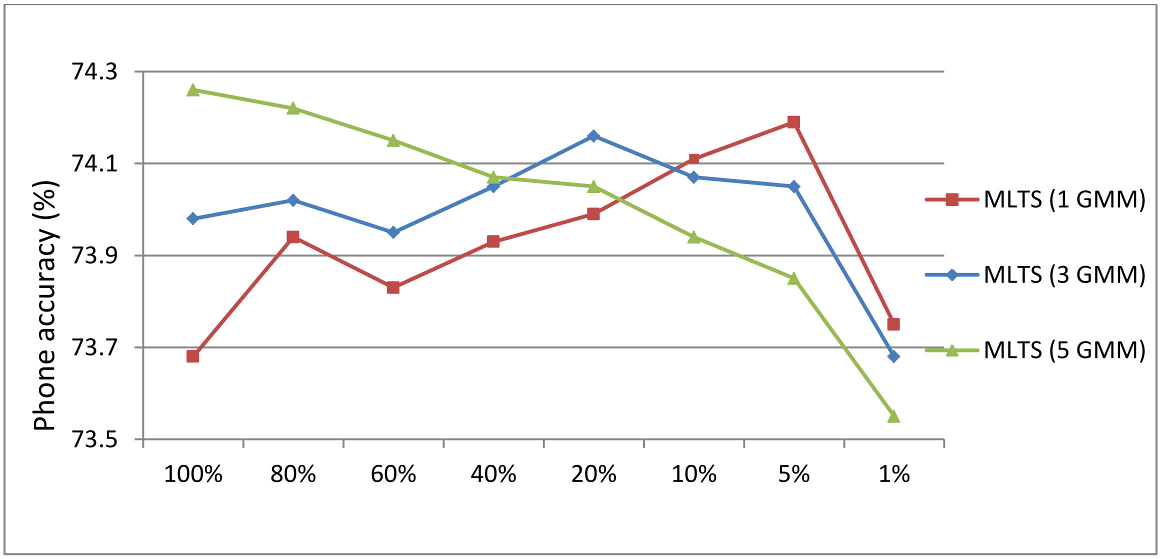

Fig.5.6: Phone accuracies (%) for MLTS versus percentages of template representatives with each frame vector labeled by 1, 3, and 5 GMMs, respectively.

It is also worth noting that for the 1 GMM case in Fig. 5.6, the accuracy performances for the high percentage of template representatives like 80% and 60% were inferior to those in the 3 GMM and 5 GMM cases due to fewer choices of GMM candidates for each aligned frame vector set. However, since the purpose of the proposed MLTS is to reduce computation and storage costs, the performance of the small percentages of template representatives as shown in the 1 GMM case is more relevant. Therefore, in Fig. 5.7, we only show the performance of template compression for the 1 GMM case with KL distance in comparison with MLTS and MDTS. It is observed that in the low percentage cases, template compression improved accuracy performance over MLTS and the best result was obtained when only 5% template representatives were used. For MDTS, the number of GMMs used to label each frame vector was 5, where using more template representatives produced higher recognition accuracy. We also did experiments using 1 and 3 GMMs to label each frame vector for MDTS, where the recognition accuracy had the similar trend with the 5 GMM case.

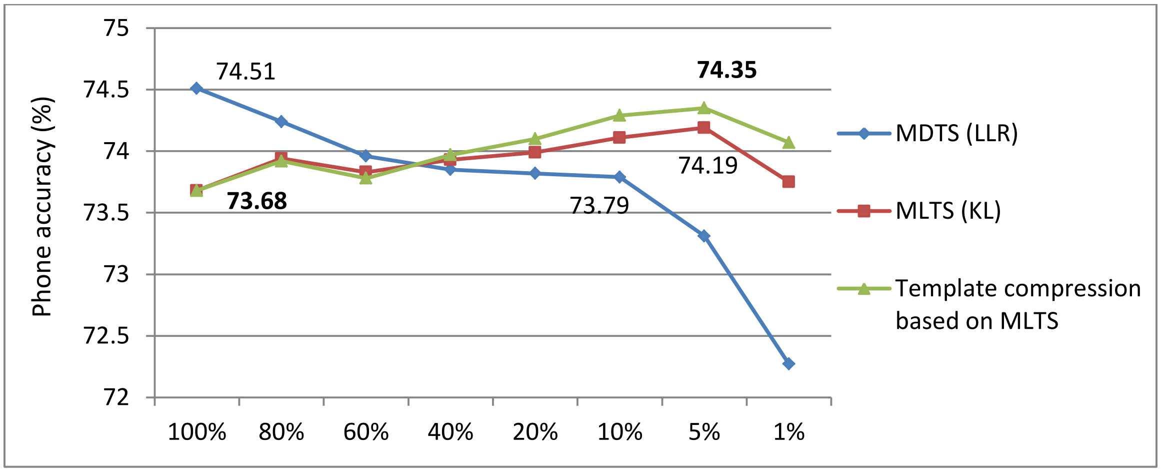

Fig.5.7: Phone accuracies (%) versus percentages of template representatives for MLTS, template compression, and MDTS

5.3 Large Vocabulary Speech Recognition

We evaluated the word accuracy performance for the telehealth task using the methods of all templates, template selection, and template compression based on the KL and the LLR local distances. The HMM baseline was trained using crossword triphone models (The phone “sp” was inserted between two words and the phone “sil” was added at the beginning and end of utterances). The word lattices were generated with token size *n*=4. In Table 5.3, we compare the recognition accuracies for the 5 doctors over six cases: 1) HMM baseline, 2) all templates with KL local distance in DTW, 3) all templates with LLR local distance in DTW, 4) MDTS with LLR distance in DTW, 5) MLTS with KL distance in DTW, and 6) template compression with KL distance in DTW. In template selection and compression, 10% templates were selected from the total templates as the representatives since at that point, the best recognition accuracy were obtained. When all templates were used with the LLR distance in DTW, the average word accuracy performance for the 5 doctors was 80.91%, which was a gain of 1.59% absolute over the baseline and was 0.33% higher than using the KL local distance in DTW. As in the case of TIMIT phone recognition, when the original templates were used, the LLR local distance worked better than the KL local distance did. By using MLTS with KL distance in DTW, the average word accuracy for the 5 doctors were 80.36% which was a gain of 1.04% absolute over the baseline and was 0.2% higher than using MDTS with the LLR distance in DTW. Using template compression with the KL distance in DTW further improved the word accuracy to 80.59% which was a gain of 1.27% absolute over the baseline and was 0.23% higher than using MLTS with the KL distance in DTW. In template compression, the number of Gaussian components included in each GMM of the condensed template was 16 which was the same as in the GMMs of the baseline HMMs, and the average number of GMMs generated for the condensed template representatives were 1048 for each of the 5 doctors. Again, we conducted a student t significance test on the average performance of the 5 doctors in the five cases from 2) to 6) against the baseline. All the five cases described above improved the recognition accuracy significantly over the HMM baselines at the significant level of $\alpha = 0.05$.

Table 5.3. Word accuracies (%) for HMM baselines, all templates, template selection, and template compression

Speakers (# of word)	Dr.1 (6421)	Dr.2 (3988)	Dr.3 (5085)	Dr.4 (3248)	Dr.5 (2759)	Weighted Average
Baselines	79.32	84.00	82.50	72.14	74.20	79.32
All templates (KL)	80.35	85.38	83.79	73.20	75.26	80.58
All templates (LLR)	80.67	85.98	84.22	73.53	75.74	80.91
MDTS (LLR)	79.97	85.03	83.55	72.90	74.94	80.16
MLTS (KL)	80.27	85.13	83.60	73.15	75.24	80.36
Template compression (KL)	80.42	85.49	83.84	73.42	75.42	80.59

For the 5 doctors, we also investigated how the average percentage of templates selected as template representatives affected recognition accuracies for MDTS, MLTS, and template compression. The three cases had similar trends as in the phone recognition task shown in Fig. 5.7. When 10% templates were selected as the template representatives, MLTS and template compression obtained the peak performance reported in Table 5.3, while the costs of computational time and memory storage were reduced largely.

5.4 Computation and Space Overheads

5.4.1 TIMIT Phone Recognition

We first compare the storage space costs of the conventional and the proposed template representation methods. In conventional template methods, a speech frame vector is represented by a 39-dimensional vector (float) which is now labeled by n GMM indices (integer) and the associated n weights (float) in the proposed method. With *n*=5 in our experiments, the proposed frame labeling method saved 5.2 times of memory space over the conventional frame vector based templates.

