# New Methods for Template Selection and Compression in Continuous Speech Recognition

*Xie Sun and Yunxin Zhao*

Department of Computer Science
University of Missouri, Columbia, MO65211 USA
xs8pb@mail.missouri.edu, Zhaoy@missouri.edu

## Abstract

We propose a maximum likelihood method for selecting template representatives, and in order to include more information in the selected template representatives, we further propose to create compressed template representatives by Gaussian mixture model (GMM) merging algorithm. A Kullback-Leibler (KL) divergence based local distance is proposed for Dynamic Time Warping (DTW) in template matching. Experimental results on the tasks of TIMIT phone recognition and large vocabulary continuous speech recognition demonstrated that the proposed template selection method significantly improved the recognition accuracy over the HMM baseline while only 5% or 10% templates were selected from the total templates, and the template compression method has provided further recognition accuracy gains over the template selection method.

**Index Terms**: template matching, template selection, KL divergence, DTW

10.21437/Interspeech.2011-405

## 1. Introduction

Template-based approaches to speech recognition describe speech spectral trajectories by speech segment formed from sequences of feature vectors. Unlike the traditional HMMs, template-based approaches commonly compare a test segment against stored templates and the reported recognition results are promising [1, 2, 3]. However, template-based methods have the shortcoming of being storage space intensive since they need to store the feature vectors of the full training set in computer memory, making them difficult to be used in large speech recognition tasks. Some newly proposed methods like template pruning and filtering [4] and template-like dimension reduction of speech observations [5] are helping to solve the problem. However, there is always a compromise between the costs in computation and space and the recognition accuracy.

In the current work, we propose novel methods for template selection and compression using KL divergence based template matching. We use GMM indices to represent the feature frame vectors in templates. The method of using GMMs to label frame vectors has been proposed in our previous work [6]. In the current work, we first investigate using the local distance measure of KL divergence for template matching, we then propose a maximum likelihood template selection algorithm to generate the GMM based template representatives, and we further integrate template information to create compressed template representatives. Compared with the case of using all templates, the newly proposed template selection algorithm significantly reduced the computation and storage complexities, and the compressed templates produced further performance improvement based on the selected template representatives. The proposed novel template selection and compression methods are effective in removing low performance templates, keeping and generating good quality template representatives, and therefore obtaining recognition performances at the same level of using all templates. We applied the newly proposed template selection and compression methods on the tasks of TIMIT phone recognition and large vocabulary speech recognition with positive results.

In Section 2, we describe the KL-divergence based template matching. In Section 3, we discuss how to select template representatives and create compressed template representatives. In Section 4, we describe the experimental setup. In Section 5, evaluation results are presented, and finally in Section 6, we give our conclusion and discuss future work.

## 2. Template matching using KL-divergence

### 2.1 Template construction and matching

We choose the template unit as context-dependent phone segments (triphone context). To construct the templates, we first carry out forced alignments of training speech data with their transcriptions 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. 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 template frame is therefore represented as:

$$x_t \rightarrow \left\{ \begin{bmatrix} m_{1(t)} \\ \vdots \\ m_{n(t)} \end{bmatrix} \begin{bmatrix} w_{1(t)} \\ \vdots \\ w_{n(t)} \end{bmatrix} \right\}, \text{ with } \sum_{i=1}^{n} w_{i(t)} \text{ normalized to 1.}$$

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 [7], and we adopt a symmetric constraint defined as:

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

The distance between two feature vector sequences $x$ and $y$ is calculated as:

$$D(x,y) = \frac{L}{N} \min_{\emptyset} \sum_{k=1}^{N} d(\emptyset_x(k), \emptyset_y(k)) \quad (2)$$

where $d$ is the 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.

