A SYMMETRIZATION OF THE SUBSPACE GAUSSIAN MIXTURE MODEL

Daniel Povey[1], *Martin Karafiát*[2], *Arnab Ghoshal*[3], *Petr Schwarz*[2]

[1] Microsoft Research, Redmond, WA, dpovey@microsoft.com
[2] Brno University of Technology, Czech Republic, {karafiat,schwarzp}@fit.vutbr.cz
[3] Saarland University, arnab.ghoshal@lsv.uni-saarland.de

ABSTRACT

Last year we introduced the Subspace Gaussian Mixture Model (SGMM), and we demonstrated Word Error Rate improvements on a fairly small-scale task. Here we describe an extension to the SGMM, which we call the symmetric SGMM. It makes the model fully symmetric between the "speech-state vectors" and "speaker vectors" by making the mixture weights depend on the speaker as well as the speech state. We had previously avoided this as it introduces difficulties for efficient likelihood evaluation and parameter estimation, but we have found a way to overcome those difficulties. We find that the symmetric SGMM can give a very worthwhile improvement over the previously described model. We will also describe some larger-scale experiments with the SGMM, and report on progress toward releasing open-source software that supports SGMMs.

Index Terms— Speech Recognition, Hidden Markov Models, Subspace Gaussian Mixture Models

1. INTRODUCTION

The Subspace Gaussian Mixture Model [1, 2] is a modeling approach based on the Gaussian Mixture Model, where the parameters of the SGMM are not the GMM parameters, but a more compact set of parameters that interact to generate the GMM parameters. The model may be described by the following equations:

$$p(\mathbf{x}|j,s) = \sum_{m=1}^{M_j} c_{jm} \sum_{i=1}^{I} w_{jmi} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{jmi}^{(s)}, \boldsymbol{\Sigma}_i) \quad (1)$$

$$\boldsymbol{\mu}_{jmi}^{(s)} = \mathbf{M}_i \mathbf{v}_{jm} + \mathbf{N}_i \mathbf{v}^{(s)} \quad (2)$$

$$w_{jmi} = \frac{\exp \mathbf{w}_i^T \mathbf{v}_{jm}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jm}}. \quad (3)$$

See the references for further explanation. The Gaussian mixture weights within the sub-states are controlled by the "weight projection vectors" $\mathbf{w}_i$ which determine how the weights vary as a function of the speech-state vectors $\mathbf{v}_{jm}$. The model is asymmetric because these weights only depend on the speech-state and not the speaker. In [2], we describe in detail how we efficiently evaluate likelihoods with such a model and estimate its parameters.

In this paper we describe a symmetric form of the SGMM. We modify Equation (3) to the following:

$$w_{jmi}^{(s)} = \frac{\exp(\mathbf{w}_i^T \mathbf{v}_{jm} + \mathbf{u}_i^T \mathbf{v}^{(s)})}{\sum_{i'=1}^{I} \exp(\mathbf{w}_{i'}^T \mathbf{v}_{jm} + \mathbf{u}_{i'}^T \mathbf{v}^{(s)})}, \quad (4)$$

Arnab Ghoshal was supported by the European Community's Seventh Framework Programme under grant agreement no. 213850 (SCALE); BUT researchers were partially supported by Czech MPO project No. FR-TI1/034.

where the vectors $\mathbf{u}_i \in \mathbb{R}^T$ (T is the speaker subspace dimension) now capture the effect of the speaker vectors on the weights. The difference is that how the mixture weights in the shared GMM structure can vary with the speaker as well as with the speech-state. Part of the motivation for this is that as shown by experiments reported in [1, 2], the fact that the weights vary with the speech-state (controlled by $\mathbf{w}_i$) is one of the most important features of the SGMM. However, symmetrizing the model like this brings up a few practical problems.

The first problem is how to efficiently evaluate likelihoods with this model. We address this issue in Section 2. Next we need to update the model parameters; in Section 3 we present the new accumulation and update equations, and the changes to the existing update equations. Space does not permit us to include derivations here; we have published some brief derivations in a separate technical report [3]. We present experimental results in Section 4 and in Section 5 we conclude and mention our progress toward releasing open-source software that implements these methods. The text between here and Section 5 will mainly be of interest to those already familar with the estimation methods used in SGMMs.