We next make a comparison on the computation and storage space overhead between the methods of using all templates with LLR local distance and using MLTS with KL local distance in DTW. For the method of using all templates, we need to store all 152715 GMM indexed training templates. There were around a total of 400 PDT tied triphone clusters for each fixed state of all phone HMMs. Accordingly, in lattice rescoring, each test segment needs to make on average of 3*(152715/400)≈1145 comparisons with the triphone templates, where the factor 3 comes from the three different PDT tyings for the three HMM states. In the MLTS method, with the number of template representatives being 5% of total templates, 95% computer memory space and computational time were saved from using all templates, while the phone recognition accuracy only decreased from 74.51% to 74.19%. For the TIMIT dataset, the average length of a phone template was 8 frames with the frame shift of 10ms. By using 5 GMM indices to label each frame vector, the storage space was around 32M for all templates and 1.6M for template representatives (about 7636), respectively. For the baseline HMM, the storage space for the acoustic model was about 18MB with the double data type (HTK format) for the whole set of model parameters when 1189 GMMs with the mixture size 24 were used. The total memory space overhead over the baseline was 177% and 8.9%, respectively, for the all template and template representative methods. A rough comparison on the runtime computation costs relative to the baseline is as follows (although the current codes have not been optimized for speed). Relative to the time of the baseline that generates the 1-best phone string hypotheses, the overhead of lattice generation was 20% and the overhead for calculating the likelihood scores to label the test data frames with the GMMs was 9%. In addition, the overhead of lattice rescoring was 68% by using all templates and only around 3% by using the template representatives (the average 1-best path search time was included). Therefore the total computation time overhead over the baseline was 97% and 32%, respectively, for the all template and the template representative methods. By using template representatives, the costs in computation time and storage space were greatly reduced while the recognition gain over the baseline only slightly decreased relative to using all templates. In Table 5.4, we summarize the memory space and computational time overhead of using all templates and MLTS for the phone recognition task.

Table 5.4 Memory and computation overheads of using all templates and template selection for the TIMIT phone recognition task

	All templates (LLR)	MLTS (KL)
Recognition accuracy gain (absolute)	1.79%	1.47%
Number of templates used	152,715	7636
Memory space used	32MB	1.6MB
Overhead time of lattice generation	20%	20%
Overhead time to label test data	9%	9%
Overhead time of lattice rescoring	68%	3%
Total memory space overhead	177%	8.9%
Total computation overhead	97%	32%

5.4.2 Telehealth Speech Recognition

In Table 5.5, we summarize the average memory space and computational time overheads of the telehealth captioning task for the cases of using all templates with LLR local distance and using MLTS with KL distance. When all templates were used, we stored 181601 GMM indexed training templates and the memory space overhead was 130%. In the MLTS method, only 10% template representatives were used and the memory space overhead was only 12.7%. The computation overhead for all templates was 335%, and it was reduced to 64% when 10% template representatives were used. The overhead time of lattice rescoring for LVCSR was much higher than that of the phone recognition task of TIMIT, since in LVCSR, a word lattice included both word and phone boundaries, template matching was performed on each triphone, and the score of a word was calculated by adding the template matching scores of all triphones in the word.

Table 5.5 Average memory and computation overheads of using all templates and template selection for 5 doctors in the teleheath task

	All templates (LLR)	MLTS (KL)
Recognition accuracy gain (absolute)	1.59%	1.04%
Number of templates used	181,601	18,160
Memory space used	39MB	3.8MB
Overhead time of lattice generation	28%	28%
Overhead time to label test data	9%	9%
Overhead time of lattice rescoring	298%	27%
Total memory space overhead	130%	12.7%
Total computation overhead	335%	64%

Chapter 6

Effectiveness of Statistical Modeling based Template Matching

Since the proposed template based methods were implemented using GMM indices extracted from a HMM baselines, we also want to know if the template methods can make consistent improvement when better HMM baselines are obtained. Since the LLR based all template method made the most improvement over the HMM baseline, we only evaluate this method for the TIMIT phone recognition task by using four better HMM baselines that were generated by Discriminative Training (DT) of Minimum Phone Error (MPE), MFCC concatenated with ensemble Multiple Layer Perceptron (MFCC+EMLP) features, DT combined with the MFCC+EMLP features, and data sampling based ensemble acoustic models integrated with DT and MFCC+EMLP features. These four HMM baselines were previously investigated for the TIMIT task in [17]. In the work here, we trained these baseline models by using HTK [56] and used them in template representations, matching, and lattice generation.

6.1 Template Matching based on the DT Baseline Model

Discriminatively trained models have been shown to significantly improve error rates, as they have more power to better differentiate between confusable sounds. Minimum Phone Error (MPE) based DT has been shown to improve HMM baseline for TIMIT in [17]. In our current work, MPE was used to train a HMM baseline, where 39-dimensional features defined by 13 MFCCs and their first and second time derivatives were used. We first used the maximum likelihood criteria to train a basic HMM baseline which had the recognition accuracy 71.86%. We then used the MPE criterion to train the discriminative models on top of the basic baseline HMMs. The discriminative models were obtained with 4 iterations of DT. The DT trained HMM baseline had the recognition accuracy of 73.25% and the accuracy gained from the MPE training over the basic HMM baseline was 1.39% absolute. We used the discriminative HMMs as a new baseline to generate the phone lattices for rescoring, where three different size lattices were generated with the token sizes of n=2, 3, and 4, with the average numbers of nodes per lattice in the order of 250, 850 and 1800, and the average numbers of arcs in the order of 450, 2350 and 6250, respectively, representing small, medium, and large lattices for the current TIMIT task. We extracted 1189 GMMs from the DT baseline model to label the frame vectors of templates.

In Fig. 6.1, we compare phone recognition performances by using the MPE based HMM baseline and template matching rescored results on three lattice sets which were generated with tokens *n*=2, 3, and 4. Template matching made the largest improvement of 1.49% absolute over the MPE baseline on the smallest lattice size (token *n*=2). When the lattice size increased to *n*=4, even though template matching still made a 1.02% improvement over the baseline, compared with the case *n*=2, the gain decreased.

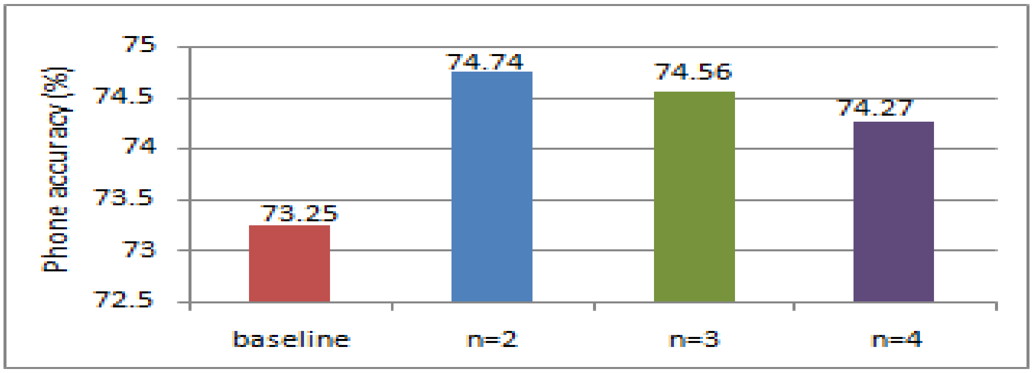

Fig. 6.1: A comparison of phone accuracy (%) for MPE trained HMM baseline and template matching based lattice rescoring on three lattice sets with tokens *n*=2, 3, and 4

6.2 Template Matching based on the Baseline Model Generated by MFCC+ EMLP Features

The concatenation of MLP features with traditional MFCC features has been proven effective in different tasks [62]. In [17], MFCC plus ensemble MLP (EMLP) features were used on the TIMIT task and a significant improvement was obtained over a HMM baseline. In the current work, the MFCC+EMLP features that were available in the spoken language and information processing LAB were also used to generate HMMs as our baseline, where 10-fold cross validation (CV) data sampling was used to build the MLP ensemble to generate the EMLP features., and for each speech frame, PCA was used to reduce the EMLP feature dimension from 39 to 15, and the reduced EMLP features were then concatenated with the original 39 MFCC-based features to form a 54-dimensional feature vector. For more details about the generation of the EMLP features, please refer to [17]. The MFCC+EMLP feature based HMM baseline had the phone recognition accuracy of 75.66%, from which 1678 GMMs were extracted to generate templates. The MFCC+EMLP baseline models were used to generate the phone lattices for rescoring. Lattices with the three token sizes as we described above were generated and used again.