### 2.2 KL-divergence based local distance

Conventionally, the distance between two frames in DTW is directly computed on speech feature vectors by using measures like the Euclidean distance or Mahalanobis distance [1]. In our template-based approach, since the GMM indices are used to represent the frame vectors, we 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 [8] to compute the distance from a GMM $m_x$ to a GMM $m_y$ as:

Copyright © 2011 ISCA

$$d(m_x \,||\, m_y) = \frac{1}{n}\sum_{i=1}^{n} \log \frac{m_x(x_i)}{m_y(x_i)} \quad (3)$$

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:

$$d_{KL}(m_x, m_y) = \frac{1}{2}(d(m_x \,||\, m_y) + d(m_y \,||\, m_x)) \quad (4)$$

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

$$d(x_t\,,\, y_{t'}) = \sum_{i=1}^{n_{x_t}} \sum_{j=1}^{n_{y_{t'}}} \left( w_{i(t)} w_{j(t')} d_{KL}(m_{i(t)}, m_{j(t')}) \right) \quad (5)$$

where $n_{x_t}$ and $n_{y_{t'}}$ are the numbers of GMMs used to label $x_t$ and $y_{t'}$, respectively and $w_{i(t)}$ and $w_{j(t')}$ are the tempate weights.

# 3. Template selection and compression

## 3.1 Maximum likelihood based template selection

Before template selection, we first use agglomerative hierarchical clustering to partition each tied triphone template class into several clusters, where the tied triphone template classes are inherited from the phonetic decision tree (PDT) produced tying structures of triphone states from baseline HMMs. Agglomerative clustering recursively merges two closest clusters into one cluster until only one cluster is left. The cluster distance function $D(C_i, C_j)$ is commonly defined by the distance of their elements $D(s_x, s_y)$. The average distance measure

$$D(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{s_x \in C_i} \sum_{s_y \in C_j} D(s_x, s_y) \quad (6)$$

is adopted in the current work. Note that $D(s_x, s_y)$ is the DTW distance of two templates defined by (2), and in this step of template clustering, the local distance $d$ is the KL distance between two GMMs defined by (5). We choose to use the minimum distance from a template to all other templates in a cluster $C_i$ as the criterion for selecting cluster template center $s^*$ as defined in (7) and (8):

$$D(s_x, C_i) = \sum_{\substack{s_x,\, s_y \in C_i \\ s_x \neq s_y}} D(s_x, s_y) \quad (7)$$

$$s^* = \text{argmin}_{s \in C_i} D(s, C_i) \quad (8)$$

In the algorithm we describe below, the cluster center $s^*$ is relabeled by a set of GMMs, and the selection of the GMMs follows a maximum likelihood criterion. So we call the template selection algorithm the maximum likelihood template selection (MLTS) algorithm. We use the DTW described in Section 2.1 to align all other templates in the cluster $C_i$ to the initialized template center $s^*$. Fig.1 illustrates an alignment result among the sequences $s^*, s_1, \ldots, s_N$ in $C_i$. The frame vectors $x_{t_1}, \ldots, x_{t_N}$ from the sequences $s_1, \ldots, s_N$, respectively, are aligned to the frame vector $x_t$ of the cluster center $s^*$. The following procedure describes the MLTS algorithm operating on the aligned frame vectors $X = \{x_t, x_{t_1}, \ldots, x_{t_N}\}$:

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-medoid algorithm with the KL distance to partition the GMM set $M$ into $l$ clusters $M_i,\ i = 1, \ldots, l$.
3. For $i = 1, \ldots, l$:
   Use the maximum likelihood criterion to select a GMM cluster center $M_i^*$ for $M_i$:
   $$M_i^* = \underset{M_i^j \in M_i}{\text{argmax}} \left( \sum_{x \in X_{M_i}} \ln p(x|M_i^j) \right) \quad (9)$$
   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 $i = 1, \ldots, l$:
   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., $p(X|M_i^*)$.
   $$w_i = \frac{p(X|M_i^*)}{\sum_{k=1}^{l} p(X|M_k^*)} = \frac{e^{\sum_{x \in X} \ln p(x|M_i^*)}}{\sum_{k=1}^{l} e^{\sum_{x \in X} \ln p(x|M_k^*)}} \quad (10)$$