2. LIKELIHOOD EVALUATION

The new form of the weights introduces some difficulties for likelihood evaluation, since the denominator of Equation (4) has a difficult dependency on $\mathbf{v}^{(s)}$. Previously the log weights $\log w_{jmi}$ were included in normalizing factors stored for each Gaussian in the system (i.e. for each j, m, i). Recomputing all the weights from scratch every time we adapt to a new speaker would take an unacceptably long time. For example, with $100k$ substates, $I = 500$, $S = T = 40$ (S and T are the speech-state and speaker subspace dimensions) this computation would take 4 seconds at one GFlop. We make this faster by a factor of T, by storing in memory the unadapted weights w_{jmi} as in Equation (3), and computing the denominator of (4) as a dot product between these weights and some speaker-specific quantities. Storing the weights w_{jmi} does introduce a significant memory overhead; it can nearly double the size of the model in memory. There is, however, no significant additional time overhead, and in any case for large vocabulary systems the memory requirements tend to be dominated by the language model or recognition network.

In the rest of this section we write down the equations we use to evaluate likelihoods. For each speaker (and $1 \leq i \leq I$), we compute

$$b_i^{(s)} = \exp \mathbf{u}_i^T \mathbf{v}^{(s)}. \quad (5)$$

Then, for each j, m we compute the following normalizing factor:

$$d_{jm}^{(s)} = \sum_i w_{jmi} b_i^{(s)}. \quad (6)$$

We then have that $w_{jmi}^{(s)} = \frac{w_{jmi} b_i^{(s)}}{d_{jm}^{(s)}}$. We store $\log d_{jm}^{(s)}$ in memory. For each frame index t and each pre-selected Gaussian index i, we compute:

$$n_i(t) = \log|\det \mathbf{A}^{(s)}| - \tfrac{1}{2}\mathbf{x}_i(t)^T \boldsymbol{\Sigma}_i^{-1}\mathbf{x}_i(t) + \log b_i^{(s)}, \quad (7)$$

where only the last term $\log b_i^{(s)}$ is new (other quantities are as defined in [2]; c.f. Eq.(36)). The contribution to the likelihood from state j, mixture m and Gaussian index i is as follows (c.f. Eq. (37) of [2]; the last term is new):

$$\log p(\mathbf{x}(t), m, i|j) = n_i(t) + n_{jmi} + \mathbf{z}_i(t)\cdot\mathbf{v}_{jm} - \log d_{jm}^{(s)}. \quad (8)$$

3. MODEL ESTIMATION

We are able to obtain auxiliary functions with the same functional form as those we used to obtain the update equations previously reported in [2] (see [4] for the original derivations). The term $-\log d_{jm}^{(s)}$ in Equation (8) is the problematic new term. We used Jensen's inequality in a reverse sense to the way it is normally used, to move the log function out of a summation; see [3] for details.

3.1. Speaker vector estimation

The auxiliary function we use to optimize $\mathbf{v}^{(s)}$ is as follows:

$$\begin{aligned}\mathcal{Q}(\mathbf{v}^{(s)}) = {} & \mathbf{y}^{(s)\,T}\mathbf{v}^{(s)} + \sum_i \gamma_i^{(s)}\mathbf{u}_i^T\mathbf{v}^{(s)} \\ & -\frac{1}{2}\sum_i \mathbf{v}^{(s)\,T}\mathbf{N}_i^T\boldsymbol{\Sigma}_i^{-1}\mathbf{N}_i\mathbf{v}^{(s)} \\ & -\gamma^{(s)}\log\sum_i a_i^{(s)} b_i^{(s)} \end{aligned} \quad (9)$$

Here the statistics $a_i^{(s)}$ are a new quantity which we introduce here (the other terms are as previously described):

$$a_i^{(s)} = \sum_{t\in\mathcal{T}(s)}\sum_{j,m}\frac{\gamma_{jmi}(t) w_{jmi}}{d_{jm}^{(s)}}. \quad (10)$$

Note that $d_{jm}^{(s)}$ depends on the speaker vector $\mathbf{v}^{(s)}$; this is an iterative EM process where we start from $\mathbf{v}^{(s)} = \mathbf{0}$, so on the first iteration $d_{jm}^{(s)}$ would equal unity. Typically we just use one or two EM iterations.

