Dirichlet Process Mixture of Mixtures Model for Unsupervised Subword Modeling

Michael Heck, *Member, IEEE,* Sakriani Sakti, *Member, IEEE,* and Satoshi Nakamura, *Fellow, IEEE*

***Abstract*—We develop a parallelizable Markov chain Monte Carlo sampler for a Dirichlet process mixture of mixtures model (DPMoMM). Our sampler jointly infers a codebook and clusters. The codebook is a global collection of components. Clusters are mixtures, defined over the codebook. We combine a non-ergodic Gibbs sampler with two layers of split and merge samplers on codebook and mixture level to form a valid ergodic chain. We design an additional switch sampler for components that supports convergence in our experimental results. In the use case of unsupervised subword modeling, we show that our method infers complex classes from real speech feature vectors that consistently show higher quality on several evaluation metrics. At the same time we infer fewer classes that represent subword units more consistently and show longer durations, compared to a standard DPMM sampler.**

***Index Terms*—acoustic unit discovery, Bayesian nonparametrics, Dirichlet process, Gibbs sampling, mixture of mixtures, unsupervised subword modeling**

I. Introduction

DIRICHLET process mixture models (DPMMs) [1], [2] are firmly established in pattern recognition and machine learning. Also known as infinite mixture models [3], they elegantly extend finite mixture models by the aspect of automatic model selection. This property makes them a popular tool for solving clustering tasks that are challenging with regards to estimating model complexity a priori. Several extensions to the original concept have been introduced over time, most notably hierarchical models [4], [5] and Dirichlet processes (DPs) with dependencies [6], [7].

DPMMs with *Gaussian* components gained increased interest in the field of low resource automatic speech processing, particularly as method for tackling the task of unsupervised subword modeling. The task is to infer acoustic units from raw audio data that are suitable to reliably represent human speech, i.e., that show low discriminability errors. DPMM samplers were used for subword model inference in an array of works related to the zero resource speech challenge [8]–[11]. The idea is that each Gaussian in a mixture model that was inferred from speech data is considered a separate acoustic class. Previous work by the authors [12]–[14] improved unsupervised subword modeling via DPMM sampling by unsupervisedly transforming the sampler's input.

A major impediment for producing better subword models however is the simplicity of the inferred model. It is a long standing modeling assumption that speech observations, i.e., feature vectors that belong to specific sound categories, are multimodally distributed [15]. In practice, Gaussian mixture models (GMMs) are well established to model acoustic units such as phones [16], [17], and methods such as continuous hidden Markov models (HMMs) make use of state-dependent GMMs as multimodal distributions to model emission probabilities of speech observations [15], [18]. It seems therefore an oversimplification to assume single unimodal distributions to be a good model representation for individual sounds. This assumption limits the inferred units to represent generally very short stationary sound phenomena [19].

DPMMs work very well when the clusters in a data set are unimodally distributed. But problems arise when clusters follow more complex, e.g., multimodal, distributions. In such cases, a model that fits unimodal distributions (e.g., single Gaussians) to clusters tends to over-fragment the feature space and to suggest more clusters than actually present. In other words, real clusters tend to be represented by multiple components, i.e., "sub-clusters". However, without dependency modeling in DPMMs, inferred "sub-clusters" are considered independent and the relations between them are lost. The consequence is that the inferred clusters do not reflect the actual structure in the data. To more accurately approximate multimodally distributed clusters, a model that assigns multiple mixture components to each cluster would be required [20].

We consider unsupervised subword modeling to be such a problem where the ability to infer multimodal clusters from a data set can provide models that represent the real underlying data distribution more accurately. The expectation is that a mixture of clusters, where each cluster is a mixture itself, should be a better representation of acoustic units with favorable characteristics. Specifically, one would expect the number of inferred classes to be lowered, the overall model size therefore be reduced. At the same time, average durations when classifying sequences of sounds should be longer. Sound representations should also be more robust across data of different speakers and show higher discriminability due to more natural modeling.

We develop a sampler for a Dirichlet process mixture of mixtures model (DPMoMM) to overcome the limitations of DPMMs and to enable the inference of a mixture of *multimodal* clusters. In our proposed DPMoMM, each cluster is a mixture of components, and the collection of clusters forms a global mixture of mixtures. Throughout this paper, we will use the following terms to describe our model. Each mixture in the global mixture of mixtures is called a *cluster*. Each cluster is a mixture of *cluster components*. Cluster components can be shared across clusters, which is why they exist on a global level. We borrow a term from automatic speech recognition and call the global collection of components that can be part of

M. Heck, S. Sakti and S. Nakamura are with Nara Institute of Science and Technology.

Manuscript received xxx xx, 201x; revised xxx xx, 201x.

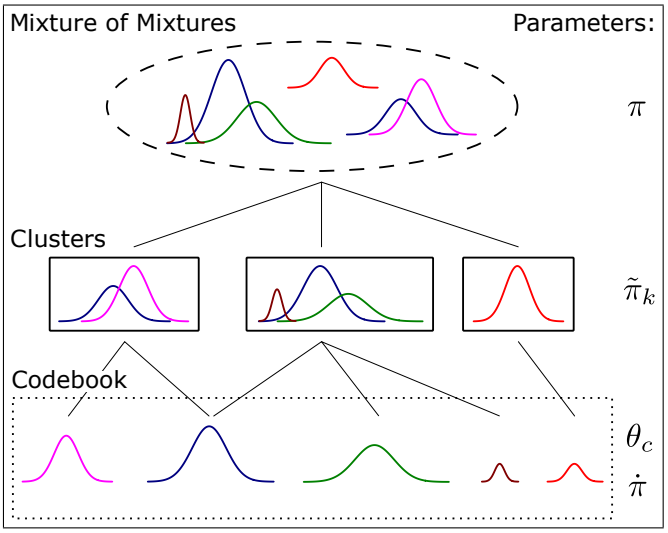


Fig. 1. Illustration of the mixture of mixtures model. The codebook is a global collection of components. Clusters are mixtures, defined over the codebook.

a cluster a *codebook*. When we speak of *codebook components* or simply *components*, we refer to the global components that make up the codebook. A *codebook component* is a cluster component (i.e., a member) in at least one cluster, and each *cluster component* is identical with exactly one codebook component. The difference is that the same component has different weights in different clusters (if it belongs to more than one). Besides that, each codebook component has a global weight. An intuitive illustration of this model is provided in Fig. 1. The detailed description of the model and its parameters is given in Section III.

We build on the idea of Chang et al. [21] and develop a split, merge and switch sampler with the following characteristics: (1) our sampler jointly infers a global codebook of components, and clusters which are mixtures that are defined over this codebook; (2) split and merge moves modify codebook components; (3) split, merge and switch moves modify cluster components; (4) all sampling steps can be parallelized across clusters and components; (5) the jointly inferred codebook and the mixture of mixtures provide two alternatives to model the same underlying data.

We demonstrate in unsupervised subword modeling use case experiments on real speech data that our DPMoMM sampler is superior to a DPMM in terms of inferring models that are better representations of the underlying data structure. Specifically, with our method we infer mixtures of *Gaussian* mixtures from real speech observations that consistently show higher quality on several subword model evaluation metrics. For that, we extract frame-wise speech feature vectors from a data set and use our proposed sampler to cluster these speech observations into classes. In a standard DPMM, each class is represented by a single Gaussian, whereas with our proposed DPMoMM, each class is represented with its own GMM. This way we infer fewer clusters that represent subword units more consistently across speakers and that show longer average durations. We also show that an additional switch sampler supports the convergence of the algorithm.

II. Related Work

The hierarchical Dirichlet process (HDP) is a well-known method for sampling mixture models that employ a hierarchy [4], [5]. The HDP can be used to infer *topics* that are shared between multiple *documents*, i.e., groups of data. An analogy between HDP and DPMoMM can be drawn to the following extent. A document in the HDP is a mixture of topics, just as a cluster in the DPMoMM is a mixture of components. Documents in the HDP share topics from a global set, just as clusters in the DPMoMM can share components from the codebook. The similarities between the models end at this point, however. The HDP differs greatly from the DPMoMM by assuming that a particular grouping of data into a finite set of documents is known a priori. Topics are assumed to be shared across documents, and each document is assumed to have its own particular distribution of these topics. Topic mixtures that describe individual documents can heavily overlap each other if they have many topics in common. In contrast, our proposed model does not assume a pre-defined grouping of data into document-like clusters. Instead, the DPMoMM sampler infers an unknown number of clusters within ungrouped data, comprised of an unknown number of components each. Our method infers a *mixture of mixtures*, i.e., a group of clusters which itself forms a mixture model with explicit mixture weights. In the DPMoMM, clusters are groups of neighboring components, and clusters do not tend to occupy the same regions in the feature space.

More related to our DPMoMM is the infinite mixture of infinite Gaussian mixtures (I^2GMM) by Yerebakan et al. [20]. The I^2GMM is a generative model that represents each cluster within a data set with its own mixture model. Here, a top layer DP defines meta-clusters, and lower layer DPs model the cluster data as a mixture of components. The top layer generates cluster parameters according to a base distribution H and defines the number and local expansion of clusters. The cluster parameters in turn define base distributions H_k for the lower layer which control the number and local expansion of cluster components. Covariance matrices are shared across components within the same cluster, leaving components to only differ in their means. In contrast to the I^2GMM, our proposed model does not define a prior over cluster appearances in form of meta-clusters. Instead, DPMoMM clusters can take on any structure that the data might inform. DPMoMM components also have their own covariance matrix each, which allows more natural approximations to the data. Further, unlike the I^2GMM, the DPMoMM supports the sharing of components across clusters, which enables the sharing of statistical strength [4].

Dependent Dirichlet processes are suitable to capture time dependencies between clusters or samples [22]–[24]. Temporal dependencies might in certain cases be reflected by locality in the feature space. With the DPMoMM, we propose a new kind of model sampler that explicitly infers a mixture of multimodal distributions to handle dependencies that are reflected by locality in the feature space.

Split and merge samplers are thoroughly discussed in an array of publications [25]–[27]. The DPMM sub-cluster al-

gorithm of Chang et al. [21] addresses several issues that previous approaches coped with. Their sampler combines a non-ergodic restricted Gibbs sampler and split and merge samplers into a valid Markov chain. The Gibbs sampler is restricted to non-empty clusters. Splits are proposed from sub-clusters that are learned jointly by deferred sampling. Moves are proposed with a Metropolis-Hastings (MH) algorithm. As instantiated-weight (IW) sampler, cluster weights are explicitly represented, as opposed to collapsed-weight (CW) samplers. Like in [28], [29], no finite approximations are used for the Dirichlet process, contrary to [29], [30]. The authors see the advantage of IW samplers in the possibility to parallelize across data points (which they refer to "inter-cluster parallelizable") and propose to use global split and merge moves to counter convergence issues. Inspired by these works, Chang et al. [21] propose moves that rely on jointly learned sub-clusters to reduce computational overhead during the MH steps.