In Fig. 6.2, we compare phone recognition performances by using the MFCC+EMLP feature based HMM baseline and template matching based lattice rescoring on the three lattice sets. Template matching made the improvements of 1.37%, 1.61%, and 1.49% absolute over the MFCC+EMLP feature baseline for the three lattice sets with tokens *n*=2, 3, and 4, respectively. It’s worth noticing that this time template matching made the most improvement over the HMM baseline when the lattice token *n*=3.

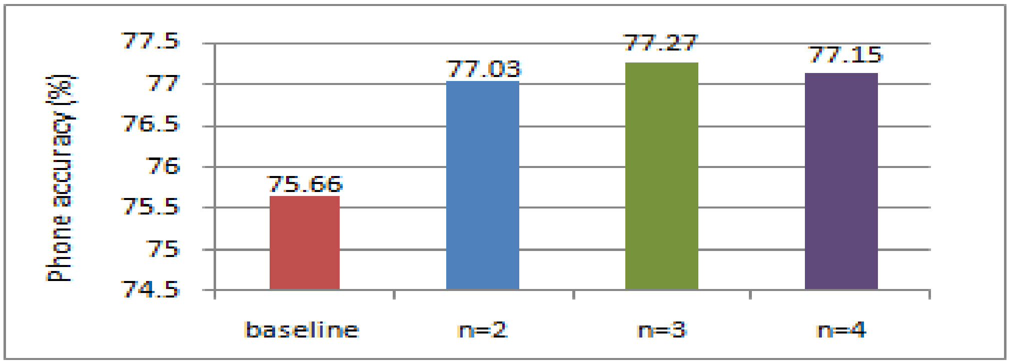

Fig. 6.2: A comparison of phone accuracy (%) for the MFCC+EMLP feature based HMM baseline and template matching based lattice rescoring on three lattice sets with tokens *n*=2, 3, and 4

6.3 Template Matching based on the Baseline Model Generated by DT Integrated with MFCC+EMLP Features

We further combined the MPE based discriminative training with the MFCC+EMLP feature based HMMs. Using the MFCC+EMLP based HMMs as the initial model, MPE based discriminative training was then performed with 4 iterations to further optimize the model parameters. The newly trained MPE+MFCC+EMLP based models had the same number of HMM parameters as the MFCC+EMLP baseline HMMs, and the phone recognition accuracy of the new baseline was 76.51%. From the newly trained baseline HMMs, 1678 GMMs were extracted to index the template frames, and three sets of phone lattices were again generated.

In Fig. 6.3, we compare phone recognition performances by using MPE+MFCC+EMLP based HMM baseline and the template matching based lattice rescoring on the three lattice sets. Template matching made the phone accuracy improvements of 1.07%, 1.25%, 1.47% absolute over the HMM baseline on the lattice sets with tokens *n*=2, 3, and 4, respectively. It is worth noting that this time template matching continued improving the baseline until the lattice token *n*=4 (We also generated a lattice set with token n=5 to evaluate template matching, but the performance gain became 1.31% which was smaller than the case of *n*=4).

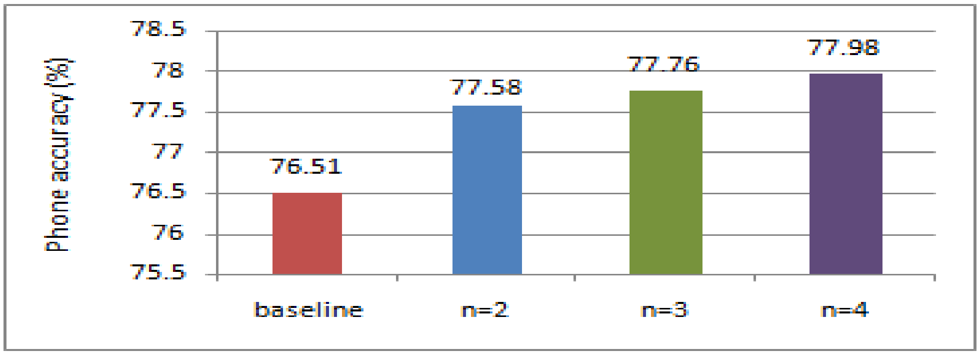

Fig. 6.3: A comparison of phone accuracy (%) for MPE+MFCC+EMLP based HMM baseline and template matching based lattice rescoring on three lattice sets with tokens *n*=2, 3, and 4

6.4 Template Matching based on the Baseline Model Generated by Ensemble Models Integrated with DT and MFCC+EMLP Features

Using data sampling approach to generate ensemble acoustic models was discussed in [17], where through data sampling, multiple training data sets were produced, and each sampled training data set was used to train one set of acoustic models which is also called a base model. For an *N*-fold cross validation (CV) data sampling, a (*N*-1)/*N* fraction of training data is included in each sampled training set. An ensemble model generated by data sampling usually makes significant performance improvement over the single model trained from the full training set, even though each individual base model in the ensemble model has lower performances [17]. In Section 6.3, we obtained the phone recognition accuracy of 76.51% for MPE+MFCC+EMLP based HMM baseline that was trained using the full training set. Here we applied a 10-fold CV data sampling to produce an ensemble of 10 base models. Each base model was trained in the mode of MPE+MFCC+EMLP. The phone recognition accuracy of the ensemble acoustic model was 77.97%, which made an improvement of 1.46% absolute over the baseline model in Section 6.3. We used the base models in the ensemble acoustic model to generate 10 lattice sets for each fixed token size *n*, and template matching was performed individually on the lattices generated by the corresponding model in the 10 base models for rescoring. The average number of GMMs extracted from each one of the 10 individual model sets to be used in template construction was 1558. In order to combine the 10 different rescored lattices from these 10 individual base models, we first used HTK to convert each lattice in the 10 lattice sets to the corresponding confusion network (CN) and then combined the 10 CNs to produce the final rescored recognition result [63].

In Fig. 6.4, we plot the recognition accuracies for the individual base models and the template matching rescored accuracy with the three lattice sets generated by the corresponding base models. We can see that template matching improved the accuracy of the individual models, and on the lattice set with token *n*=4, template matching made the largest improvement. In Table 6.1, the phone recognition accuracies were averaged for the 10 base models and for the lattice rescoring results based on the 10 base models with the token size *n*=2, 3, and 4, respectively. In Table 6.2, we show the phone recognition accuracy results for the baseline models of MPE+MFCC+EMLP that was trained by using the full training set, the ensemble acoustic models, and the CN integration of lattice rescoring results on each of the three lattice sets. When token *n*=4, the lattice rescoring results based on the CN integration produced the best recognition accuracy of 79.55% which was a 1.58% absolute improvement over the ensemble acoustic model baseline. In addition, we also implemented the lattice rescoring based on lattices generated by the ensemble model (more details about computing the acoustic scores by combining the 10 base models can be found in [54]). We did lattice rescoring on the same lattices using templates extracted and constructed from the 10 based models. Since the lattice rescoring by the 10 base models was implemented on the same lattices, for an arc in a lattice, there were 10 different scores generated and the average scores were taken as the final scores for the arcs in lattices. The best phone recognition accuracy generated by this lattice rescoring method was 78.96% with the token size *n =4*.