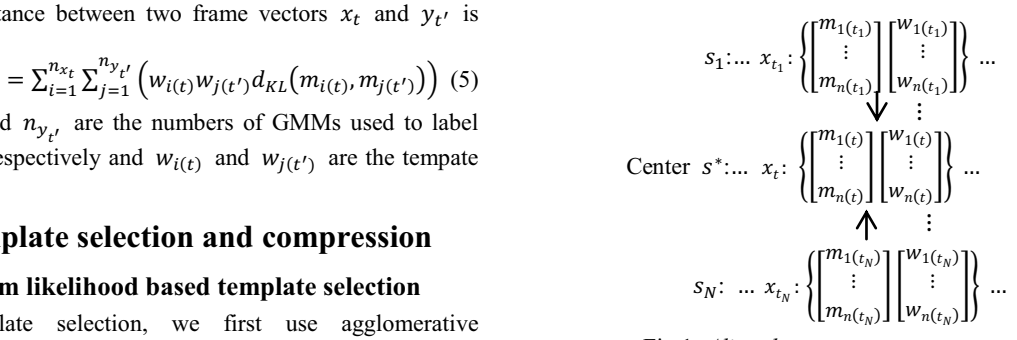


Fig.1: *Align the sequences* $s_1, \ldots, s_N$ to $s^*$

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 $x_t$ of $s^*$ is represented by $M_i^*$ and $w_i,\ i = 1, \ldots, l$, in the same form as that defined in Section 2.1. The difference is that we use the *n*-best GMMs to label a frame vector in Section 2.1, but here for a template representative the aligned multiple frame vectors are used to select a set of GMM indices for the frame labeling. The selected GMMs represent better the frame vectors in each cluster and they are thus more informative for the template representatives.

## 3.2 Template compression

Instead of keeping a cluster center GMM and excluding all other GMMs from a cluster in labeling a template representative frame, here 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 $\sigma$ 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 $M_i^j$ to the cluster center $M_i^*$. An outlier GMM $M_i^j$ which is $t$ times standard deviation away from $\bar{d}$, i.e.,

$$|d_i^j - \bar{d}| > t\sigma \quad (11)$$

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 is the smallest. The entropy increase is calculated as [9]:

$$\Delta E(f_1\,,\, f_2) = \log|\Sigma| - \frac{w_1}{w_1+w_2}\log|\Sigma_1| - \frac{w_2}{w_1+w_2}\log|\Sigma_2| \quad (12)$$

where $w_1$ and $w_2$ are the normalized mixture weights for $f_1$ and $f_2$, $\Sigma_1$ and $\Sigma_2$ are the diagonal covariance matrices of $f_1$ and $f_2$, and $\Sigma$ is the diagonal covariance matrix of the Gaussian density generated by merging $f_1$ and $f_2$. The mean $\mu$, covariance $\Sigma$, and mixture weight $w$ of the newly generated Gaussian component are defined as [9]:

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

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

$$w = w_1 + w_2 \quad (13)$$

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.

# 4. Experimental setup

## 4.1 Corpora

We performed phone recognition experiments on the tasks of TIMIT phone recognition and large vocabulary speech recognition. The TIMIT training set consisted of 3696 sentences from 462 speakers and the standard test set included 1344 sentences spoken by 168 speakers. The large vocabulary speech recognition task is on telehealth automatic captioning, which includes spontaneous speech from 5 doctors and the vocabulary size was 46K. Please refer to [10] for a detailed description of this task.

## 4.2 Experiment set up and lattice rescoring

For the TIMIT dataset, the phone set of 39 phones and a phone bi-gram language model (LM) were used. The HMM baseline was trained with the GMM mixture sizes of 24. Phone lattices were generated for each test sentence. The total number of triphone templates was 152715. For the telehealth task, speaker dependent acoustic models were trained for 5 healthcare provider speakers Dr. 1-Dr. 5. The baseline GMMs included 16 Gaussian components. The average number of triphone templates was 181601 per speaker for the 5 doctors. Trigram language models were trained on both in-domain and out-of-domain datasets, where word-class mixture trigram language models with weights obtained from forward weight adjustment were used [11]. Word lattices were generated for each test sentence with phone boundaries. For both tasks, 10000 Monte Carlo simulation data samples were generated to calculate a KL distance between two GMMs [8]. Speech features consisted of 13 MFCCs and their first and second order time derivatives. Crossword triphone models were trained and used in decoding search.

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 their average score as the match score.