This algorithm was used with success for unsupervised subword modeling in the scope of the zero resource challenge [8]. Chen et al. [10] inferred a DPMM from raw speech, with Gaussians as components, and used the Gaussian posteriorgrams extracted after sampling as new speech representation. In our own work, we extended this approach by developing a framework that unsupervisedly learns feature transformations from inferred classes [12]–[14]. We showed that these transformations in turn improve the input to the DPMM sampler so that even better classes can be inferred, according to a sound class discriminability measure. We further demonstrated that the inferred classes can be used to model sounds for speech recognition purposes [19]. However, we found there is need for a method to find more complex classes that are generalizing across speakers and that cover consistent sequences with longer durations. We show in this work how we enhance the DPMM to be a DPMoMM that jointly learns components and mixtures of components and how we successfully use our model to improve unsupervised subword modeling.

Discriminative (as alternative to generative) non-parametric models such as infinite (structured) support vector machines (i(S)SVMs) [31], [32] have also been successfully applied in the ASR domain to dynamically model speech concepts. The idea of the iSVM and iSSVM is to divide the feature space into regions to be handled by a mixture of experts, i.e., specialized sub-models, where the number of experts is inferred from the data and the mixture underlies a DP prior. However, the number of actual concepts to be modeled is known beforehand, and the SVM training is supervised and relies on labels. In contrast, our DPMoMM infers concepts from data only without prior knowledge of any sort.

III. DP Mixture of Mixtures Model

In this section, we develop our DP mixture of mixtures model (DPMoMM). Definitions and mathematical expressions are kept general and are not restricted to a specific type of mixture components. Within the scope of this work, we use our sampling algorithm to infer mixtures of Gaussian mixtures in the use case of unsupervised subword modeling, for which we originally developed this method (see Section VIII). We begin by reviewing the standard DPMM and the augmented DPMM of Chang et al. [21].

A. Graphical Representation

Fig. 2 is a representation of the general DPMM in plate notation. x_i is an observed data point i out of N data points, and z_i is the corresponding discrete label for that data point. π denotes the theoretically infinite dimensional vector of mixture weights. α is commonly referred to as the concentration parameter for the Dirichlet process, which governs the likelihood for new classes to be generated during sampling, and λ is the hyper-parameter for the Dirichlet process base measure. θ_k denotes the parameters of cluster k, e.g., mean and covariance in the case of Gaussians. The generative process of the DPMM is expressed as follows:

$$x_i \sim p(x_i|\theta_{z_i}), \qquad \theta_k \sim H(\lambda), \tag{1}$$

$$z_i \sim \text{Discrete}(\pi), \qquad \pi \sim \text{GEM}(\alpha), \tag{2}$$

where $\text{GEM}(\cdot)$ denotes the stick-breaking process, and H is the DP base measure. The generative story of a data point x_i is this. A discrete cluster label z_i is sampled from the set of all possible clusters, which are distributed according to the weights in π. Given the cluster label, x_i is drawn from the cluster with parameters θ_{z_i}.

Fig. 3 is Chang et al.'s [21] augmented DPMM using auxiliary variables. Each regular cluster is augmented with two explicit sub-clusters, denoted as l for "left" and r for "right". The goal is to design a model that is tailored toward splitting clusters. By picking suitable distributions for these sub-clusters, then they can provide good split proposals for their regular parent cluster. Each data point is assigned to either the "left" or "right" sub-cluster with a sub-cluster label $\bar{z}_i \in \{l, r\}$. The naming convention implies that the sub-clusters are designed towards separating the data points into distinct groups within the parent cluster. Sub-clusters have their own weights $\bar{\pi}_k = \{\bar{\pi}_k^l, \bar{\pi}_k^r\}$ and parameters $\bar{\theta}_k = \{\bar{\theta}_k^l, \bar{\theta}_k^r\}$. It is important to note that in this *auxiliary space* the data points x_i generate the labels $\bar{z}_i$, in contrast to the regular space where z_i generate x_i.

Our proposed DPMoMM is depicted in Fig. 4. As before, x_i is an observed data point i out of N data points that belong to a cluster k. β governs the global mixture proportions, and π is the vector of weights for clusters, sampled according to a stick-breaking process. z_i denotes the cluster assignment label for the corresponding data point. In our model, clusters are composed of components, which are represented by the following variables. $\tilde{c}_{ki}$ is the label of the cluster component conditioned on cluster k that the corresponding data point is assigned to. $\tilde{\pi}_k$ are the cluster component weights, governed by β and conditioned on cluster k. θ_c are the parameters for the component that generated x_i. All cluster components are defined globally in the *codebook*. The codebook is the weighted collection of all existing components in the DPMoMM, i.e., it is a global mixture of components. Given the codebook, sampling operations can be performed on component level, independent of their respective cluster memberships. $\dot{c}_i$ assigns

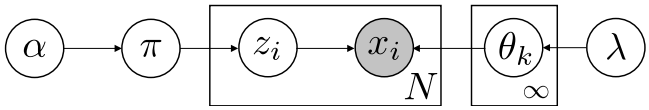


Fig. 2. Standard DPMM.

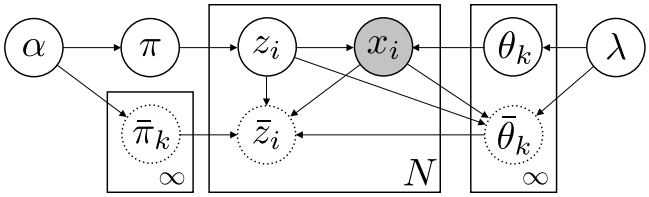


Fig. 3. Augmented DPMM by Chang et al. [21]. Auxiliary variables are denoted by dotted circles.

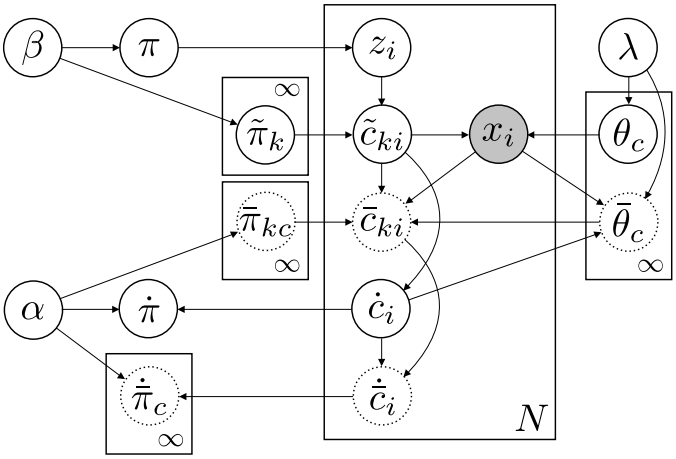


Fig. 4. Proposed DP mixture of mixtures model (DPMoMM). Auxiliary variables are denoted by dotted circles.

a data point i to a codebook component and is derived from the cluster component label $\tilde{c}_{ki}$. Codebook components have global weights $\dot{\pi}$, governed by a separate concentration parameter α. λ is again the hyper-parameter for the DP base measure. The generative process of the DPMoMM is formally expressed as follows:

$$x_i \sim p(x_i|\theta_{\tilde{c}_{z_i i}}), \qquad \theta_c \sim H(\lambda), \tag{3}$$

$$z_i \sim \text{Discrete}(\pi), \qquad \pi \sim \text{GEM}(\beta), \tag{4}$$

$$\tilde{c}_{ki} \sim \text{Discrete}(\tilde{\pi}_k), \qquad \tilde{\pi}_k \sim \text{GEM}(\beta), \tag{5}$$

$$\dot{c}_i = \tilde{c}_{z_i i}, \qquad \dot{\pi} \sim \text{GEM}(\alpha). \tag{6}$$

The generative story for any data point x_i in a DPMoMM is as follows. A cluster label z_i is sampled from the set of all possible clusters, which are distributed according to the weights in π. Then, a component label $\tilde{c}_{z_i i}$ is sampled from the set of components that belong to the cluster with label z_i, which are distributed according to the weights in $\tilde{\pi}_{z_i}$. Given the cluster and cluster component labels, x_i is drawn from the cluster component with parameters $\theta_{\tilde{c}_{z_i i}}$. The codebook is a by-product of this generative process. The codebook component labels are copies of the cluster component labels according to (6), and the weights of the codebook components are conditioned on these labels. Fig. 1 is an illustration of the hierarchy within a DPMoMM. The sampling of the full model is described in detail in Section IV.

An intuition of what the DPMoMM represents can be given as follows. Assuming the tackled task is topic clustering, one can view DPMoMM clusters as groups of closely related topics (modeled by cluster components). If for instance there are three components modeling the topics "cars", "trucks" and "motorbikes", then a cluster that contains these components would model the meta-topic "personal vehicles".

Similar to the augmented DPMM, we define an auxiliary space to enable a split and merge sampling approach. Each component in the DPMoMM is augmented with two sub-components, parametrized by $\bar{\theta}_c = \{\bar{\theta}_c^l, \bar{\theta}_c^r\}$, to provide good split candidates for a component split move proposal. $\bar{c}_{ki} \in \{l, r\}$ is the label that assigns the corresponding data point to a sub-component of $\tilde{c}_{ki}$ within cluster k, and $\bar{\pi}_{kc} = \{\bar{\pi}_{kc}^l, \bar{\pi}_{kc}^r\}$ denotes the weights for sub-components of component c within cluster k. On codebook level, sub-component labels $\dot{\bar{c}}_i$ are derived from the cluster sub-component labels $\bar{c}_{ki}$, i.e., $\dot{\bar{c}}_i = \bar{c}_{z_i i}$. $\dot{\bar{\pi}}_c = \{\dot{\bar{\pi}}_c^l, \dot{\bar{\pi}}_c^r\}$ are the sub-component weights for each codebook component, governed by α. The choice of the auxiliary parameter distributions follows Chang et al. [21], which is reflected in the way we sample these variables in Section IV.