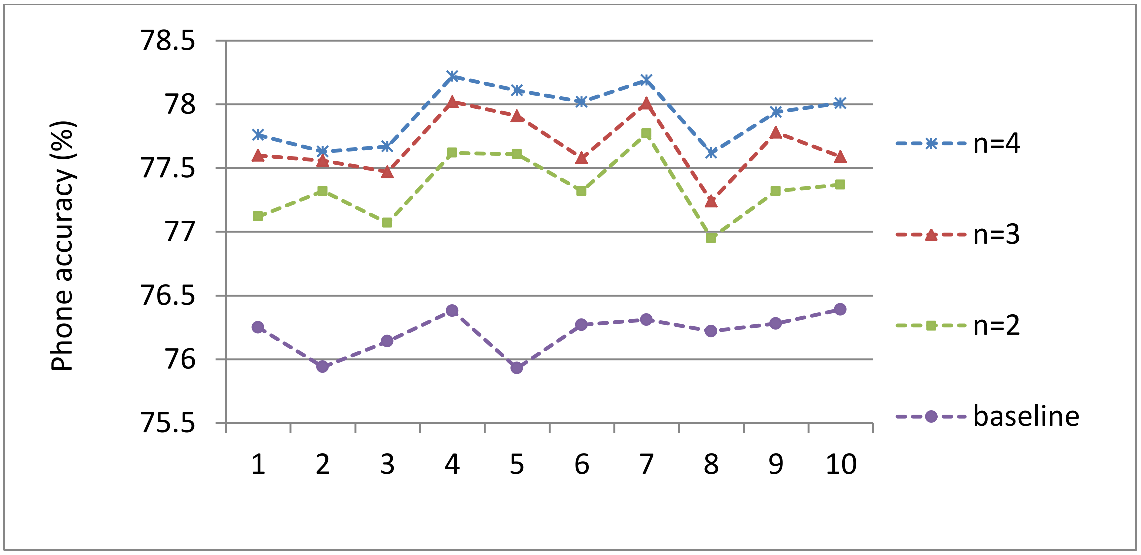

Fig. 6.4: Recognition accuracies (%) of individual base models and template matching based rescored results of each base model with three lattice sets (token *n*=2, 3, and 4).

Table 6.1. Average recognition accuracies (%) for base models and lattice rescoring results on three lattice sets with tokens *n*=2, 3 and 4.

Method	Base models	*n*=2 (rescoring)	*n*=3 (rescoring)	*n*=4 (rescoring)
Averaged accuracy	76.21	77.35	77.68	77.92

Table 6.2. Recognition accuracies (%) for baseline single model trained using the full training set, the ensemble acoustic models, and the CN integration of the lattice rescoring results on three lattice sets

Method	Phone accuracy	
MPE+ MFCC+EMLP based single model	76.51	
MPE+MFCC+EMLP based ensemble models	77.97	
CN integration of template matching based lattice rescoring	*n* =2	78.89
	n=3	79.28
	n=4	79.55

6.5 Experimental Result Analysis

We summarize the accuracy performances of the different baselines and their best lattice rescoring results with the corresponding lattice sizes in Table 6.3. The template matching approach has made consistent and significant improvements over the HMM baselines with increasing recognition accuracies. In addition, the best lattice size determined by the number of tokens n for template matching increased when the quality of the baseline models improved. In Fig.6.5, we show the average rank of correct phone string references when they were inserted into the different lattice sets for the three methods of baseline generation. The MPE had the lowest phone baseline accuracy among these three methods, and in this case template matching produced the best performance for the lattice set with token *n*=2. We also notice that when the token size for the lattices generated by MPE models increased from 2 to 4, the average rank of correct references significantly "decreased". For the method of MFCC+EMLP features, its baseline accuracy was better than MPE. When the lattice token increased from 2 to 3, the average rank of the correct references was stable. However, when the lattice token further increased from 3 to 4, the rank largely "decreased". In this case, template matching obtained the best recognition accuracy for the lattice size generated by token *n*=3. The method of MPE+MFCC+EMLP was the best baseline among these three methods, and template matching achieved the best recognition accuracy for the lattice size with token *n*=4. In this case, the average rank of the reference phone strings was almost unchanged when the lattice token increased from 2 to 4.

Table 6.3. Summary of recognition accuracies (%) of the four baselines and the best lattice rescoring results with the corresponding lattice size

Method	Baseline	Template Matching	# of tokens
MPE	73.25	74.74	2
MFCC+EMLP	75.66	77.27	3
MPE+MFCC+EMLP	76.51	77.96	4
MPE+MFCC+EMLP+ Ensemble models	77.97	79.55	4

In general, the phone recognition results of higher recognition accuracy should be closer to the correct references. When the lattice size increases, if the average rank of the correct references "decreased" largely, then the possibility of picking up the correct hypotheses by rescoring becomes smaller, whereas if the rank of the correct references stays stable, then the additional correct hypotheses provided by large lattices are more likely to be picked up by rescoring and better recognition accuracy can be achieved. Based on these observations and reasoning, we conclude that the template matching approach has a better capacity to pick up correct hypotheses from the large lattices when better baseline models are used. To support the analysis, in Table 6.4, we also provide the percentages of the inserted correct references being ranked the first in the TIMIT test data with the different lattice sets produced by the three token sizes.

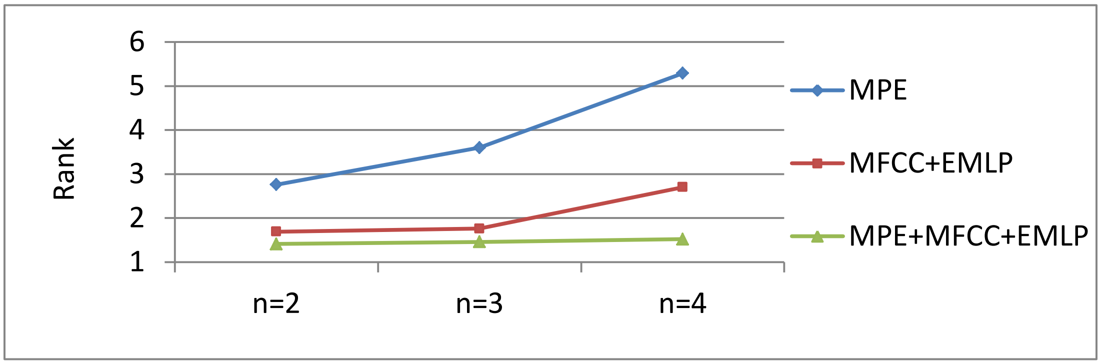

Fig. 6.5: Average ranks of correct references in three lattice sets for the three methods of baseline generation

Table 6.4. Percentages of the correct references ranking the first in the TIMIT test data with three different lattice sizes

Method	n=2	n=3	n=4
MPE	84.2%	82.7%	78.6%
MFCC+EMLP	88.6%	88.2%	85.9%
MPE+MFCC+EMLP	89.8%	89.8%	89.5%

Chapter 7

Integrate Template Matching with Prosodic Information

In this chapter, we investigate new methods to integrate prosodic information of phone duration, phone pitch and phone energy into our current template matching framework in order to further improve speech recognition accuracy.

7.1 Method of Prosodic Information Extraction

As we discussed above, phone duration, energy, and pitch are prosodic features that may help improve speech recognition accuracy. In this section, we discuss how to extract these three types of prosodic information.

7.1.1 Phone Duration and Energy Extraction

The key issue in extracting phone duration and energy prosodic information is to get phone boundaries, which can be obtained by using the aligned transcriptions of training data and the lattices generated for test data. After the boundary information is obtained, the extraction of duration is straightforward, and the energy of a phone segment is calculated by using the following formula:

$$E = log \sum_{n=1}^{N} s_n^2 \qquad (7.1)$$

where s_n , $n = 1 \ldots N$ are speech samples and N is the number of samples in the phone segment.

7.1.2 Phone Pitch Extraction

Phone pitch is more difficult to extract compared with phone duration and energy and also it is easier to make mistakes. We extract the phone pitch information from speech data by using CMU Yin algorithm [64] which is an algorithm for estimating the fundamental frequency (*F0*) of speech and is based on the well-known autocorrelation method with a number of modifications that are combined to prevent errors. The algorithm has several desirable features such as low error rates, no upper limit on the frequency search range, efficiency, low latency, and few parameters. For more detail about the algorithm, please refer to [64]. Yin algorithm extracts the pitch information for each speech frame. The pitch value *P* of a phone was calculated from the pitch values of the speech frames P_n within a phone segment as:

$$P = \frac{1}{N}\sum_{n=1}^{N} P_n \tag{7.2}$$

where *N* is the number of frames for a phone.

7.2 Integrate Prosodic Information with Template Matching

In this section, we investigate an integration of the prosody information with the template matching methods discussed in the earlier chapters. Two methods of calculating the prosodic information scores were evaluated. One was a parametric method based on GMM and the other was a non-parametric method. The prosodic information scores were used to integrate with three template based methods as we discussed in the previous chapters, including the LLR local distance based all templates method, the KL local distance based all template method, and the LLR local distance based MDTS method.