# 5. Evaluation results

## 5.1 TIMIT phone recognition task

In Fig. 2, we compare the recognition performance for four different cases: 1) HMM baseline; 2) use all templates in rescoring; 3) use maximum likelihood template selection (MLTS) in rescoring; 4) use template compression in rescoring. In the case 1), the HMM baseline had the phone recognition accuracy of 72.72%, and from the baseline model 1189 GMMs were extracted for template generation in the cases 2) to 4). In case 4), the number of Gaussian components included in each GMM of the compressed templates was 24 and the total number of newly generated GMMs for the compressed template representatives was 749. When all templates were used, template matching obtained a phone accuracy of 74.18% which was slightly better than the accuracy performance of 74.13% of the MLTS algorithm that used only 5% template representatives, with the latter using significantly less computation time and storage space. When the template compression algorithm was used, recognition accuracy increased to 74.29%, where the threshold value in Eq. (11) was set to $t$=2. The number of GMMs used to label the template frame vectors for the cases 2) to 4) was 5, which was empirically determined to be the best choice on the TIMIT task [6]. We conducted a significance test (Student's $t$-test) on the performance differences between our proposed methods of ML template selection and compression and the HMM baseline, where the proposed methods improved the recognition accuracy significantly over the baseline at the significance level of $\alpha = 0.05$ (please refer to [6] for further details of the test method).

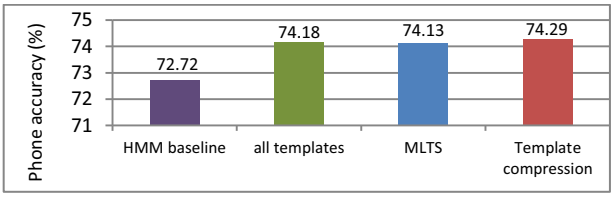


Fig.2: *Phone accuracies (%) for HMM baseline, all templates, MLTS and template compression*

In Table 1, we investigate how the threshold value $t$ of Eq. (11) for removing GMM outliers affected the recognition performance, where $t$=2 gave the best phone accuracy performance, 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 1. *Phone accuracies (%) from using different threshold values for the compressed template representatives*

| Threshold $t\sigma$ | 1σ | 2σ | 3σ | ∞ |
|---|---|---|---|---|
| Accuracy (%) | 73.98 | 74.29 | 73.47 | 70.98 |

In Fig. 3, we show how the percentages of the selected template representatives (from the total templates) affected the recognition accuracy for MLTS and template compression. In the case of 100% templates, MLTS and template compression were not used, and each frame vector was labeled by 5 GMMs as discussed above. For the rest of the percentage cases from 80% to 1%, the frame vectors in the original templates were labeled by 1 GMM only, and 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 template 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 that when the percentage was reduced from 100% to 60%, phone accuracy decreased since there were less templates to be used in rescoring and on average each frame vector of the template representatives was labeled by less than 2 GMMs. When the percentage changed 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. The best accuracy performances of 74.13% and 74.29% were obtained when only 5% selected and compressed template representatives were used. When the percentage reduced to 1%, the template representatives became insufficient and the accuracy performance dropped.

It is worth noting that we also evaluated the MLTS and the template compression methods based on labeling each frame vector of the original training templates by 5 GMMs. The accuracy performances for the high percentage cases of template representatives like 80% and 60% were better than those in Fig. 3 (for the MLTS method, the phone accuracies were 74.08% and 73.97%, respectively) since for each aligned frame vector set there were more GMM candidates which allowed each frame vector of the template representatives to be labeled by 5 GMMs. However, for the low percentage cases

like 10% and 5% their performances were inferior to those in Fig. 3 (for the MLTS method, the phone accuracies were 73.83% and 73.79%, respectively) due to a larger overlap of GMMs in the original frame vector labels which reduced discrimination among templates. Since the purpose of the proposed MLTS and template compression is to reduce computation and storage costs, the performance of the small percentage cases as shown in Fig. 3 is more relevant.

In Table 2, we provide the phone classification accuracies for MLTS and template compression with different percentages of template representatives of all templates, where the phone label boundaries defined in TIMIT were used in the classification. When 5% templates were selected as the template representatives, the MLTS and template compression methods obtained the best classification accuracies of 78.0% and 78.4%, respectively. From the pattern of phone classification accuracy, we observe that the phone classification accuracies keep a consistent trend with the phone recognition accuracies in Fig.3.