Algorithm 1 DPMoMM sampling algorithm

```
Randomly initialize K_init clusters with 1 component each
while stop criterion not met do
    Propose cluster merges and splits                ▷ Sec. VI
    Propose cluster component switches               ▷ Sec. VI
    Propose component merges and splits              ▷ Sec. V
    for all clusters with split components do
        Duplicate and update                         ▷ Sec. V
    end for
    Sample parameters and labels                     ▷ Sec. IV
end while
```

B. Sampling Algorithm

Chang et al.'s DPMM sampler [21] is an instantiated-weight sampler that combines non-ergodic Markov chains into an ergodic chain and proposes splits from learned sub-clusters and merges of clusters. Their algorithm runs a Gibbs sampler, which samples the parameters and weights of each cluster and its sub-clusters, followed by sampling the cluster and sub-cluster labels for each data point. The Gibbs sampler is restricted to non-empty clusters. After Gibbs sampling, a split and merge sampler proposes with an MH algorithm to either split or merge clusters into new clusters. Gibbs sampling and split and merge sampling iterate until convergence or until a stop criterion is fulfilled.

Our proposed sampler uses a similar structure. Algorithm 1 is an outline of our algorithm in pseudo-code. We combine a restricted Gibbs sampler with a split, merge and switch sampler for clusters and a split and merge sampler for

components. The Gibbs sampler samples the parameters and weights of each codebook component and its sub-components, the weights of each cluster, and the weights of each cluster component. This is followed by sampling the cluster, component and sub-component labels for each data point. A split, merge and switch sampler proposes to either split or merge clusters or to move a cluster component from one cluster to another. A split and merge sampler for components proposes to either split or merge codebook components. Illustrations of the possible component and cluster moves are given in Fig. 6. Gibbs sampling, split, merge and switch sampling for clusters, and split and merge sampling for components iterate until convergence or until a stop criterion is fulfilled.

The Gibbs sampling steps and the split and merge moves for the codebook components are equivalent to the original DPMM sampler of Chang et al. [21], and the codebook together with the global component weights is exactly the model that the original sampler would infer.

The following sections explain in detail the individual non-ergodic samplers that make up our proposed algorithm. The non-restricted Gibbs sampler is explained in Section IV. Section V explains the component split and merge sampler which are technically identical with the sampler in [21], but applied to the codebook components. Because changes of the codebook components lead to changes of the clusters, we introduce novel update steps for clusters. These are explained accordingly in the respective subsections. Lastly, we propose cluster split, merge and switch moves in Section VI. Fig. 5 is an illustration of how our proposed DPMoMM algorithm behaves, compared to a DPMM.

IV. Restricted Gibbs Sampling

In this section we lay out the details of the restricted Gibbs sampler that we employ. The Dirichlet process uses an infinite length prior on the cluster labels z_i, cluster component labels $\tilde{c}_{ki}$ and codebook component labels $\dot{c}_i$. However, any label can only point to a finite number of entities, i.e., the clusters and the components that exist in any current state of the model. Because the restricted Gibbs sampler does not create new clusters and components itself, the dimensions of the infinite vectors π, $\tilde{\pi}_k$, $\dot{\pi}$ and θ are technically finite during Gibbs sampling. Posterior distributions of weights are conditioned on the assignments of data points. The *restricted* conditional distributions of the DPMoMM are

$$p(\pi|z,\beta) = \mathrm{Dir}(N_1,\ldots,N_K,\beta), \tag{7}$$
$$p(\tilde{\pi}_k|z,\tilde{c},\alpha,\beta) = \mathrm{Dir}(B_k^1,\ldots,B_k^C,\alpha), \tag{8}$$
$$p(\dot{\pi}|\dot{c},\alpha) = \mathrm{Dir}(\dot{N}_1,\ldots,\dot{N}_C,\alpha), \tag{9}$$
$$p(\theta_c|x,\dot{c},\lambda) \propto f(\{x\}_c,\theta_c)f(\theta_c,\lambda), \tag{10}$$
$$p(z_i=k|x,\pi,\theta) \propto \pi_k \sum_{c=1}^{C} \tilde{\pi}_k^c f(x_i,\theta_c), \tag{11}$$
$$p(\tilde{c}_{ki}=c|x,z,\tilde{\pi},\theta) \propto \tilde{\pi}_{z_i}^c f(x_i,\theta_c), \tag{12}$$
$$p(\dot{c}_i=c|x,\dot{\pi},\theta) \propto \dot{\pi}_c f(x_i,\theta_c), \tag{13}$$

with

$$N_k = \sum_{i=1}^{N} 1_{z_i=k}, \qquad N_k^c = \sum_{i=1}^{N} 1_{\substack{z_i=k\\ \tilde{c}_{ki}=c}}, \tag{14}$$

$$\dot{N}_c = \sum_{k=1}^{K} N_k^c, \qquad B_k^c = \begin{cases} \frac{N_k^c+\beta}{C_k} & \text{if } N_k^c>0,\\ 0 & \text{else,}\end{cases} \tag{15}$$

where K is the current number of non-empty clusters, C is the current number of non-empty codebook components and C_k is the current number of non-empty cluster components in cluster k. x is the vector of data points, z is the vector of cluster labels, $\tilde{c}$ is the vector of cluster component labels, $\dot{c}$ is the vector of codebook component labels, and $\tilde{\pi} = \{\tilde{\pi}_1,\ldots,\tilde{\pi}_K\}$. $\{x\}_c$ denotes all data points assigned to the global codebook component c. $f(\cdot)$ is a particular parametrized probability density function. E.g., $f(\{x\}_c,\theta_c)$ is the likelihood of the data subset $\{x\}_c$ given the cluster parameters θ_c, and $f(x_i,\theta_c)$ is the likelihood of the single data point x_i given θ_c. $1_{(\cdot)}$ is an indicator function that equals to 1 if the condition $(\cdot)$ holds true and is 0 otherwise.

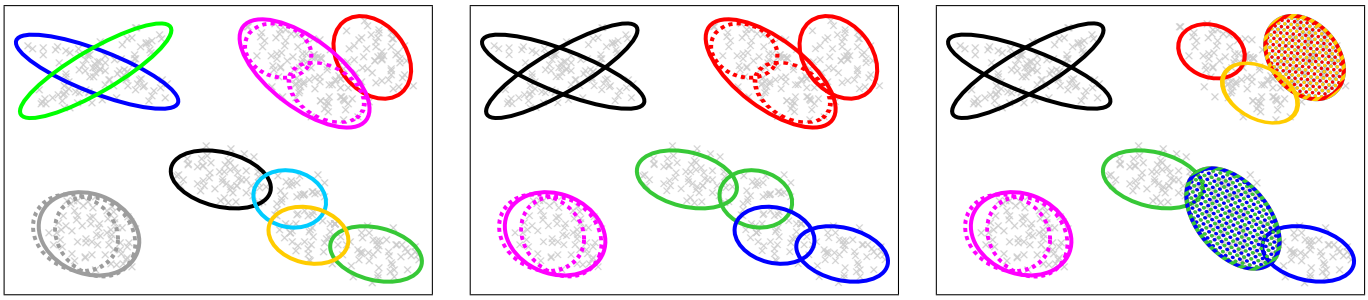
Given these probabilities, the sampling is as follows. Conditioned on the labels in the current state, sample the parameters of each codebook component, and all cluster and component weights. Conditioned on all cluster and component parameters in the current state, for each data point, sample a label for a cluster, then sample a label for a component within the cluster. The conditional distribution in (11) shows that the generative process described further above prefers cluster components to be neighbors in the feature space. During the model sampling described in this and the following sections, the algorithm will cause the clusters to be groups of nearby components so as to maximize the likelihood of the data.

For our proposed algorithm, we lay out the Gibbs sampler as follows. Given (7)-(13), the posterior distributions of weights, labels and component parameters are expressed as

$$(\pi_1,\ldots,\pi_K,\pi_{K+1}) \sim \mathrm{Dir}(N_1,\ldots,N_K,\beta), \tag{16}$$
$$(\tilde{\pi}_k^1,\ldots,\tilde{\pi}_k^C,\tilde{\pi}_k^{C+1}) \sim \mathrm{Dir}(B_k^1,\ldots,B_k^C,\alpha), \tag{17}$$
$$(\dot{\pi}_1,\ldots,\dot{\pi}_C,\dot{\pi}_{C+1}) \sim \mathrm{Dir}(\dot{N}_1,\ldots,\dot{N}_C,\alpha), \tag{18}$$
$$\theta_c \overset{\propto}{\sim} f(\{x\}_c,\theta_c)f(\theta_c,\lambda), \tag{19}$$
$$z_i \overset{\propto}{\sim} \sum_{k=1}^{K} \pi_k \left(\sum_{c=1}^{C} \pi_k^c f(x_i,\theta_c)\right) 1_{z_i=k}, \tag{20}$$
$$\tilde{c}_{ki} \overset{\propto}{\sim} \sum_{c=1}^{C} \pi_k^c f(x_i,\theta_c) 1_{\tilde{c}_{ki}=c}, \tag{21}$$
$$\dot{c}_i = \tilde{c}_{z_i i}, \tag{22}$$

with

$$\pi_{K+1} = 1 - \sum_{k=1}^{K} \pi_k, \tag{23}$$
$$\tilde{\pi}_k^{C+1} = 1 - \sum_{c=1}^{C} \tilde{\pi}_k^c, \tag{24}$$
$$\dot{\pi}_{C+1} = 1 - \sum_{c=1}^{C} \dot{\pi}_c. \tag{25}$$



(a) (b) (c)

Fig. 5. Illustration of the algorithm. Components are drawn with solid lines, sub-components with dotted lines. Only some exemplary sub-components are illustrated. *(a)*: Single component clusters inferred by a DPMM. This also corresponds to the codebook of the DPMoMM; *(b)*. Clusters inferred by a DPMoMM, where components of the same color belong to the same cluster; *(c)*: The same DPMoMM after a component split in the upper right cluster and a component merge in the lower right cluster. The original clusters are duplicated, and the bi-colored components are now shared across clusters.