We investigate how different prosodic information of duration, pitch and energy affect speech recognition accuracy separately in order to know which prosodic information has more influential effect on speech recognition accuracy. We also explore first combining duration, energy, and pitch together and then integrating with template based methods to see whether a better speech recognition performance can be obtained.

7.2.1 Integration of GMM based Prosodic Scores with Template based Methods

The basic idea of the GMM based method is to model the prosody information duration, energy, and pitch by using the Gaussian Mixture Models (GMMs). Since speech is affected by many factors as we introduced in Chapter 1, a phone pronunciation from a speaker can be different under different situations and different speakers may have different pronunciations for the same phone. The exact distributions of the prosodic features are unknown. However, GMMs are able to approximate any distribution, which makes it a good choice for modeling prosodic feature distribution. Phone duration, energy, and pitch extracted as discussed in Section 7.1 can be used to train GMMs and the number of GMM components can be tuned using a development set which will be discussed in Section 7.3. The prosody likelihood score $S_{prosodic}$ can be calculated by plugging the prosodic feature value into GMMs. We combine the score of the previous LLR or KL based all template methods and the newly calculated GMM based prosodic score by using a weighted average:

$$S = w_1 * S_{TM} + w_2 * \log(S_{prosodic}) \tag{7.3}$$
$$w_1 + w_2 = 1\,, w_1 \geq 0, w_2 \geq 0$$

where S_{TM} represents a negated distance score obtained from the previous LLR or KL based all template method for measuring similarity here. The weights w_1 and w_2 can be tuned by using a development set.

For the LLR based MDTS method, we also use prosodic information to help template selection. The log prosodic scores were first negated to become a dissimilarity score and then added to the distance scores. Given a cluster C_i, the template-to-cluster distance is defined as:

$$D(s_x, C_i) = w_3 * \frac{1}{|C_i|} \sum_{\substack{s_x, s_{x'} \epsilon C_i \\ s_x \neq s_{x'}}} D(s_x, s_{x'}) + w_4 * (-1 * \log(S_{prosodic})) \qquad (7.4)$$

where w_3 and w_4 are the combination weights with $w_3 \geq 0$, $w_4 \geq 0$, and w_3+$w_4 = 1$. Again, the weights can be tuned on a development set. A template s^* is selected as the cluster template representative when it satisfies the following condition:

$$s^* = \underset{s_x \epsilon C_i}{\operatorname{argmin}} D(s_x, C_i)$$

For both prosodic scores in Eq. (7.3) and (7.4), $S_{prosodic}$ can be a single score from phone duration, energy, or pitch, and it can also be a combined score from these three different types of prosodic information:

$$S_{prosodic} = w_{dur} * S_{Prosodic}^{dur} + w_{energy} * S_{Prosodic}^{energy} + w_{pitch} * S_{Prosodic}^{pitch} \qquad (7.5)$$

where the weights are positive and sum to 1: w_{dur}+ w_{energy}+ w_{pitch} =1. Again these three weights can be tuned on a development set.

7.2.2 Integration of Non-Parametric Prosodic Scores with Template based Methods

Prosodic score can also be derived directly from distance calculation and we call the method non-parametric since no model is used in calculating the prosodic scores. For the LLR or KL based all template method, a distance between a template *N* and a test segment *T* is calculated by:

$$S = w_1 * S_{TM} + w_2 * S_{prosodic} \qquad (7.6)$$

where S_{TM} has the same definition as above, w_1 and w_2 are combination weights, and $S_{prosodic}$ is defined by :

$$S_{prosodic} = -1 * \left|\log(T_{prosodic}) - \log(N_{prosodic})\right| \qquad (7.7)$$

where $T_{prosodic}$ and $N_{prosodic}$ are the prosodic values of the template T and a test segment *N*. $S_{prosodic}$ is negated to measure the similarity between T and N , to be consistent with S_{TM}.

The rationale behind the score $S_{prosodic}$ is that it is close to zero if a phone hypothesis is supported by a set of templates with similar prosodic feature values; otherwise, the scores will decrease and therefore penalize the hypothesized phones with aberrant prosodic features. There are two reasons for using the absolute difference of the log-transformed features: first, it is more consistent with the template matching DTW scores and LM log probabilities, and second, taking absolute value focuses on the measurement of relative difference between *T* and *N*.

For the method MDTS (LLR), prosodic information can be used to help select template representatives as:

$$D(s_x, C_i) = \sum_{\substack{s_{x'} \epsilon C_i \\ s_x \neq s_{x'}}} \left\{ w_3 * D(s_x, s_{x'}) + \; w_4 * \left| T_{prosodic}^{s_x} - T_{prosodic}^{s_{x'}} \right| \right\} \qquad (7.8)$$

where w_3 and w_4 are the combination weights and $T_{prosodic}^{s_x}$ and $T_{prosodic}^{s_{x'}}$ are the prosodic values of s_x and $s_{x'}$, respectively. The template s^* is selected as the cluster template representative when it satisfies the following condition:

$$s^* = \arg min_{s_x \epsilon C_i} D(s_x, C_i) \qquad (7.9)$$

In addition, the combination of the three types of prosodic information is defined in the same way as in Eq. (7.5).

7.3 Experimental Results

The experimental dataset is the telehealth corpus which was described in Section 5.1.2. The prosodic information was integrated with the LLR based all template method, the KL based all template method, and the LLR based MDTS method, where both the GMM based method and the non-parametric method were used to compute the prosodic scores.

7.3.1 Experimental Results of Integrating GMM based Prosodic Scores with Template based Methods

We used the prosodic information extracted from the aligned transcriptions of training data to train GMM models with the different component sizes. One doctor's data set was used as the development set, and for the GMM based prosodic scores, several different sizes (*n*=1, 2, 4, 6, and 8) of GMM components were tried. In Appendix, we show that, in general, when *n*=4, the best recognition performance was obtained for combining the prosodic scores of duration, energy and pitch with the template based methods. Therefore, the number of GMM components was set to 4 for the other four doctors. In addition, the combination weights of Eq. (7.3), Eq.(7.4) and Eq.(7.5) were not tuned in Table I to Table IV in Appendix . However, in Table 7.1, experimental results were presented for the same doctor (Dr.1) with tuned weights. For other four doctors, the same weights were taken when combining the prosodic information scores with the template method scores. In Table V to Table VIII in Appendix, the experimental results for the four other doctors are also presented. From these tables, we notice that, in general, duration combined with the template based methods gave the best recognition performance. Energy held the second place, and pitch was the last. In addition, when duration, energy, and pitch were first combined by using different weights and then integrated with the template method based scores, improved performance was obtained. In Table 7.2, the average scores were listed based on the other four doctor's scores from Table V to Table VIII in Appendix. All templates with LLR local distance obtained 0.38% absolute accuracy improvement and all templates with KL local distance got 0.3% absolute accuracy improvement when combined with the three prosodic information scores. Besides, MDTS (LLR) obtained 0.15% improvement when the three prosodic scores were combined to help select 10% template representatives. MDTS used the same weights (w_1) as the LLR based all templates when combining with the prosodic scores.

Table 7.1 Word accuracies (%) of integrating template based methods and GMM based prosodic scores for Dr.1

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	80.67	80.74 (w_1=0.74)	80.76 (w_1=0.82)	80.40 (w_1=0.89)	80.81 (w_{dur}=0.48, w_{Energy}=0.41, w_1 =0.82)
All templates (KL)	80.35	80.29 (w_1=0.78)	80.25 (w_1=0.88)	79.91 (w_1=0.92)	80.51 (w_{dur}=0.47, w_{Energy}=0.42, w_1 =0.87)
MDTS (LLR)	79.97	79.91 (w_3=0.75)	79.87 (w_3=0.86)	79.54 (w_3=0.9)	79.99 (w_{dur}=0.45, w_{Energy}=0.40, w_3 =0.83)

Table 7.2 Average word accuracies (%) of integrating template based methods and GMM based prosodic scores for the other four doctors

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	80.83	81.1	80.96	80.78	81.21
All templates (KL)	80.37	80.53	80.43	80.16	80.67
MDTS (LLR)	80.07	80.08	79.89	79.49	80.22