The update for $\mathbf{v}^{(s)}$ is similar to the update for $\mathbf{v}_{jm}$ previously described, except that we use multiple iterations in the update phase (we do not bother with this while updating $\mathbf{v}_{jm}$, because it is part of a larger E-M process in which we do a large number of iterations). The iterations of speaker vector update are indexed p, with $1 \le p \le P$ (e.g., $P = 3$). We write the p'th iteration of the speaker vector as $\mathbf{v}^{(s,p)}$; if we are on the first iteration of the E-M process we would be starting from $\mathbf{v}^{(s,0)} = 0$ (or otherwise the previously estimated value). We first compute $\mathbf{H}^{(s)}$, which is the quadratic term in our "old" update:

$$\mathbf{H}^{(s)} = \sum_{i=1}^{I}\gamma_i^{(s)}\mathbf{N}_i^T\boldsymbol{\Sigma}_i^{-1}\mathbf{N}_i. \quad (11)$$

On the p'th iteration we compute the following quantities as the linear and quadratic terms in a local approximation to the auxiliary function:

$$\mathbf{g}^{(p)} = \mathbf{y}^{(s)} + \sum_{i=1}^{I}(\gamma_i^{(s)} - \gamma^{(s)}\tilde{w}_i^{(s,p-1)})\mathbf{u}_i - \mathbf{v}^{(s,p-1)}\mathbf{H}^{(s)} \quad (12)$$

$$\mathbf{F}^{(p)} = \mathbf{H}^{(s)} + \sum_{i=1}^{I}\gamma^{(s)}\tilde{w}_i^{(s,p-1)}\mathbf{u}_i\mathbf{u}_i^T. \quad (13)$$

The quantity $\tilde{w}_i^{(s,p)}$ is an appropriately averaged weight quantity computed given $\mathbf{v}^{(p)}$ as the speaker vector:

$$\tilde{w}_i^{(s,p)} \equiv \frac{a_i^{(s)}\exp\mathbf{u}_i^T\mathbf{v}_i^{(s,p)}}{\sum_i a_i^{(s)}\exp\mathbf{u}_i^T\mathbf{v}_i^{(s,p)}}. \quad (14)$$

The update equation on the p'th iteration is, ignoring the possibility of non-invertibility, $\mathbf{v}^{(s,p)} = \mathbf{v}^{(s,p-1)} + \mathbf{F}^{(p)\,-1}\mathbf{g}^{(p)}$, but for greater robustness we do as follows, where the solve_vec function is as defined in [2]:

$$\mathbf{v}^{(s,p)} = \mathbf{v}^{(s,p-1)} + \mathrm{solve\_vec}(\mathbf{F}^{(p)}, \mathbf{g}^{(p)}, 0, K^{\max}). \quad (15)$$

Note that there is the theoretical possibility of divergence here, but we do not check for it as we have not seen it happen in practice.

3.2. Speech-state vector and speech-state weight projection estimation

We now require an additional type of statistic in order to update the speech-state vectors $\mathbf{v}_{jm}$ and the speech-state weight projections $\mathbf{w}_i$. This will allow us to handle the term in the auxiliary function that comes from the denominator of (4). The statistics are:

$$a_{jmi} = \sum_{t,j,m,i}\frac{\gamma_{jmi}(t)}{d_{jm}^{(s[t])}} b_i^{(s[t])} \quad (16)$$

Here, $s[t]$ represents the speaker active on frame t. Note that the $b_i^{s[t]}$ quantities and the alignments $\gamma_{jmi}(t)$ will not have the same values as the corresponding quantities used to compute $a_i^{(s)}$ in Equation (10), because we will compute (16) on a different pass through the speaker's data, after $\mathbf{v}^{(s)}$ has been estimated.

In the update equations described in [2] for $\mathbf{v}_{jm}$ and $\mathbf{w}_i$, the quantity w_{jmi} appears. This needs to be replaced by a quantity which we write as $\tilde{w}_{jmi}$, which is an appropriately averaged form of the speaker-specific weights. The statistics a_{jmi} are used to compute this. We define

$$\tilde{w}_{jmi} = \frac{w_{jmi} a_{jmi}}{\sum_i w_{jmi} a_{jmi}}. \quad (17)$$

Whenever this quantity appears in the update equations it should always be computed given the most "updated" values available for $\mathbf{v}_{jm}$ and $\mathbf{w}_i$. This means that $\tilde{w}_{jmi}$ must be recomputed inside the loop over p used in [2] in the update of $\mathbf{w}_i$.

The modifications to the updates in [2] simply consist of replacing w_{jmi} with $\tilde{w}_{jmi}$ throughout. For $\mathbf{v}_{jm}$ this involves changing Eqations (58) and (59); for $\mathbf{w}_i$ it involves changing the auxiliary function of (68), and the update equations (71) and (72).