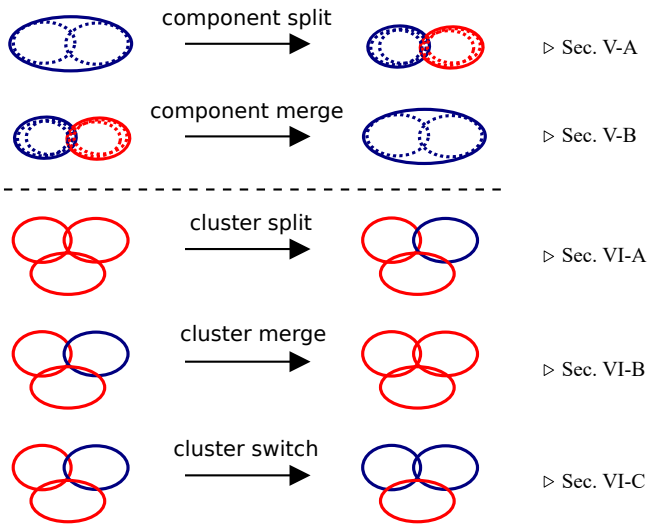


Fig. 6. Overview of the component moves (top) and cluster moves (bottom).

Note that the codebook component labels $\dot{c}_i$ are not sampled explicitly but derived from the cluster component labels. This is done to not break the assignment of data points to components on the global level; a data point that is assigned to component c within a cluster must also belong to component c within the codebook for the split and merge sampling logic to work properly.

For the split and merge sampling steps described in Section V, we make use of auxiliary variables that are jointly sampled with the regular variables. These auxiliary variables describe the sub-components which augment all the regular components. The sampled sub-components serve as good split candidates for eventual component splits. The auxiliary variables are sampled as follows:

$$(\bar{\pi}^l_{kc}, \bar{\pi}^r_{kc}) \sim \mathrm{Dir}(\frac{N^{c,l}_k + \alpha}{2}, \frac{N^{c,r}_k + \alpha}{2}), \tag{26}$$

$$(\dot{\bar{\pi}}^l_c, \dot{\bar{\pi}}^r_c) \sim \mathrm{Dir}(\frac{\dot{N}^l_c + \alpha}{2}, \frac{\dot{N}^r_c + \alpha}{2}), \tag{27}$$

$$\bar{\theta}^l_c \overset{\propto}{\sim} f(\{x\}^l_c, \bar{\theta}^l_c) f(\bar{\theta}^l_c, \lambda), \tag{28}$$

$$\bar{\theta}^r_c \overset{\propto}{\sim} f(\{x\}^r_c, \bar{\theta}^r_c) f(\bar{\theta}^r_c, \lambda), \tag{29}$$

$$\bar{c}_{ki} \overset{\propto}{\sim} \sum_{s\in\{l,r\}} \bar{\pi}^s_{\bar{c}_{ki}} f(x_i, \bar{\theta}^s_{\bar{c}_{ki}}) 1_{\bar{c}_{ki}=s}, \tag{30}$$

$$\dot{\bar{c}}_i = \bar{c}_{z_i i}, \tag{31}$$

with

$$N^{c,s}_k = \sum_{i=1}^{N} 1_{\substack{z_i=k\,,\\ \bar{c}_{ki}=c\\ \bar{c}_{ki}=s}} \qquad \dot{N}^s_c = \sum_{k=1}^{K} N^{c,s}_k, \tag{32}$$

where $\{x\}^l_c$ and $\{x\}^r_c$ denote the subsets of data points that are assigned to the left and right sub-components of c. Note that, analogous to (22), the labels $\dot{\bar{c}}_i$ are not sampled explicitly to not break the assignment of data points to sub-components on the global level.

V. Component Split and Merge Sampler

The split and merge moves for components are performed on the global codebook level and therefore rely on the codebook level variables that are jointly sampled with the other model variables. The moves are designed for efficiency by reducing the overhead of computational costs during the MH step and enabling parallelization across components. Components are equipped with auxiliary variables for sub-components. The parameters of the sub-components are sampled in the same fashion as the parameters for the regular "parent" components. Conveniently, samples for the variables of the regular components can be obtained by sampling the auxiliary variables, since we draw from a joint parameter space.

For performing split and merge moves for components with an MH-MCMC method, candidate moves, or proposals are required. Let $O = \{\dot{\pi}, \theta, \dot{c}\}$ be the set of component variables and $\bar{O} = \{\dot{\bar{\pi}}, \bar{\theta}, \dot{\bar{c}}\}$ be the set of sub-component variables. We propose a new set of random variables $\{\hat{O}, \hat{\bar{O}}\}$ for components and sub-components and compute the Hastings ratio of the form

$$HR = \frac{p(\hat{O}, x) p(\hat{\bar{O}}|x, \hat{c})}{p(O, x) p(\bar{O}|x, \dot{c})} \frac{q(O, \bar{O}|\hat{O}, \hat{\bar{O}})}{q(\hat{O}, \hat{\bar{O}}|O, \bar{O})}, \tag{33}$$

where q is called the proposal distribution and x denotes the collection of all observations. The Hastings ratio weights the state of the model before and after actually performing a move,

with the numerator standing for the post-move state and the denominator the pre-move state. As can be seen requires the Hastings ratio a reverse proposal to the proposed move. In the case of proposing a split move, that would be a merge, and vice versa. A proposed move is accepted with the probability

$$\min(1, HR). \tag{34}$$

A. Split Moves

The sub-components are utilized as good split move candidates for the MH algorithm. Proposing a split move is typically a non-trivial task. The construction of a prospective move is necessary for an MH framework, where a proposal is weighed against the status quo and accepted or rejected with a certain probability. Theoretically, any kind of split proposal can lead to an ergodic chain [21]. However, proposals with low probability of being accepted unnecessarily increase the computational load, since in case of a rejection, all previous computational efforts are in vain. Iterative fitting of sub-components with the help of the auxiliary variables introduced above circumvents the risk of wasted computational time. The sub-components are sampled jointly with the normal components. During the MH-step, sub-components pose good proposals for split moves. Moreover, split move computations can be parallelized across components.

The proposal distribution for proposing a component split move with the help of the auxiliary variables is defined as follows. First, a split or merge move $Q \in \{Q^c_{\text{c-split}}, Q^{m,n}_{\text{c-merge}}\}$ is selected randomly. $Q^c_{\text{c-split}}$ denotes a move for splitting component c into m, n, and $Q^{m,n}_{\text{c-merge}}$ is a merge of components m, n into c. New sets of model variables are sampled as follows, each conditioned on Q.

If $Q = Q^c_{\text{c-split}}$:

$$(\{\hat{\dot{c}}\}_m, \{\hat{\dot{c}}\}_n) = \text{split}_c(\dot{c}, \dot{\bar{c}}), \tag{35}$$

$$(\hat{\dot{\pi}}_m, \hat{\dot{\pi}}_n) = \dot{\pi}_c \cdot (\dot{\pi}^l_c, \dot{\pi}^r_c), \tag{36}$$

$$(\hat{\theta}_m, \hat{\theta}_n) \sim q(\hat{\theta}_m, \hat{\theta}_n | x, \hat{\dot{c}}, \hat{\dot{\bar{c}}}), \tag{37}$$

$$(\hat{\bar{O}}_m, \hat{\bar{O}}_n) \sim p(\hat{\bar{O}}_m, \hat{\bar{O}}_n | x, \hat{\dot{c}}), \tag{38}$$

with $(\dot{\pi}^l_c, \dot{\pi}^r_c) \sim \text{Dir}(\hat{\dot{N}}_m, \hat{\dot{N}}_n)$.

If $Q = Q^{m,n}_{\text{c-merge}}$:

$$\{\hat{\dot{c}}\}_c = \text{merge}_{m,n}(\dot{c}), \tag{39}$$

$$\hat{\dot{\pi}}_c = \dot{\pi}_m + \dot{\pi}_n, \tag{40}$$

$$\hat{\theta}_c \sim q(\hat{\theta}_c | x, \hat{\dot{c}}, \hat{\dot{\bar{c}}}), \tag{41}$$

$$\hat{\bar{O}}_c \sim p(\hat{\bar{O}}_c | x, \hat{\dot{c}}). \tag{42}$$

The function $\text{split}_c(\cdot)$ splits the label assignments of component c so that labels are assigned to the two new components m and n, whereas the function $\text{merge}_{m,n}(\cdot)$ does the reverse move and merges the label assignments of components m and n so that the respective data points are assigned to a new component c. Sampling new parameters given the new label assignments is done with the restricted Gibbs sampler. The sub-component auxiliary variables are sampled jointly with the variables for the regular components. This specific joint sampler is also called deferred MH sampler [21] and conveniently sets $q(\hat{\bar{O}}|x, \hat{O}) = p(\hat{\bar{O}}|x, \hat{O})$. Components are marked as "splittable" after the variables show signs of a burn-in, which is the case when the likelihood $f(\{x\}^l_c, \theta^l_c) f(\{x\}^r_c, \theta^r_c)$ for all data points assigned to component c begins to oscillate with the iterations. It can be shown [21] that the Hastings ratio for a component split can be expressed as

$$\begin{aligned} HR_{\text{c-split}} &= \frac{p(x|\hat{\dot{c}})}{p(x|\dot{c})} \frac{p(\hat{\dot{c}})}{p(\dot{c})} \frac{\alpha \Gamma(\hat{\dot{N}}_m)\Gamma(\hat{\dot{N}}_n)}{\Gamma(\dot{N}_c)} \\ &= \frac{f(x|\hat{\dot{c}}, \hat{\theta}_m) f(x|\hat{\dot{c}}, \hat{\theta}_n)}{f(x|\dot{c}, \theta_c)} \frac{\alpha \Gamma(\hat{\dot{N}}_m)\Gamma(\hat{\dot{N}}_n)}{\Gamma(N_c)}. \end{aligned} \tag{43}$$

Splits of codebook components affect the clusters that contain them as cluster components. All clusters that contain a split component require an update of their variables. The update step for clusters is subject to design, as there are various ways to execute it. The naive approach is to split the affected component within each cluster and update the corresponding variables. We opted for an update scheme that keeps the cluster sizes unchanged so as to separate codebook growth and mixture of mixtures growth entirely.

After performing a component split, we proceed as follows. All clusters that contain the respective component are duplicated with a function

$$(\{\hat{\tilde{c}}_k\}_c, \{\hat{\tilde{c}}_{k'}\}_c) = \text{duplicate}^c_k(\tilde{c}, \bar{c}), \tag{44}$$

where $\bar{c}$ is the vector of cluster sub-component labels.

The original cluster k keeps all unchanged components and the "left" split result. The duplicate k' also keeps all unchanged components and the "right" split result. Data points with $z_i = k$ that are assigned to the split component can be reassigned so that the ones labeled with $\bar{c}_{ki} = l$ belong to cluster k, and data points with and $\bar{c}_{ki} = r$ now belong to cluster k'. Data points that are assigned to unchanged components within the duplicated cluster however can not be reassigned unambiguously. Because a data point can never belong to two clusters at the same time, the duplication automatically invalidates the labels z_i and weights π. To re-establish a valid state for the sampler, we re-sample all variables. Therefore, the split sampler for components is the last step before the next iteration of the restricted Gibbs sampling in our implementation, as can be seen in Algorithm 1. Fig. 5b and 5c illustrate this step.