7.3.2 Experimental Results of Integrating Non-parametric Prosodic Scores with Template based Methods

Prosodic information scores of duration, energy, and pitch based on the non-parametric method were also used to combine with the template methods. In Table 7.3, experimental results are presented for Dr.1 whose data was used to tune the combination weights as we did in Section 7.3.1, and the tuned weights in Eq. (7.5), Eq.(7.6) and Eq.(7.8) are listed in Table 7.3. For the other four doctors, the same weights were taken when combining prosodic information scores with template method scores. In Table IX to Table XII in Appendix, the experimental results for the other four doctors are also presented. From these tables, the same trend can be noticed that duration obtained the best recognition performance when combined with template based methods, energy was the second best, and pitch was the last. In addition, when three different types of prosodic information of duration, energy, and pitch were first combined and then integrated with the scores of template method, improved performance was obtained as in the GMM based method. In Table 7.4, the average scores were listed based on the other four doctor's scores from Table IX to Table XII. All templates with LLR local distance got 0.39% absolute accuracy improvement when combined with the three prosodic information scores together and all templates with KL local distance obtained 0.38% absolute accuracy improvement. In addition, MDTS (LLR) obtained 0.13% improvement when combined with the three prosodic scores to select 10% template representatives. MDTS took the same weights (w_1) as the LLR based all template method when combining with prosodic scores. From the averaged experimental results for the other four doctors in Table 7.2 and Table 7.4, we notice that for the LLR based all template and MDTS methods, when integrating with the prosodic scores computed by the non-parametric method, improvements similar with the GMM based method were obtained. However, for the KL based all template method, using non-parametric method to calculate prosodic scores gave slightly better performance than GMM based method did.

Table 7.3 Word accuracies (%) of integrating template based methods and non-parametric prosodic scores for Dr.1

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	80.67	80.83 (w_1=0.62)	80.81 (w_1=0.66)	80.52 (w_1=0.77)	80.88 (w_{dur}=0.46, w_{Energy}=0.41, w_1 =0.7)
All templates (KL)	80.35	80.43 (w_1=0.78)	80.34 (w_1=0.84)	80.02 (w_1=0.94)	80.57 (w_{dur}=0.45, w_{Energy}=0.39, w_1 =0.82)
MDTS (LLR)	79.97	79.87 (w_3=0.69)	79.77 (w_3=0.75)	79.45 (w_3=0.86)	79.95 (w_{dur}=0.47, w_{Energy}=0.41, w_3 =0.76)

Table 7.4 Average word accuracies (%) of integrating template based methods and non-parametric prosodic scores for the other four doctors

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	80.83	81.14	81.04	80.83	81.22
All templates (KL)	80.37	80.62	80.53	80.24	80.75
MDTS (LLR)	80.07	80.14	79.97	79.6	80.20

7.4 Effects of Prosodic Information on Phoneme Recognition

In this section, we analyze how different phonemes are affected in recognition when prosodic information was used, and we provide an insight on which types of prosodic information help improve speech recognition accuracy. For the telehealth tasks, around 50 phones (including filled pauses) were used, but some of these 50 units had very few occurrences in the training set. Therefore, our analysis of the experimental results was based on 41 phones of the 50 units in which 39 came from the standard CMU phone definition in [65] and two other units were “sil” and “sp” which stand for the silence at the beginning and end of utterances and the pauses between words, respectively.

From the experimental results, we noticed that using only one type of prosodic information could not make much improvement for recognition accuracy (like duration) and it could even degrade recognition accuracy (like pitch). However, when three different types of prosodic information were combined together, they helped consistently improve recognition accuracy. Therefore, our analysis focuses on the phoneme accuracies when the three types of different prosodic information of duration, energy, and pitch were combined. We found that the recognition accuracies for most phonemes did not change much after prosodic information was integrated and there were only a few phonemes whose recognition accuracies improved or degraded significantly when using prosodic information. In order to show the trend of how prosodic information affects the phoneme accuracies, we list the three most improved phonemes and the three most degraded phonemes in Table 7.5 and Table 7.6 for the GMM and non-parametric based prosodic score calculation methods, respectively. From these two tables, we can observe that prosodic information had positive effects on vowel sound recognition. Actually, when we speak English, we follow the rule of "longer, louder and higher" [66] for vowel sound pronunciation in words. The three factors in this rule exactly correspond to the three types of prosodic information of duration, energy, and pitch, and this is also why they have positive effect on recognition of vowel sounds. On the other hand, prosodic information had negative effects on certain consonant phonemes since the consonant sounds listed in Table 7.5 and 7.6 belong to the stop and fricative consonant categories which do not have obvious prosodic characteristics. Therefore, prosodic information extraction for these kinds of consonant sounds was difficult, and using prosodic scores for these consonants may bring in more phoneme confusion.

Table 7.5 Three most improved and degraded phonemes in speech recognition after the integration of template based methods and GMM based prosodic scores

	Improved phonemes (Duration+Energy+Pitch)	Degraded phonemes (Duration+Energy+Pitch)
All templates (LLR)	AH, IY, OW	B, K, T
All templates (KL)	IH, OW, AW	B, G,SH
MDTS (LLR)	AH, AA, AO	D, P, S

Table 7.6 Three most improved and degraded phonemes in speech recognition after the integration of template based methods and non-parametric based prosodic scores

	Improved phonemes (Duration+Energy+Pitch)	Degraded phonemes (Duration+Energy+Pitch)
All templates (LLR)	AH, IH, IY	F, K, S
All templates (KL)	AA, IH, AO	D, K, P
MDTS (LLR)	AH, IY, AE	B, DH, S

Chapter 8
Conclusion

In this dissertation, we have formulated a novel approach of integrating template matching with statistical modeling to improve the accuracy of continuous speech recognition. The main contributions of this work are in the following five aspects:

1) A novel method was proposed to use GMM indices to represent speech frame vectors for template matching. The newly proposed method for template representation has both advantages of statistical modeling and template matching, which not only improves speech recognition accuracy but also saves lots of memory space and computation time compared with traditional frame vectors based templates.
2) Local distances of log-likelihood ratio and KL divergence were investigated and found successful for the proposed statistical modeling based template matching method.
3) Phonetic Decision Trees (PDT) was introduced to cluster triophone templates and to assign unseen triphone templates into known clusters, which solves the traditional unseen allophone problem in the template-based approach, and enables the template based methods to be used in LVCSR tasks.
4) The effectiveness of LLR based all template method was validated for TIMIT phone recognition based on rescoring the lattices generated by four different HMM baseline systems, which indicates that the LLR based all template method can consistently improve speech recognition accuracy on top of enhanced baseline systems.

5) Methods of template selection and compression based on the LLR and KL local distances were proposed. The KL based MLTS method significantly improved speech recognition accuracies and reduced computation and storage complexities, and the compressed templates produced further performance improvement.
6) An integration of the template based methods with prosodic information was explored for LVCSR. Three types of prosodic information of phone duration, phone energy, and phone pitch were studied. GMM based and non-parametric methods were proposed to calculate prosodic scores. Experimental results on the telehealth task show that prosodic information can help further improve speech recognition accuracy through improving vowel recognition.

The current work can potentially be extended in the future to further improve LVCSR by integrating the method of discriminative training with the current template matching approach. For example, discriminative training can be used to enlarge the distances among templates of different speech sound units to obtain discriminative templates. In addition, the templates that are used to calculate the template matching scores may be assigned different weights according to the degrees of similarity between the templates and each test segment to produce more reliable combined template matching scores.

APPENDIX

The experimental results obtained by integrating the template based methods with the GMM based prosodic scores (duration, energy, and pitch) are presented for the development set of Dr.1 in Table I to Table IV. The numbers of GMM components used were 1, 2, 4, 6 and 8. The weights in Eq.(7.3) and Eq.(7.4) for combining the prosodic information scores with template method based scores were equal (w_1= w_2=w_3=w_4=0.50), and the weights in Eq.(7.5) for combining the different prosodic information scores were also equal (w_{dur}=w_{energy}=w_{pitch}=1/3). From Table I to Table IV, we notice that, in general, when the number of GMM components was equal to 4, recognition accuracy peaked. Therefore, for the GMM based method, the number of GMM components was set to 4 for the other four doctors.