3.3. Speaker-space weight projection estimation: overview

We now describe how we estimate the speaker-space weight projection vectors $\mathbf{u}_i$. We experimented with two versions of the weight projection algorithm, which we call the "more exact" and "less exact" algorithms. Ideally we would like the estimation of $\mathbf{u}_i$ to be perfectly symmetric with the estimation of $\mathbf{w}_i$. The problem is that this requires us to have some per-speaker statistics available in the update phase. Although the amount of statistics we require for each speaker is fairly compact (just the vectors $\mathbf{v}^{(s)}$ and some count-like quantities of dimension $I \simeq 500$), we are concerned that for extremely large corpora these could become difficult to fit in memory during the update phase. For this reason we also experimented with a less exact version of the update for $\mathbf{u}_i$ that avoids storing any per-speaker quantities.

3.4. Speaker-space weight projection: more exact estimation

For the "more exact" estimation method, we need to store three kinds of quantities: $a_i^{(s)}$, $\mathbf{v}^{(s)}$ and $\mathbf{s}_i$. The first two are speaker-specific quantities which would have to be stored in the form of a list, one for each speaker. The count-like quantities $a_i^{(s)}$ are as given by Equation (10), although we would compute them given the fully-updated value of the speaker vector $\mathbf{v}^{(s)}$. The linear term $\mathbf{s}_i$ is:

$$\mathbf{s}_i = \sum_s \gamma_i^{(s)} \mathbf{v}^{(s)}. \tag{18}$$

The counts $\gamma_i^{(s)} = \sum_{t \in \mathcal{T}(s),j,m} \gamma_{jmi}(t)$ are already computed for some of the other update types descibed in [2]. In the update phase, we maximize the following auxiliary function:

$$\mathcal{Q}(\mathbf{u}_i) = \mathbf{u}_i^T \mathbf{s}_i - \sum_s a_i^{(s)} \exp \mathbf{u}_i^T \mathbf{v}^{(s)}. \tag{19}$$

The optimization process is an iterative one where on each iteration $1 \leq p \leq P$ we compute linear and quadratic terms $\mathbf{g}_i^{(p)}$ and $\mathbf{F}_i^{(p)}$ and maximize the corresponding quadratic approximation to the auxiliary function. On each iteration we check that the auxiliary function did not decrease.

The optimization procedure for a particular value of i is as follows: Set $\mathbf{u}_i^{(0)} \leftarrow \mathbf{u}_i$ (i.e. the value before update). For $p = 1 \ldots P$ (e.g. $P = 3$), compute:

$$\mathbf{g}_i^{(p)} = \mathbf{s}_i - \sum_s a_i^{(s)} \exp(\mathbf{u}_i^{(p-1)\,T} \mathbf{v}^{(s)}) \mathbf{v}^{(s)} \tag{20}$$

$$\mathbf{F}_i^{(p)} = \sum_s a_i^{(s)} \exp(\mathbf{u}_i^{(p-1)\,T} \mathbf{v}^{(s)}) \mathbf{v}^{(s)}\, \mathbf{v}^{(s)\,T} \tag{21}$$

Then the candidate new value of $\mathbf{u}_i^{(p)}$ is $\mathbf{u}^{\mathrm{tmp}} = \mathbf{u}_i^{(p-1)} + \mathbf{F}_i^{(p)\,-1} \mathbf{g}_i^{(p)}$, or more safely

$$\mathbf{u}^{\mathrm{tmp}} = \mathbf{u}_i^{(p-1)} + \mathrm{solve\_vec}(\mathbf{F}_i^{(p)}, \mathbf{g}_i^{(p)}, \mathbf{0}, K^{\max}) \tag{22}$$

with solve_vec as defined in [2], and then we do as follows: while $\mathcal{Q}(\mathbf{u}^{\mathrm{tmp}}) < \mathcal{Q}(\mathbf{u}_i^{(p-1)})$, with $\mathcal{Q}$ defined as in Equation (19), set

$$\mathbf{u}^{\mathrm{tmp}} \leftarrow \tfrac{1}{2}(\mathbf{u}^{\mathrm{tmp}} + \mathbf{u}^{(p-1)}). \tag{23}$$

Then, once the auxiliary function is no longer worse than before, we set set $\mathbf{u}^{(p)} \leftarrow \mathbf{u}^{\mathrm{tmp}}$. After the iteration over p is completed, we set $\hat{\mathbf{u}}_i \leftarrow \mathbf{u}_i^{(P)}$.