B. Merge Moves

The Hastings ratio for a proposed component merge also requires a reverse proposal. Where there is only one way to merge two sets of label assignments, there are $2^{\dot{N}_c - 1} - 1$ ways to split a set of labels into non-empty partitions. However, since split proposals in this algorithm are determined by the sub-components, the Hastings ratio will be zero if the labels after a proposed reverse split do not match the pre-merge labels. Therefore, the probability for accepting a component merge rapidly diminishes with increasing number of assigned data points. This behavior is approximated by automatically rejecting all component merges.

In order to mitigate for slow convergence in certain situations, a random merge sampler is introduced instead to propose component merge moves whose reverse move is a random split, in contrast to the sub-component based deterministic split proposals. Two random components are sampled and a merge proposal is computed. The reverse split proposal is generated by a random partitioning of the data points assigned to the respective component. The split proposal will generally have a diminishing acceptance probability, whereas the corresponding merge move is much more likely to be sensible. The Hastings ratio for a random merge proposal is as follows:

$$\begin{aligned} HR_{\text{c-merge}} &= \frac{p(x|\hat{\dot{c}})}{p(x|\dot{c})}\frac{p(\hat{\dot{c}})}{p(\dot{c})}\frac{\Gamma(\alpha)\Gamma(\dot{N}_m)\Gamma(\dot{N}_n)}{\Gamma(\alpha+\hat{\dot{N}}_c)\Gamma(\frac{\alpha}{2})^2} \\ &= \frac{p(x|\hat{\dot{c}})}{p(x|\dot{c})}\frac{\Gamma(\hat{\dot{N}}_c)}{\alpha\Gamma(\dot{N}_m)\Gamma(\dot{N}_n)}\frac{\Gamma(\alpha)\Gamma(\dot{N}_m)\Gamma(\dot{N}_n)}{\Gamma(\alpha+\hat{\dot{N}}_c)\Gamma(\frac{\alpha}{2})^2} \\ &= \frac{f(x|\hat{\dot{c}},\hat{\theta}_c)}{f(x|\dot{c},\theta_m)f(x|\dot{c},\theta_n)}\frac{\Gamma(\alpha)\Gamma(\hat{\dot{N}}_c)}{\alpha\Gamma(\alpha+\hat{\dot{N}}_c)\Gamma(\frac{\alpha}{2})^2}. \end{aligned} \quad (45)$$

A random split of component $\hat{\dot{c}}$ is sampled for the reverse split proposal, therefore the weights for the split results are Dirichlet distributed.

Analogous to the split moves, all clusters that contain a merged component need to be updated. Fortunately, the cluster updates after component merges are much simpler. All clusters that contained any of the two merged components replaces the respective component with the merge result. Cluster labels z_i and weights π remain unchanged. Cluster component labels $\tilde{c}_{ki}$ and weights $\tilde{\pi}_k$ are updated if cluster k contains both components involved in the merge:

$$\{\hat{\tilde{c}}_k\}_c = \text{merge}_{m,n}(\tilde{c}_k), \quad (46)$$

$$\hat{\tilde{\pi}}^c_k = \hat{\tilde{\pi}}^m_k + \hat{\tilde{\pi}}^n_k. \quad (47)$$

New values for cluster sub-component auxiliary variables $\bar{O}_k = \{\bar{\pi}_k, \bar{c}_k\}$ are sampled for all clusters to be updated according to

$$\hat{\bar{O}}^c_k \sim p(\hat{\bar{O}}^c_k|x,\hat{\tilde{c}}_k). \quad (48)$$

Again, Fig. 5b and 5c illustrate the outcome of this step.

VI. Cluster Split, Merge and Switch Sampler

Cluster split, merge and switch moves modify the assignments of components to clusters. A split move splits a cluster – which is a mixture of components – into two smaller mixtures. A merge move merges two clusters into one larger mixture. A switch move moves one component from one cluster to another cluster. In all these cases the components themselves, i.e., their parameters, are not modified. Cluster moves rely on the local, cluster dependent component parameters to produce good move proposals for a MH step. Cluster moves are efficient because they can be easily computed based on the existing data partitions.

Let $M = \{\pi, \tilde{\pi}, z, \tilde{c}\}$ be the set of cluster and cluster component variables and $\bar{M} = \{\bar{\pi}, \bar{c}\}$ be the set of cluster sub-component variables. A new set of random variables $\{\hat{M}, \hat{\bar{M}}\}$ is proposed by any of the possible moves. The Hastings ratio for a move is of the form

$$HR = \frac{p(\hat{M},x)p(\hat{\bar{M}}|x,\hat{\tilde{c}})}{p(M,x)p(\bar{M}|x,\tilde{c})}\frac{q(M,\bar{M}|\hat{M},\hat{\bar{M}})}{q(\hat{M},\hat{\bar{M}}|M,\bar{M})}. \quad (49)$$

As before, a proposed move is accepted with the probability defined in (34). Note that the component parameters θ_c are not subject to updates during cluster moves. That is because neither assignments of data points to codebook components nor codebook component parameters are modified.

A. Split Moves

Analogous to component splits, where good split candidates are provided by auxiliary sub-components, we resort to the cluster components themselves as support for producing good proposals. To generate a good split proposal for cluster k, we consider all $2^{C_k-1}-1$ possible non-empty partitions into two separate mixtures and propose the most promising split, according to the Hastings ratio.

The computational overhead grows with the number of components in a cluster. For instance, a cluster with 16 components can be partitioned into two clusters in 32.767 different ways, which would require a same amount of computations for Hastings ratios. In order to control the expected maximum computational load for splits, we introduce a parameter $c_{\max}$ into the algorithm which caps the maximum size of clusters. Setting this parameter accordingly prevents clusters to grow too large. By setting the value arbitrarily high, mixtures may grow to any size. To maintain computation feasible for very large clusters as well, we could also limit the considered partitions to a random subset of possibilities smaller than the Stirling number above.

The cluster split sampler's design essentially follows the considerations of the component split sampler in Section V-A. As is the case for components, we can also parallelize the cluster split move computations. The proposal distribution for a cluster split move is defined as follows. First, we randomly select a split move or a merge move $Q \in \{Q^m_{\text{k-split}}, Q^{a,b}_{\text{k-merge}}\}$, where $Q^m_{\text{k-split}}$ denotes a move for splitting cluster m into a, b, and $Q^{a,b}_{\text{k-merge}}$ is a merge of clusters a, b into m. New sets of model variables are sampled as follows, conditioned on Q.

If $Q = Q^m_{\text{k-split}}$:

$$(\{\hat{z}\}_a, \{\hat{z}\}_b) = \text{split}_m(z,\tilde{c}), \quad (50)$$

$$(\hat{\pi}_a, \hat{\pi}_b) = \pi_m \cdot (\pi^a_m, \pi^b_m), \quad (51)$$

$$(\hat{\tilde{\pi}}^1_a, \ldots, \hat{\tilde{\pi}}^C_a) \sim \text{Dir}(\hat{B}^1_a, \ldots, \hat{B}^C_a), \quad (52)$$

$$(\hat{\tilde{\pi}}^1_b, \ldots, \hat{\tilde{\pi}}^C_b) \sim \text{Dir}(\hat{B}^1_b, \ldots, \hat{B}^C_b), \quad (53)$$

with $(\pi^a_m, \pi^b_n) \sim \text{Dir}(\hat{N}^a_m, \hat{N}^b_m)$.

If $Q = Q^{a,b}_{\text{k-merge}}$:

$$\{\hat{z}\}_m = \text{merge}_{a,b}(z,\tilde{c}), \quad (54)$$

$$\hat{\pi}_m = \pi_a + \pi_b, \quad (55)$$

$$(\hat{\tilde{\pi}}^1_m, \ldots, \hat{\tilde{\pi}}^C_m) \sim \text{Dir}(\hat{B}^1_m, \ldots, \hat{B}^C_m), \quad (56)$$

with

$$\hat{B}^c_k = \begin{cases} \frac{\hat{N}^c_k+\beta}{C_k} & \text{if } \hat{N}^c_k > 0, \\ 0 & \text{else.} \end{cases} \quad (57)$$

The function $\text{split}_m(\cdot)$ splits the label assignments of cluster m so that labels are assigned to the two new clusters a and b, whereas the function $\text{merge}_{a,b}(\cdot)$ does the reverse move and merges the label assignments of clusters a and b so that the respective data points are assigned to a new cluster m. The component labels $\tilde{c}_{ki}$ do not require an update since component IDs are valid globally. To promote a more stable splitting behavior, clusters are marked "splittable" if all components within the respective clusters are also marked "splittable" (see Section V-A). We express the Hastings ratio for a cluster split proposal as follows:

$$\begin{aligned} HR_{\text{k-split}} &= \frac{p(x|\hat{z})}{p(x|\tilde{c})}\frac{p(\hat{z})}{p(z)}\frac{\Gamma(\beta+N_m)\Gamma(\beta\frac{\hat{N}_a}{N_m})\Gamma(\beta\frac{\hat{N}_b}{N_m})}{\Gamma(\beta)\Gamma(\hat{N}_a)\Gamma(\hat{N}_b)} \\ &= \frac{p(x|\hat{z})}{p(x|\tilde{c})}\frac{\beta\Gamma(\hat{N}_a)\Gamma(\hat{N}_b)}{\Gamma(N_m)}\frac{\Gamma(\beta+N_m)\Gamma(\beta\frac{\hat{N}_a}{N_m})\Gamma(\beta\frac{\hat{N}_b}{N_m})}{\Gamma(\beta)\Gamma(\hat{N}_a)\Gamma(\hat{N}_b)} \\ &= \frac{f(x|\hat{\tilde{c}},\hat{\Theta}_a)f(x|\hat{\tilde{c}},\hat{\Theta}_b)}{f(x|\tilde{c},\Theta_m)}\frac{\beta\Gamma(\beta+N_m)\Gamma(\beta\frac{\hat{N}_a}{N_m})\Gamma(\beta\frac{\hat{N}_b}{N_m})}{\Gamma(N_m)\Gamma(\beta)}, \end{aligned} \tag{58}$$

with $\hat{\Theta}_k = \{\hat{\pi}_k, \{\theta\}_k\}$, and $\{\theta\}_k$ being the parameters of all codebook components that are also cluster components in k.