Table I Word accuracies (%) of integrating template based methods and GMM based duration scores for Dr.1

Duration	Baseline	1GMM	2GMMs	**4GMMs**	6GMMs	8GMMs
All templates (LLR)	80.67	80.63	80.66	**80.66**	80.37	80.25
All templates (KL)	80.35	80.05	80.11	**80.21**	79.74	79.68
MDTS (LLR)	79.97	79.83	79.86	**79.88**	79.68	79.22

Table II Word accuracies (%) of integrating template based methods and GMM based energy scores for Dr.1

Energy	Baseline	1GMM	2GMMs	**4GMMs**	6GMMs	8GMMs
All templates (LLR)	80.67	80.47	80.58	**80.65**	80.32	80.22
All templates (KL)	80.35	79.80	79.84	**80.13**	79.65	79.54
MDTS (LLR)	79.97	79.55	79.68	**79.85**	79.48	79.01

Table III Word accuracies (%) of integrating template based methods and GMM based pitch scores for Dr.1

Pitch	Baseline	1GMM	2GMMs	**4GMMs**	6GMMs	8GMMs
All templates (LLR)	80.67	79.84	80.20	**80.36**	79.92	79.66
All templates (KL)	80.35	79.37	79.72	**79.75**	79.42	78.83
MDTS (LLR)	79.97	79.12	79.26	**79.37**	79.01	78.67

Table IV Word accuracies (%) of integrating template based methods and GMM based prosodic (duration, energy and pitch) scores for Dr.1

Energy + Duration+Pitch	Baseline	1GMM	2GMMs	**4GMMs**	6GMMs	8GMMs
All templates (LLR)	80.67	80.65	80.77	**80.78**	80.42	80.31
All templates (KL)	80.35	80.17	80.18	**80.31**	80.01	79.98
MDTS (LLR)	79.97	79.92	79.95	**79.98**	79.88	79.68

Table V Word accuracies (%) of integrating template based methods and GMM based prosodic scores for Dr.2

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	85.98	86.10	85.95	85.51	86.15
All templates (KL)	85.38	85.26	85.23	84.78	85.46
MDTS (LLR)	85.03	85.10	85.06	84.56	85.15

Table VI Word accuracies (%) of integrating template based methods and GMM based prosodic scores for Dr.3

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	84.22	84.33	84.16	83.95	84.37
All templates (KL)	83.79	83.76	83.60	83.33	83.85
MDTS (LLR)	83.55	83.67	83.25	82.93	83.80

Table VII Word accuracies (%) of integrating template based methods and GMM based prosodic scores for Dr.4

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	73.53	74.02	73.88	73.98	74.49
All templates (KL)	73.20	73.59	73.60	73.39	73.67
MDTS (LLR)	72.90	73.05	72.94	72.81	73.20

Table VIII Word accuracies (%) of integrating template based methods and prosodic scores for Dr.5

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	75.74	76.06	75.94	75.57	76.11
All templates (KL)	75.26	75.39	75.35	74.98	75.51
MDTS (LLR)	74.94	75.03	74.90	74.38	75.09

Table IX Word accuracies (%) of integrating template based methods and non-parametric prosodic scores for Dr.2

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	85.98	86.18	85.99	85.54	86.20
All templates (KL)	85.38	85.39	85.30	84.97	85.55
MDTS (LLR)	85.03	85.08	84.92	84.41	85.12

Table X Word accuracies (%) of integrating template based methods and non-parametric prosodic scores for Dr.3

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	84.22	84.35	84.21	84.11	84.39
All templates (KL)	83.79	83.85	83.68	83.41	84.00
MDTS (LLR)	83.55	83.48	83.21	82.85	83.73

Table XI Word accuracies (%) of integrating template based methods and non-parametric prosodic scores for Dr.4

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	73.53	74.04	73.91	74.04	74.49
All templates (KL)	73.20	73.64	73.63	73.48	73.75
MDTS (LLR)	72.90	72.91	72.89	72.57	73.02

Table XII Word accuracies (%) of integrating template based methods and non-parametric prosodic scores for Dr.5

	Baseline	Duration	Energy	Pitch	Duration+Energy+Pitch
All templates (LLR)	75.74	76.09	76.01	75.69	76.16
All templates (KL)	75.26	75.47	75.42	75.09	75.60
MDTS (LLR)	74.94	75.00	74.73	74.31	75.01

BIBLIOGRAPHY

[1] Rabiner, L. and Juang, B., Fundamentals of Speech Recognition, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] Lavie, A., Waibel, A., et al., 1997. Janus-III: Speech-to-Speech Translation in Multiple Languages, Proc. of ICASSP, pp.21-24, Munich, Germany.

[3] Ostendorf, M., Digalakis, V. and Kimball, O.A., "From HMMs to segment models: a unified view of stochastic modeling for speech recognition," IEEE Trans. on SAP, Vol. 4, pp. 360-378, 1996.

[4] Gish, H. and Ng, K.., "Parametric trajectory models for speech recognition," Proc. of ICSLP, Vol. 1, pp. 466–469, 1996.

[5] Glass, J., "A probabilistic framework for segment-based speech recognition," Computer Speech and Language, Vol.17, pp.13-152, 2003.

[6] Deng, L., Yu, D. and Acero, A., "A long-contextual-span model of resonance dynamics for speech recognition: parameter learning and recognizer evaluation," Proc. of IEEE Workshop on ASRU, 2005.

[7] Baron, D., Shriberg, E., and Stolcke, A., "Automatic punctuation and disfluency detection in multi-party meetings using prosodic and lexical cues," in Proceedings of the International Conference on Spoken Language Processing, J. H. L. Hansen and B. Pellom, eds., vol. 2, Denver, Sept. 2002, pp. 949–952.

[8] Sankaranarayanan, A. and Shrikanth S., N., "Improved speech recognition using acoustic and lexical correlates of pitch accent in a N-best rescoring framework, " ICASSP2007, p.873-876.

[9] Aradilla, G., Vepa, J. and Bourlard, H., "Using pitch as prior knowledge in template-based speech recognition", Proc. Int. Conf. Acoust., Speech, Signal Process., vol.I, p.445 , 2006.

[10] De Wachter, M., Matton, M., Demuynck, K. and Wanbacq, P., “Template-based continuous speech recognition,” IEEE Trans. on ASLP, Vol. 15, No.4, May 2007.

[11] Ramasubramanian, V., Kulkarni, K., and Kammerer,B., “Acoustic modeling by phoneme templates and modified one-pass DP decoding for continuous speech recognition,” in Proc. ICASSP, Apr.2008, pp. 4105–4108.

[12] Kanevsky, D., Sainath, T. N., Ramabhadran, B.,and Nahamoo, D., “An analysis of sparseness and regularization in exemplar-based methods for speech classification,” in Proc. Interspeech, Sept. 2010, pp. 2842–2845.

[13] Seppi, D. and Van Compernolle, D., “Data pruning for template-based automatic speech recognition,” Proc. Interspeech, pp. 901-904, 2010.

[14] Sundaram, S. and Bellegarda, J. R., “Latent perceptual mapping: a new acoustic modeling framework for speech recognition,” Proc. Interspeech, pp. 881-884, 2010.

[15] Aradilla, G., Vepa, J. and Bourlard H., “Improving speech recognition using data-driven approach,” Proc. Eurospeech, pp. 3333-3336, 2005.

[16] Axelrod, S. and Maison, B., “Combination of hidden Markov models with dynamic time warping for speech recognition,” Proc. ICASSP, Vol. 1, pp. 173-176, 2004.

[17] Chen, X. and Zhao, Y., “Integrating MLP features and discriminative training in data sampling based ensemble acoustic modeling Data sampling based ensemble acoustic modeling,” Proc. Interspeech, pp.1349-1352, 2010.

[18] Deng, L. and O'Shaughnessy, D., Speech Processing-a Dynamic and Optimization-Oriented Approach, Marcel Dekker, 2003.

[19] Hermansky, H., "Perceptual Linear Predictive (PLP) Analysis of Speech," J. Acoustic Society of America, 87, pp. 1738-1752, 1990.

[20] Theodoridis, S. and Koutroumbas, K., Pattern Recognition. Academic Press, 1999.

[21] Haeb-Umbach, R. and Ney, H., "Linear discriminant analysis for improved large vocabulary continuous speech recognition," icassp, vol. 1, pp.13-16, Acoustics, Speech, and Signal Processing, 1992. ICASSP-92 Vol 1., 1992 IEEE International Conference on, 1992.

[22] Young, S., Evermann, Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Valtchev, V., and Woodland, P., 2009. The HTK Book. Cambridge.

[23] Hyvarinen, A. and Oja, E., "Independent Component Analysis: a Tutorial," http://www.cis.hut.fi/aapo/papers/IJCNN99_tutorialweb/

[24] Tomita, M., "An efficient augmented-context-free parsing algorithm," Computer Linguistics, 13 (1-2), pp.31-46, 1987.