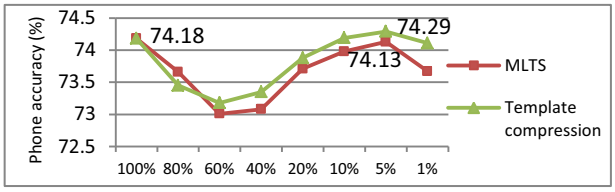


Fig.3: *Recognition accuracies (%) when the percentage of matching templates changed for MLTS* and template compression

Table 2. *Phone classification accuracies (%) for MLTS and template compression with different percentages of matching templates*

| Percentage | 100% | 80% | 60% | 40% |
|---|---|---|---|---|
| MLTS | 78.1 | 77.1 | 75.0 | 75.2 |
| Template compression | 78.1 | 76.3 | 75.4 | 75.9 |
| Percentage | 20% | 10% | 5% | 1% |
| MLTS | 77.1 | 77.6 | **78.0** | 76.7 |
| Template compression | 77.4 | 78.1 | **78.4** | 77.9 |

### 5.2 Large vocabulary speech recognition task

We further validated the proposed methods on the large vocabulary speech recognition task of telehealth captioning. On average, 1905 GMMs were extracted from the baseline HMMs for each of the 5 doctors for template generation. In Table 3, we compare the recognition accuracies for the 5 doctors over the four different cases as we did in Section 5.1. When all templates were used, template matching made a 1.15% improvement over the HMM baseline for the 5 doctors on average. When the MLTS method was used, the average performance slightly reduced to 79.44% while only 10% selected template representatives were used for template matching. When the compressed template representatives were used, the average recognition accuracy increased to 79.7%. The number of Gaussian components included in each compressed template was 16, and the average number of the newly generated GMMs for the compressed template representatives was 1048 per doctor. The GMMs were used to label the frame vectors in the same way as we did for the phone recognition task. Again, We conducted a significance test (Student's *t*-test) on the performance differences between our proposed methods of ML template selection and compression and the HMM baseline regarding the average word accuracy performance of the 5 doctors, where our proposed methods improved the recognition accuracy significantly over the HMM baselines at the significant level of $\alpha = 0.05$.

Table 3. *Word accuracies (%) for HMM baselines, all templates, MLTS and template compression*

| Speakers (# of word) | Dr.1 (3248) | Dr.2 (5085) | Dr.3 (3988) | Dr.4 (2759) | Dr.5 (6421) | Ave. |
|---|---|---|---|---|---|---|
| Baselines | 72.14 | 82.50 | 84.00 | 74.20 | 79.32 | 78.43 |
| All templates | 73.19 | 83.76 | 85.34 | 75.26 | 80.33 | 79.58 |
| MLTS | 73.11 | 83.55 | 85.08 | 75.21 | 80.24 | 79.44 |
| Template compression | 73.41 | 83.81 | 85.46 | 75.41 | 80.40 | 79.70 |

We also investigate how the percentage of template representatives affected the recognition accuracies for MLTS and template compression for the 5 doctors. When the percentage changed from 100% to 1%, the recognition accuracies for these two methods showed similar trends as in the phone recognition task, and the two methods obtained their best recognition performances of 79.44% and 79.70%, respectively, when 10% template representatives were used.

## 6. Conclusion and future work

In this paper, we have presented new algorithms for template selection and compression for statistical modeling based template matching in continuous speech recognition. Based on our experimental results from the TIMIT phone recognition and large vocabulary speech recognition tasks, we conclude that the proposed template selection and compression algorithms have significantly improved the recognition performance over the HMM baseline while largely saved computation time and memory space than using all templates, which is important for extending template matching methods to large speech recognition tasks. For future work, we shall evaluate adding additional information such as duration as an extra condition in selecting templates to further improve speech recognition accuracy.

## 7. References


[1] 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.

[2] 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.

[3] 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.

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

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

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

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

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

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

[10] Zhao, Y., et al., "An automatic captioning system for telemedicine," ICASSP, pp. I-957 – I-960, 2006.

[11] 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.