B. Merge Moves

For a prospective merge, two random clusters are sampled and a merge proposal is computed. A cluster merge is only permitted if the component size of the merge result is not exceeding $c_{\max}$. The reverse proposal is a random partition of the cluster components into two separate mixtures. Analogous to the cluster split proposal above, the Hastings ratio for a cluster merge proposal is expressed as follows:

$$\begin{aligned} HR_{\text{k-merge}} = &\frac{f(x|\hat{z},\hat{\Theta}_m)}{f(x|z,\Theta_a)f(x|z,\Theta_b)} \\ &\times \frac{\Gamma(N_a+N_b)\Gamma(\beta)}{\beta\Gamma(\beta+N_a+N_b)\Gamma(\beta\frac{N_a}{\hat{N}_m})\Gamma(\beta\frac{N_b}{\hat{N}_m})}. \end{aligned} \tag{59}$$

As mentioned earlier, (11) suggests that the likelihood of a cluster is higher if the components of the respective cluster are located closer to each other in the feature space. During inference, β controls the importance of proximity for grouping nearby components into clusters. The Hastings ratios (58) and (59) suggest that with larger β the probability of accepting a split proposal becomes higher, and the probability of the reverse merge move becomes smaller.

C. Switch Moves

The component split and merge moves (see Section V) produce clusters with shared components due to the specifics of the cluster update steps. After a component split, all clusters that contain the split component are duplicated. After a component merge, all clusters who contained the previous components now share the new component (unless a cluster already contained both original components).

The number of clusters that share the same component can grow very quickly if the algorithm decides to perform many component splits. Another factor is the maximal cluster size $c_{\max}$. The larger clusters can get, the more likely they duplicate due to eventual component splits. In cases where a large amount of clusters overlap (that is, many clusters share the same components) and therefore cover the same region in the feature space, the algorithm can suffer from convergence issues due to high ambiguity during label sampling.

To mitigate this issue, we developed a switch move sampler that supports algorithm convergence. A switch move is an operation, where all data points that are assigned to component a in cluster m are re-assigned to the same component a in cluster n. Given a pair of clusters, we consider a switch move for a in both directions and propose the direction which is most promising, according to the Hastings ratio. Switch move proposals are sampled by a random switch sampler, where for a prospective split, two random clusters are sampled and the proposal is computed. New sets of model variables are sampled as follows:

$$(\{\hat{z}\}_m, \{\hat{z}\}_n, \{\hat{\tilde{c}}\}_m, \{\hat{\tilde{c}}\}_n) = \text{switch}^a_{m,n}(z, \tilde{c}), \tag{60}$$

$$(\hat{\pi}_m, \hat{\pi}_n) = (\pi_m - \pi_m\tilde{\pi}^a_m, \pi_n + \pi_m\tilde{\pi}^a_m), \tag{61}$$

$$(\hat{\tilde{\pi}}^1_m, \ldots, \hat{\tilde{\pi}}^C_m) \sim \text{Dir}(\hat{B}^1_m, \ldots, \hat{B}^C_m), \tag{62}$$

$$(\hat{\tilde{\pi}}^1_n, \ldots, \hat{\tilde{\pi}}^C_n) \sim \text{Dir}(\hat{B}^1_n, \ldots, \hat{B}^C_n). \tag{63}$$

The function $\text{switch}^a_{m,n}(\cdot)$ re-assigns all data points in component a of cluster m to component a of cluster n by updating the component labels $\tilde{c}_{ki}$ and the cluster labels z_i.

A switch move can be interpreted as splitting one component a off of a cluster m and merging a single-component cluster with a cluster n that already contains a as component. The Hastings ratio for a switch proposal is therefore expressed as follows:

$$\begin{aligned} &HR_{\text{switch}} \\ &= \frac{p(x|\hat{z})}{p(x|z)}\frac{\beta\Gamma(N^a_m)\Gamma(\hat{N}_m)\Gamma(N_n)}{\Gamma(N_m)\Gamma(N_n)}\frac{\Gamma(\beta+N_m)\Gamma(\beta+N_n)}{\Gamma(\beta)} \\ &\times \frac{\Gamma(\beta\frac{N^a_m}{N_m+N_n})\Gamma(\beta\frac{\hat{N}_m}{N_m+N_n})\Gamma(\beta\frac{N_n}{N_m+N_n})}{\Gamma(N^a_m)\Gamma(\hat{N}_m)\Gamma(N_n)} \\ &\times \frac{\Gamma(\hat{N}_m)\Gamma(\hat{N}_n)}{\beta\Gamma(N^a_m)\Gamma(\hat{N}_m)\Gamma(N_n)}\frac{\Gamma(\beta)}{\Gamma(\beta+\hat{N}_m)\Gamma(\beta+\hat{N}_n)} \\ &\times \frac{\Gamma(N^a_m)\Gamma(\hat{N}_m)\Gamma(N_n)}{\Gamma(\beta\frac{\hat{N}_n}{N_m+N_n})\Gamma(\beta\frac{\hat{N}_m}{N_m+N_n})} \\ &= \frac{p(x|\hat{z})}{p(x|z)}\frac{\Gamma(\hat{N}_m)\Gamma(\hat{N}_n)}{\Gamma(N_m)\Gamma(N_n)}\frac{\Gamma(\beta+N_m)\Gamma(\beta+N_n)}{\Gamma(\beta+\hat{N}_m)\Gamma(\beta+\hat{N}_n)} \\ &\times \frac{\Gamma(\beta\frac{N^a_m}{N_m+N_n})\Gamma(\beta\frac{N_n}{N_m+N_n})}{\Gamma(\beta\frac{\hat{N}_n}{N_m+N_n})}. \end{aligned} \tag{64}$$

The reverse move requires to split the data points that are assigned to component a in cluster n and assign these data points to cluster m. For the same reasons than in the case of normal component merges (see Section V-B), the probability for the reverse proposal quickly approaches zero with increasing data size and the reverse move will be rejected. We approximate this behavior by automatically rejecting all reverse switch moves.

VII. Use Case: Unsupervised Subword Modeling

In the following section, we will demonstrate the benefits of our DPMoMM sampler on real speech data in the use case of unsupervised subword modeling.

The objective of unsupervised subword modeling is to construct a representation of speech sounds that is robust to variation within and across speakers and that maximizes class discrimination [8]. Previous work of Chen et al. [10], [11] and ourselves [12]–[14] already achieved good results using a DPMM sampler to tackle this task in the context of the zero resource speech challenges [8], [9]. The general procedure is to cluster speech feature vectors into classes by sampling a Dirichlet process *Gaussian* mixture model (DPGMM) [5], which is an infinite Gaussian mixture model (IGMM) [3]. Speech would then be represented by frame-level posteriorgrams, for instance, or simply by a sequence of textual class labels, where the classes are the components in the sampled mixture.

The official evaluation metric for the zero resource speech challenge is the minimal pair ABX phone discriminability between phonemic minimal pairs [33]. Another popular metric for evaluating speech representations especially with respect to information and sequentiality is the normalized mutual information (NMI). We use both metrics to evaluate the output of our sampler in the following section. The details of the evaluation measures are explained in this section.

A. ABX Phone Discriminability

Let A and X be two speech representations of sound categories a, and B a speech representation of sound category b. The ABX phone discrimination error for categories a and b is

$$c(a,b) = 1 - \frac{1}{|a| \cdot |b| \cdot (|a|-1)} \sum_{A \in a} \sum_{B \in b} \sum_{X \in a \setminus \{A\}} \left(1_{d(A,X)<d(B,X)} + \frac{1}{2} 1_{d(A,X)=d(B,X)}\right) \quad (65)$$

where $d(\cdot,\cdot)$ is the dynamic time warping (DTW) distance defined over sequences of frame based speech representations (posteriorgrams, textual labels, etc.). Any proper distance measure can be used for computing the DTW distance. We evaluate two types of representations, textual labels and posteriorgrams. For the comparison of label sequences, we use the Levenshtein distance (ABX_{ls}). For evaluating posteriorgram sequences, we use the Kullback-Leibler divergence (ABX_{kl}).

We collect discrimination errors for all possible pairings of phone triplets and average them over all contexts for a given pair of central phones, over all pairs of central phones and over all speakers.

B. Symmetric Normalized Mutual Information

The mutual information of two random variables is a measure for mutual dependence. Normalized mutual information is defined as

$$\text{NMI}(X;Y) = \frac{H(X) - H(X|Y)}{H(X)} \quad (66)$$

and gives a measure of how good X can be predicted, given the knowledge about Y. In our use case, X corresponds to the random variable over the "true" distribution of sounds, approximated by the sequence of phones for any target data set. Y is the random variable over the estimated distribution of sounds, given for instance by the label output of a DPMM sampler for the same data. $H(X)$ is the entropy of the true transcription and is used as normalizing factor in the denominator.

$\text{NMI}(\cdot)$ is not symmetric, which makes the comparison of random variables that are defined over very different inventory sizes difficult. To guarantee a fair comparison, we suggest to use a symmetric NMI [34] of the form

$$\text{NMI}_{\text{max}}(X;Y) = \frac{H(X) - H(X|Y)}{\max(H(X), H(Y))} \quad (67)$$

With this, we have a fair measure of correlation at hand for the frame-level phone transcriptions of any target data and the frame-level label output of a DPMM or DPMoMM.

VIII. Experimental Evaluation

A. Data

We use two separate data sets known from the zero resource speech challenge [8]. Because the sets vary in size, language and speech quality, comparable experimental results should be a good indicator for the robustness of our sampler. One set contains spontaneous, conversational interview-style American English (4h 59min), extracted from the Buckeye corpus [35]. The other set contains carefully uttered, read speech in Xitsonga (2h 29min), a southern African Bantu language. The latter is an excerpt of the NCHLT corpus [36]. The references for English and Xitsonga contain 165k and 72k phone-triplet annotations for 39 and 53 unique center phones, respectively.

From each of the two data sets, we extract two sets of speech observations. Specifically, from each data set, we extract two types of frame-wise feature vectors using the Kaldi speech recognition toolkit [37]. We can use about 1.7M frames for English and 0.8M frames for Xitsonga as input to the DPMM and DPMoMM samplers. The frame width is 25 milliseconds and the frame shift is 10 milliseconds. The first type of features is perceptual linear predictive (PLP) speech feature vectors [38] with first and second order derivatives (PLP+Δ+$\Delta\Delta$). The second type is stacked PLP vectors that were transformed unsupervisedly by linear discriminant analysis (PLP+LDA) with the method described in [13]. LDA is a commonly used in speech recognition to optimize speech features towards discriminability [39]. We conduct experiments for each of the two input feature vector types. Overall, we conduct all our experiments four times, once for each data set and feature vector type.