[25] Jelinek, F., 1991. Up from trigrams! The struggle for improved language models. Proc. of Eurospeech, pp.1037-1040, Genoa, Italy.

[26] Woodland, P., Evermann, G., Gales, M., Hain, T., Liu, A., Moore,G., Povey, D., Wang, L., "CU-HTK April 2002 Switchboard System," Proc. of NIST Rich Transcription workshop, Vienna, VA, USA, 2002.

[27] Kneser, R. and Ney, H., "Improved backing-off for M-gram language modeling," Proc.of ICASSP, pp.181-184, Detroit, MI, USA, 1995.

[28] Chen, S. and Goodman, J., "An empirical study of smoothing techniques for language modeling," Technical Report, TR-10-98, Center for Research in Computing Technology,Harvard University, 1998.

[29] Katz, S.M., "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. ASSP-35, pp. 400-401, March, 1987.

[30] Ney, H. and Ortmanns, S., "Dynamic programming search for continuous speech recognition," IEEE Signal Processing Magazine, vol. 16, issues 5, pp. 64-83, 1999.

[31] Ortmanns, S., Ney, H. and Aubert, X., "A word graph algorithm for large vocabulary continuous speech recognition," Computer Speech and Language, 11(1), pp. 43-72, 1997.

[32] De Wachter, M., Demuynck, K., Wanbacq, P. and Van Compernolle, D., "A locally weighted distance measure for example based speech recognition," Proc. ICASSP, pp. 181-184, 2004.

[33] Maier, V. and Moore, R. K., "An investigation into a simulation of episodic memory for automatic speech recognition," Proc. Interspeech, pp. 1245-1248, 2005.

[34] Sakoe, H. and Chiba, S., "Dynamic programming algorithm optimization for spoken word recognition," IEEE Transactions on Acoustics, Speech and Signal Processing, 26(1):43–49, 1978.

[35] De Wachter, M., Example Based Continuous Speech Recognition, Ph.D. thesis, Katholieke Universiteit Leuven, ESAT, May 2007.

[36] Cutler, A. Dahan, D. and Van Donselaar, W., "Prosody in the comprehension of spoken language: A literature review", Language and Speech, 40 (2), pp. 141-202.

1997.

[37] Mehta, G. and Cutler, A., "Detection of target phonemes in spontaneous and read speech", Language and Speech, 31, pp. 135-156, 1988.

[38] Weintraub, M., Taussig, K. and Hunicke-Smith, K., and A. Snodgrass, "Effect of speaking style on LVCSR performance", in Proc. Inter. Conf. on Spoken Language Proc., pp.16–19, 1996.

[39] Chung, G. and Seneff, S., "A Hierarchical Duration Model for Speech Recognition Based on the ANGIE Framework," Speech Communication, 27, 113-134, 1999.

[40] Rao Gadde, V. R., "Modeling Word Duration," in Proc. Inter. Conf. on Spoken Language Proc., 1:601-604, 2000.

[41] Sun, X. and Evanini, K., "Gaussian mixture modeling of vowel durations for automated assessment of non-native speech," ICASSP 2011.

[42] Lee, S., Hirose, K. and Minematsu, N., "Incorporation of prosodic modules for large vocabulary continuous speech recognition," in Proc. ISCA Workshop on Prosody in Speech Recognition and Understanding, pg. 97-101, 2001.

[43] Stolcke, A., "Improvements to the SRI LVCSR system," presented at the NIST Rich Transcription Workshop, May 2002.

[44] Hirose, K., Minematsu, N. and Terao, M., "Statistical language modeling with prosodic boundaries and it use for continuous speech recognition," in Proc. Inter. Conf. on Spoken Language Proc., 2:937-940, 2002.

[45] Wang, C. and Seneff, S., "Prosodic scoring of recognition outputs in the JUPITER domain," in *Proc. ISCA Workshop on Prosody in Speech Recognition and Understanding*, pp. 151-156, 2001.

[46] Noth, E., Batliner, A., Kießling, A., and Kompe, R., "Verbmobil: The Use of Prosody in the Linguistic Components of a Speech Understanding System," *IEEE Trans. Speech and Audio Proc.,* 8(5):519-532, 2000.

[47] Ostendorf, M., Shafran, I. and Bates, R., "Prosody models for conversational speech recognition," in Proceedings of the 2nd Plenary Meeting and Symposium on Prosody and Speech Processing, 2003.

[48] Wang, C. and Seneff, S., "Lexical stress modeling for improved speech recognition of spontaneous telephone speech in the jupiter domain," in Proceedings of the 7th European Conference on Speech Communication and Technology, 2001.

[49] Hasegawa-Johnson, M., Cole, J., Shih, C., Chen, K., Cohen, A., Chavarria, S., Kim, H., Yoon, T., Borys, S. and Choi, J.-Y., "Speech recognition models of the interdependence among syntax, prosody and segmental acoustics," in Proceedings of HLT/NAACL, 2004.

[50] Seppi, D., Demuynck, K., and Van Compernolle, D., "Template-based automatic speech recognition meets prosody," Proc. Interspeech, 2011.

[51] Sun, X., and Zhao, Y., "New methods for template selection and compression in Continuous speech recognition," Proc. Interspeech, 2011.

[52] Hershey, J. R. and Olsen, P. A., "Approximating the Kullback-Leibler divergence between Gaussian mixture models," Proc. ICASSP, Vol. 4, pp. 317 – 320, 2007.

[53] Li, Y. and Li, L., "A greedy merge learning algorithm for Gaussian Mixture Model," Third International Symposium on IITA, 506–509, 2009.

[54] Xue, J., and Zhao, Y., "Random forests of phonetic decision trees for acoustic modeling in conversational speech recognition," IEEE Trans. Audio Speech

Language Process, vol. 16, no. 3, pp. 519–528, Mar. 2008.

[55] Zhao, Y., Zhang, X., Hu, R., Xue, J., Li, X., Che, L., Hu, R., and Schopp, L., “An automatic captioning system for telemedicine,” ICASSP, pp. I-957 – I-960, 2006.

[56] http://htk.eng.cam.ac.uk/.

[57] Stolcke, A., “SRILM – An extensible language modeling toolkit,” Proc. ICSLP, pp. 901-904, 2002.

[58] Kneser, R. and Ney, H., “Improved backing-off for m-gram language modeling,” Proc. ICASSP, pp. 181-184, 1995.

[59] Zhang, X., Zhao, Y., and Schopp, L., “A novel method of language modeling for automatic captioning in telemedicine,” IEEE Trans. Inf. Technol. Biomed., vol. 11, no. 3, pp. 332–337, May 2007.

[60] Sun, X. and Zhao, Y., “Integrate template matching and statistical modeling for speech recognition,” Proc. Interspeech, pp. 74-77, 2010.

[61] Duda, R. O., Hart, P. E., and Stork, D. G., Pattern Classification, 2nd edition, Stork John Wileyand Sons, Inc., 2000

[62] Zhu, Q., Stolcke, A., Chen, B., and Morgan, N., “Using MLP features in SRI’s conversational speech recognition system,” in Proc. ICSLP, vol. 2, pp. 921–924,2005.

[63] Evermann, G. & Woodland, P. C. (2000b). Posterior probability decoding, confidence estimation and system combination. Proceedings of the Speech Transcription Workshop, College Park.

[64] De Cheveigné, A. and Kawahara, H., “YIN, A fundamental frequency estimator for speech and music,” J. Acoust. Soc. Amer., vol. 111, no. 4, pp. 1917–1930, 2002.

[65] cmusphinx.svn.sourceforge.net/svnroot/cmusphinx/trunk/cmudict/cmudict.0.7a.phones

[66] Labov, W., Ash, S. and Boberg, C., The Atlas of North American English, Mouton de Gruyter, 2006.

VITA

Xie Sun was born on June. 27, 1979, in Shen Yang, Liao Ning, China. He received B.E. in Computer Science from Northwestern Polytechnical University at Xi’an, China, M.S. degrees in Computer and Network from École Nationale Supérieure d'Électronique, d'Électrotechnique, d'Informatique, d'Hydraulique et des Télécommunications (ENSEEIHT) at Toulouse, France, and Ph.d degree in Computer Science from University of Missouri at Columbia, Missouri, US. He is currently an ASR research scientist of Li Creative Technologies, Florham Park, New Jersey, USA.