3.5. Speaker-space weight projection: less exact estimation

For the less exact version of the computation of the speaker weight projections, we avoid storing any lists of speaker-specific quantities and instead accumulate statistics sufficient to form a local quadratic approximation of the auxiliary function, which we directly maximize (without convergence checks) in the update phase. In this case we store the following statistics:

$$\mathbf{t}_i = \sum_s \left(\gamma_s^{(i)} - a_i^{(s)} b_i^{(s)} \right) \mathbf{v}^{(s)} \tag{24}$$

$$\mathbf{U}_i = \sum_s a_i^{(s)} b_i^{(s)} \mathbf{v}^{(s)}\, \mathbf{v}^{(s)\,T}. \tag{25}$$

The (weak-sense) auxiliary function we maximize is as follows, where Δ_i is the change in $\mathbf{u}_i$:

$$\mathcal{Q}(\Delta_i) = \mathbf{t}_i^T \Delta_i - \frac{1}{2} \Delta_i^T \mathbf{U}_i \Delta_i, \tag{26}$$

and our update equation is $\hat{\mathbf{u}}_i \leftarrow \mathbf{u}_i + \Delta_i$ with $\Delta_i = \mathbf{U}_i^{-1} \mathbf{t}_i$, or more generally, to handle the singular cases,

$$\hat{\mathbf{u}}_i \leftarrow \mathbf{u}_i + \mathrm{solve\_vec}(\mathbf{U}_i, \mathbf{t}_i, \mathbf{0}, K^{\max}), \tag{27}$$

with the function solve_vec as defined in [2].

4. EXPERIMENTAL RESULTS

We report experiments on CallHome English and Switchboard.

Our Callhome English setup is as described in [1, 2]. We used PLP features with ceptral mean and variance normalization. We tested with the trigram LM built as described in [2].

GMM:	52.5					
	#Substates					
	2700	4k	6k	9k	12k	16k
SGMM:	48.8	48.2	48.0	47.7	**47.4**	47.5
+spk-vecs:	47.6	47.0	46.4	46.4	46.1	**45.9**
+symmetric,exact:	46.3	45.6	45.2	44.8	44.5	**44.4**
+symmetric,inexact	46.5	45.6	45.0	44.6	**44.4**	

Table 1. CallHome English: WERs without CMLLR adaptation

Table 1 shows experiments without CMLLR adaptation; the only normalization is cepstral mean and variance normalization. Using the symmetric model reduced WER from 45.9% to 44.4%, a 1.5% absolute improvement. The inexact update gave the same improvement as the exact update.

GMM:	49.7					
+SAT:	46.0					
	#Substates					
	2700	4k	6k	9k	12k	16k
SGMM+spk-vecs:	46.5	45.5	45.2	45.4	44.8	**44.7**
+symmetric,exact	44.9	44.4	44.1	43.2	**42.8**	42.9
+symmetric,inexact	45.2	44.1	43.5	43.4	**43.3**	

Table 2. CallHome English: WERs with CMLLR adaptation

Table 2 shows experiments with CMLLR adaptation. The exact update gives 1.9% absolute improvement and the inexact update

gives 1.4% absolute improvement. Note that these are the same models as the previous table, tested with CMLLR, and we attribute the difference between the exact and inexact models on this setup to statistical noise; further experiments will have to tell us whether, in general, there is a difference between the exact and inexact updates.

GMM	#Gauss per state						
	20	26	32	34	36	38	40
-		36.8	36.6	36.4	36.4	36.4	36.4
CMLLR		34.8	34.5	34.4	34.3	34.5	34.3
STC		35.4	35.3		35.2		
+CMLLR		33.1	32.9		32.9		
SGMM	#Substates						
	30k	40k	50k	75k	100k	150k	200k
unadapted	35.7	35.7	35.1	34.7	34.3	33.9	33.7
CMLLR			32.2				
+spk-vecs	32.0	31.7	31.4	31.2	30.8		
+symmetric	31.9	31.7	31.3	31.0	30.6		

Table 3. Switchboard: WERs, with VTLN

GMM	#Gauss per state				
					36
-					39.2
CMLLR					37.0
STC					38.0
+CMLLR					35.2
SGMM	#Substates				
	30k	40k	50k	75k	100k
unadapted	37.9	37.5	37.1	36.6	36.3
CMLLR+spk-vecs	33.9	33.5	33.4		
+symmetric	33.8	33.0	33.2		