B. Procedure

We use our DPMoMM sampler to cluster a set of speech feature vectors into classes. In our use case, our algorithm jointly samples *Gaussian* mixtures as well as a global codebook of Gaussians. This codebook is precisely what the original DPMM sampler of Chang et al. [21] would infer. Due

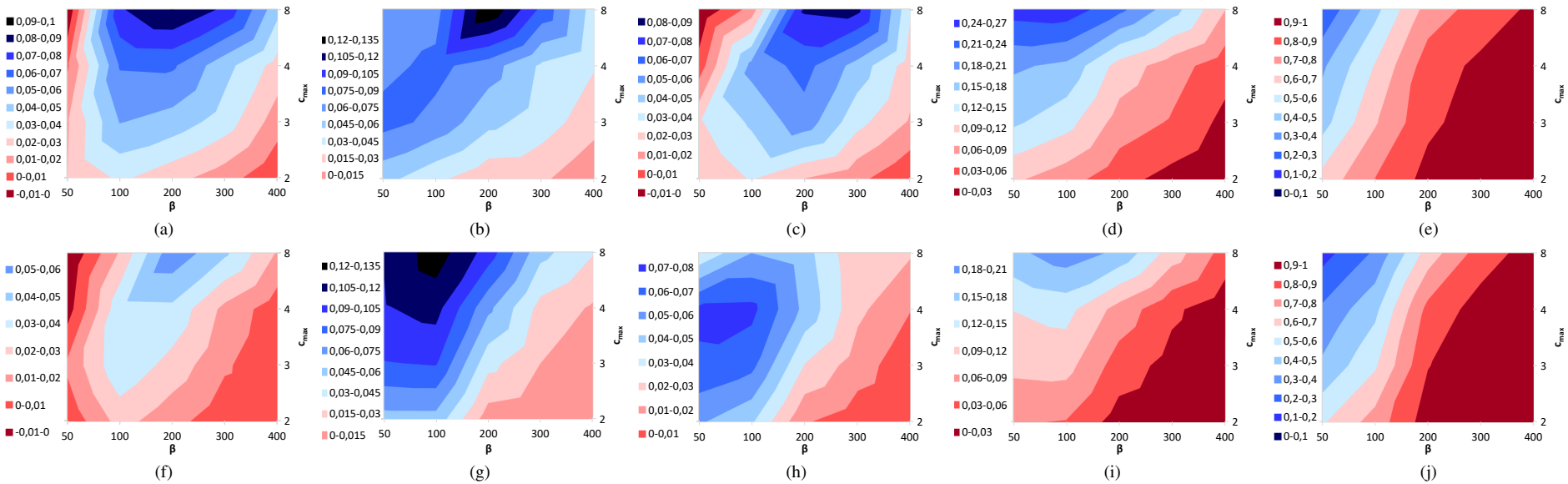


Fig. 7. Relative performance improvements when interpreting clusters instead of codebook components as acoustic classes. The inputs are 39 dimensional PLP+Δ+ΔΔ speech features. *(a)-(e)*: Results on Xitsonga; *(f)-(j)*: Results on English; *(a),(f)* Relative improvement of NMI_{max}, *(b),(g)* Relative improvement of ABX_{ls}, *(c),(h)* Relative improvement of ABX_{kl}, *(d),(i)* Relative unit length increase, *(e),(j)* Cluster to component amount ratio. Paired t-tests on all ABX task outputs yield $p \ll 0.0001$.

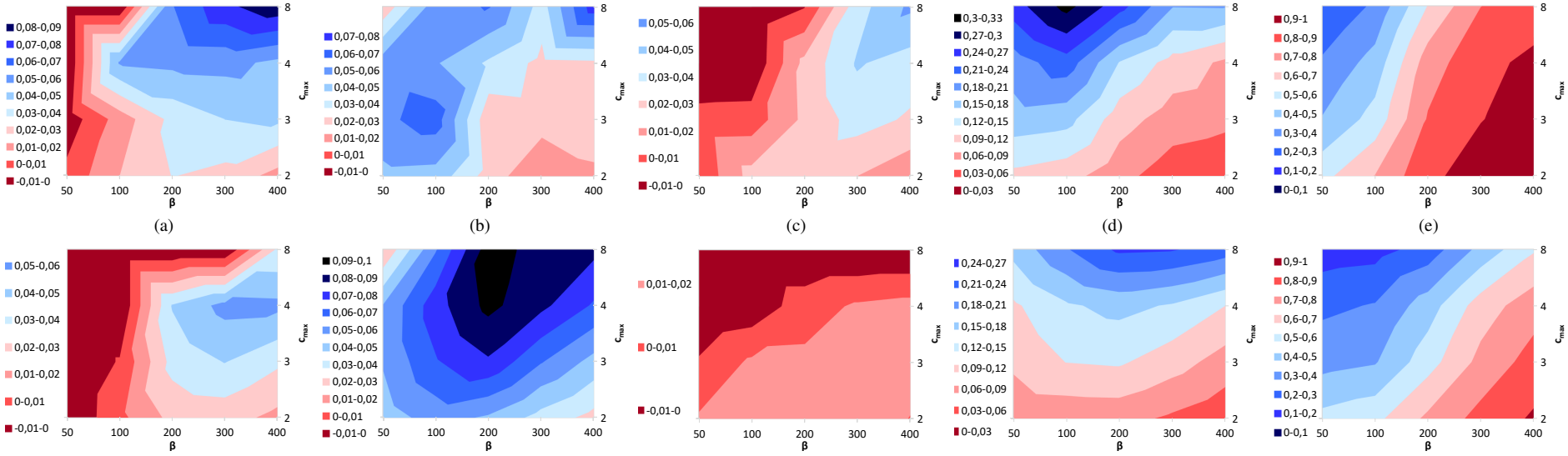


Fig. 8. Relative performance improvements when interpreting clusters instead of codebook components as acoustic classes. The inputs are 20 dimensional PLP+LDA speech features. *(a)-(e)*: Results on Xitsonga; *(f)-(j)*: Results on English; *(a),(f)* Relative improvement of NMI_{max}, *(b),(g)* Relative improvement of ABX_{ls}, *(c),(h)* Relative improvement of ABX_{kl}, *(d),(i)* Relative unit length increase, *(e),(j)* Cluster to component amount ratio. Paired t-tests on all ABX task outputs yield $p \ll 0.0001$.

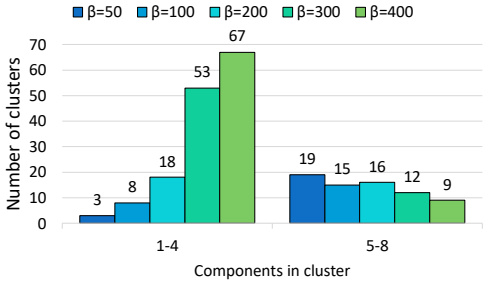


Fig. 9. Distribution of cluster sizes by the example of sampling Xitsonga data for 500 iterations. Higher β values lead to more clusters with fewer components. The same behavior was observed on the English data set.

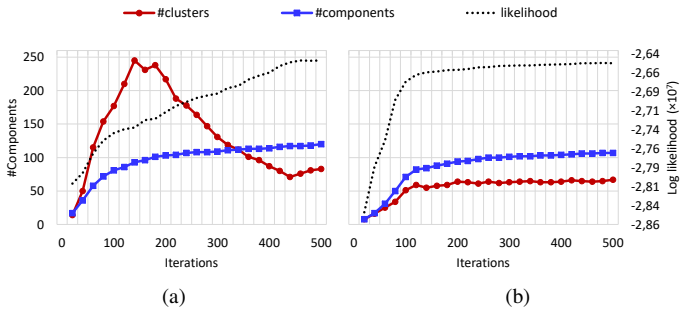


Fig. 10. Sampling behavior on Xitsonga data when *(a)* not using switch moves for cluster components, and *(b)* using switch moves. Without switch moves, the number of clusters might grow fast, and convergence is slow. With switch moves, this issue does not occur. The same behavior was also observed on the English data set.

to the joint sampling we can directly compare the modeling quality of inferred Gaussian mixtures and the single Gaussians, or in other words the DPMoMM and the codebook, i.e., a DPGMM/IGMM.

After sampling the DPMoMM, the data can be represented by frame-wise labels. We use the symmetric NMI to compare the quality of label sequences. For that, we compute the symmetric NMI once using the labels for clusters and once using the labels for codebook components and calculate the relative improvement. In the same way, we also compare the ABX phone discriminability using the frame-wise labels as representation and calculate the relative improvement from using cluster labels instead of codebook component labels.

The ABX phone discriminability can also be computed for posteriorgrams as representation for the data. In that case, either a posteriorgram over clusters or over codebook components is computed for each speech frame. The two kinds of posteriorgrams are scored and compared to get a value for the relative improvement by using *cluster posteriorgrams* instead of *component posteriorgrams*.

We run every sampling for 1000 iterations. Each sampling step is parallelized across 30 threads. For all conducted experiments we sample each model 5 times, score each output and average the results. It is known that the influence of α diminishes in very high data regimes [21]. Chen et al. [10] conducted an experience study and confirmed that the value of α does not impact the outcome of sampling a DPMM given high dimensional speech feature vectors. Their samples are extracted from the same data sets that we use in our work and are similar in nature. In several informal experiments in connection with earlier work we also observed this behavior and therefore set $\alpha = 1$ for all our experiments.

C. The Impact of β

We compare the use of clusters versus using the codebook components as model for the underlying data. The latter corresponds to output that the original sampler of Chang et al. [21] produces. We test on both data sets, English and Xitsonga, and use either PLP+LDA features or PLP+Δ+$\Delta\Delta$ as input. Fig. 7 and 8 show the relative improvements that our proposed method achieved as contour plots.

We observed that using a very small value for the mixture concentration parameter β tends to result in few sampled mixtures that each contain a maximum amount of components. In other words, the sampler is over-confident in grouping components together based on minimal proximity. In contrast to synthetic data, real data tends to be comprised of overlapping classes. The aggressive grouping is one of the consequences of this fact. With larger β, we observed that fewer components are grouped together to form mixtures, which results in a larger number of clusters that contain fewer components. Fig. 9 exemplarily plots the distribution of cluster sizes for the Xitsonga data with different values for β. Fig. 7e, 7j and 8e, 8j show that the number of clusters approximates the number of codebook components as β increases.