Table 4. Switchboard: WERs, no VTLN

Next we discuss Switchboard experiments. Our Switchboard system was trained on 278 hours of data from Switchboard I and II, and CallHome English. Models were tested on the Hub5 Eval01 test set (just over 6 hours long). We used PLP features with cepstral mean and variance normalization, and Vocal Tract Length Normalization (VTLN). The bigram language model used during decoding was taken from the AMI RT'07 system described in [5]; we used a recognition lexicon of 50K words. Our baseline GMM models were built with HTK [6]. Tables 3 and 4 show results with VTLN, and without VTLN, respectively. We did the baseline experiments with Constrained MLLR (CMLLR; a.k.a. fMLLR), and Semi-tied Covariance (STC; a.k.a. MLLT). With the SGMMs, we used the exact update for the $\mathbf{u}_i$ quantities in the symmetric case. In both cases the symmetric extension to the model gives a much smaller improvement than on the CallHome setup. We are not sure of the reason for this. Note that we do not show Speaker Adapted Training (SAT) results for the GMM baseline because we did not see an improvement (we tried SAT after STC which would anyway reduce the gains).

Table 5 compares the average acoustic likelihood in the three passes of decoding, which reveals the effect of the symmetric modification on the likelihood. The "baseline" rows are the SGMM with speaker vectors and CMLLR, but without the symmetric modification. As expected, the likelihood before adaptation is slightly worse (because there is a mismatch between the models, which were adaptively trained, and the data), but it gets better after adaptation. In both cases the likelihood improvement per frame, after adaptation, was about 0.1 (in natural-logarithm units). This makes it hard to interpret the differences in results between the CallHome and Switchboard setups, because the effect on the likelihoods is so similar. We intend to do further experiments on other data-sets to find which results are more typical.

		Decoding pass		
		1 (no-adapt)	2 +spk-vecs	3 +CMLLR
Call Home	SGMM+spk-vecs	-65.44	-63.62	-62.56
	+symmetric	-65.57	-63.50	-62.45
Switch-board	SGMM+spk-vecs	-60.07	-57.78	-58.86
	+symmetric	-60.17	-57.68	-56.76

Table 5. Acoustic likelihoods on the three test-time decoding passes

5. CONCLUSIONS

We have described a modification to the Subspace Gaussian Mixture Model which we call the Symmetric SGMM. This is a very natural extension which removes an asymmetry in the way the Gaussian mixture weights were previously computed. The extra computation is minimal but the memory used for the acoustic model is nearly doubled. Our experimental results were inconsistent: on one setup we got a large improvement of 1.5% absolute, and on another setup it was much smaller.

We would also like to report our progress on releasing open-source software that supports the SGMM modeling approach. An official announcement, with additional co-authors, will follow within the next year. We are developing an open-source (Apache-licensed) C++ speech recognition toolkit that uses the OpenFst library [7]. Most aspects of the toolkit are not related directly to SGMMs, but SGMMs will be one of the acoustic models the toolkit natively supports. Most likely the toolkit will already have been released by the time this is published.

6. REFERENCES

[1] D. Povey, Lukáš Burget, et al., "Subspace Gaussian Mixture Models for Speech Recognition," in *ICASSP*, 2010.

[2] D. Povey, Lukáš Burget, et al., "The Subspace Gaussian Mixture Model – a Structured Model for Speech Recognition," *Computer Speech and Language*, vol. 25, no. 2, pp. 404–439, 2011.

[3] D. Povey, "The Symmetric Subspace Gaussian Mixture Model," Tech. Rep. MSR-TR-2010-138, Microsoft Research, 2010.

[4] D. Povey, "Subspace Gaussian Mixture Models for Speech Recognition," Tech. Rep. MSR-TR-2009-64, Microsoft Research, 2009.

[5] T. Hain, L. Burget, J. Dines, G. Garau, M. Karafiat, M. Lincoln, J. Vepa, and V. Wan, "The AMI(DA) system for meeting transcription," in *Proc. Rich Transcription 2007 Spring Meeting Recognition Evaluation Workshop*, Baltimore, USA, May 2007.

[6] S. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (for version 3.4)*, Cambridge University Engineering Department, 2009.

[7] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: a general and efficient weighted finite-state transducer library," in *CIAA*, 2007.