The behavior of the cluster inference dependent on β is best explained by analyzing the sampling of weights during restricted Gibbs sampling and the Hastings ratio for cluster split and merge moves. According to (4)-(5), (7)-(8) and (16)-(17), the distributions of cluster and the cluster component weights are governed by β, whose impact is twofold. Its value determines the probability mass that is reserved for generating a new cluster by the split sampler, and it regulates the weights of the cluster components. A large β will motivate the generation of more clusters and cause cluster component weights to take on more similar values, therefore keeping more cluster components alive for a longer time. This in turn encourages more cluster splits, which is also reflected in the Hastings ratios for cluster moves. With larger β, (58) takes on a larger value, i.e., the probability of accepting a split proposal becomes higher, and (59) takes on a smaller value, i.e., the probability of the reverse merge move becomes smaller. Intuitively, the effect is that only closely related components remain grouped in form of a cluster, and less dense clusters are likely to be split to form new clusters with less components. The resulting DPMoMM tends to be made up by many clusters with mostly low amounts of components, and only few clusters with higher amounts of components, if the data suggests so. A small value for β has the exact opposite effect and a sampled DPMoMM will have few clusters with mostly large amounts of cluster components.

D. Convergence and the Switch Sampler

During our experiments, we found that under certain conditions, the number of clusters can grow rapidly and stay

TABLE I
PERFORMANCE COMPARISON OF STANDARD DPMMS AND PROPOSED DPMOMMS APPLIED TO BOTH DATA SETS. PAIRED T-TESTS ON ALL ABX TASK OUTPUTS YIELD $p \ll 0.0001$.

Features	Sampler	Clusters modeled by	β	$c_{\max}$	K	C	K/C	$\text{NMI}_{\max}$	ABX_{ls}	ABX_{kl}	avg. seg. len.
Results for the Xitsonga data set											
PLP+Δ+ΔΔ	DPMoMM	Single Gaussians	-	-	154	-	1.0	0.275	21.31%	14.03%	2.16 frames
		Gaussian Mixtures	200	8	114	154	**0.74**	**0.302**	**18.44%**	**12%**	**2.76** frames
PLP+LDA	DPMoMM	Single Gaussians	-	-	142	-	1.0	0.283	20.29%	13.15%	2.22 frames
		Gaussian Mixtures	400	8	120	142	**0.848**	**0.31**	**18.74%**	**12.47%**	**2.68** frames
PLP+fMLLR	DPMM [13]	Single Gaussians	-	-	139	-	-	-	-	12.2%	-
	DPMoMM	Single Gaussians	-	-	144	-	1.0	0.293	19.33%	12.44%	2.29 frames
	DPMoMM	Gaussian Mixtures	*400*	*8*	126	144	**0.876**	**0.315**	**18.06%**	**11.93%**	**2.7** frames
Results for the English data set											
PLP+Δ+ΔΔ	DPMoMM	Single Gaussians	-	-	232	-	1.0	0.232	28.85%	18.99%	2.06 frames
		Gaussian Mixtures	100	4	110	232	**0.473**	**0.241**	**25.65%**	**17.61%**	**2.37** frames
PLP+LDA	DPMoMM	Single Gaussians	-	-	152	-	1.0	0.248	25.88%	16.18%	2.33 frames
		Gaussian Mixtures	300	4	98	152	**0.649**	**0.262**	**23.74%**	**15.99%**	**2.73** frames
PLP+fMLLR	DPMM [13]	Single Gaussians	-	-	156	-	-	-	-	15.7%	-
	DPMoMM	Single Gaussians	-	-	157	-	1.0	0.253	25.28%	15.84%	2.31 frames
	DPMoMM	Gaussian Mixtures	*300*	*4*	93	157	**0.591**	**0.266**	**23.28%**	**15.47%**	**2.64** frames

large for a long stretch of iterations. This is the case when the allowed cluster size $c_{\max}$ is large and β is set to facilitate many clusters with a large number of components. Under such circumstances, the duplication of clusters during the cluster update step in the component split sampler (see Section V-A) becomes more likely and more frequent, especially if multiple components in the same cluster are subject to splitting.

We developed the switch move sampler to mitigate this issue and support convergence. Fig. 10 exemplarily shows the sampling behavior of our DPMoMM sampler on Xitsonga data with and without using switch moves for cluster components. As can be seen, the number of clusters might grow fast without switch moves, and convergence is slow. With switch moves, however, this issue does not occur. The same behavior was observable for both of our data sets. Without switch moves, a considerably larger amount of time would be required by the sampler to converge. With switch moves, convergence is fast.

E. Use Case Performance

Fig. 7a, 7f and 8a, 8f show the relative improvements of cluster label sequences over codebook component label sequences on the symmetric NMI metric. The best results are always achieved by allowing larger clusters. The optimal values for β seems to lie within a certain range. This range is the same for our two data sets, English and Xitsonga, but it seems to be input feature dependent. Sampling from PLP+Δ+ΔΔ features benefits from a β value between 100 and 300, where sampling from PLP+LDA features might even benefit from a β larger than 400, which during our experiments was the highest value that we tested. These observations transfer to the evaluation of cluster and component labels with the ABX phone discriminability test, as can be seen in Fig. 7b, 7g and 8b, 8g.

Interestingly, we observed considerable performance improvements when we extracted cluster posteriorgrams and compared them to component posteriorgrams with the ABX discriminability test, especially with PLP+Δ+ΔΔ as input (see Fig. 7c, 7h and 8c, 8h). This is noteworthy because despite the much lower dimensionality of cluster posteriorgrams,

TABLE II
PERFORMANCE COMPARISON OF ACOUSTIC UNIT RECOGNIZERS TRAINED ON DPMM AND DPMOMM LABELS. PAIRED T-TESTS ON ALL ABX TASK OUTPUTS YIELD $p \ll 0.0001$.

Labels of	Units	$\text{NMI}_{\max}$	ABX_{ls}	ABX_{kl}	avg. seg. len.
Results for the Xitsonga data set					
Gaussians	144	0.319	18.06%	12.09%	2.59 frames
Mixtures	126	**0.327**	**16.45%**	**11.17%**	**2.95** frames
Results for the English data set					
Gaussians	157	0.263	24.48%	16.5%	2.77 frames
Mixtures	93	**0.278**	**22.21%**	**15.76%**	**3.12** frames

performance not just equals, but even increases, compared to the component posteriorgrams. This is an indicator that the clusters inferred by our proposed DPMoMM are not just accumulations of related classes, but in fact a better approximation to the real underlying multimodal distributions of more complex classes in the data.

The number of inferred clusters does not necessarily correlate with the average length of sample sequences that use the clusters as classes. Fig. 7d, 7i and 8d, 8i show that it is the maximum number of allowed components per cluster $c_{\max}$ that governs the average unit length, rather than β. The average unit lengths is higher if the clusters are allowed to grow larger. A too large β somewhat slows down this trend. The increased unit lengths for larger clusters is an indicator that the grouping of components into mixtures allows the mixture of mixtures model to capture wider acoustic phenomena, an ability that is highly valued in unsupervised subword modeling.

Overall, our results as depicted in Fig. 7 and 8 reveal that with some tuning of β and $c_{\max}$, relative improvements on all evaluation metrics can be as high as 13.5%. The contour plots also suggest that larger mixtures might result in even larger improvements. It is a strong support for our initial motivation for this work that good results can be achieved with settings that lead to the inference of as little as 27% of the amount of clusters than there are codebook components, which considerably reduces model complexity. At the same time, the found clusters can serve as models with average durations of sound instances that are up to 31% longer. Table I

compares the performance of standard DPMMs and good DPMoMMs that we sampled in absolute numbers. During our experiments we sampled models that are best on one metric, but sub-optimal on others. The exemplary DPMoMMs in Table I perform reasonably well on all evaluation metrics and give a good impression of what can be expected if the parameters are tuned reasonably. Note that we did not yet exhaust the exploration of the hyper-parameter space. We expect that even better results are possible with larger values for β and especially $c_{\max}$.

F. State-of-the-art Zero Resource Challenge Performance

In [13], we transformed PLP features with LDA, maximum likelihood linear transforms (MLLT) [40], [41] and feature-space maximum likelihood linear regression (fMLLR) [42], [43] (denoted as "PLP+fMLLR") – a method commonly used for speaker adaptation – to improve the input to a DPMM sampler. The extracted posteriorgrams achieved the to-date best results on the zero resource speech challenge 2015 Xitsonga and English data sets.

We repeated these experiments with our novel sampler, using the hyper-parameters that we tuned for the PLP+LDA features. Table I compares the previously published performance of [13] with using a DPMoMM for sampling instead. We could infer more compact models having fewer clusters and at the same time reduce the discriminability errors – the official evaluation method of the challenge – even further, thus establishing a new state-of-the-art with our proposed method.

G. Acoustic Unit Recognition Performance

In [19], we conducted experiments where the output of a DPMM sampler served as labels for training a context-dependent "triphone" acoustic unit recognizer. In the proposed framework, the DPMM defines the number and distributions of units dynamically and without any prior supervision, given only extracted speech observations, i.e., frame-based feature vectors. We trained an HMM acoustic model and n-gram language model using collapsed DPMM component labels. The resulting DPMM-HMM acoustic unit recognizer was evaluated by solving the ABX sound class discriminability task. Our results showed that it is possible to build a DPMM-HMM acoustic unit recognizer that is competitive with supervisedly trained phone recognizers.

For this work, we conducted experiments with a comparable setup, but using the DPMoMM for subword modeling instead. The labels that we used come from the models that were inferred from PLP+fMLLR-transformed features as listed in Table I. We trained language-dependent acoustic unit recognizers given the DPMoMM cluster labels. We then compared the performance to recognizers that were trained on the DPMoMM component labels instead, which correspond to standard DPMM labels. Since we sampled each DPMoMM five times, we also trained each recognizer five times, scored five times and averaged the results. In addition to the ABX test, we also evaluated the NMI scores. Training and testing was done with 12-fold cross-validation. Table II compares the results of the trained acoustic unit recognizers using cluster labels vs. using single component labels as training references. The recognizers trained on cluster labels perform significantly better across data sets and across all evaluation metrics.

IX. Conclusion

We developed a Dirichlet process mixture of mixtures model (DPMoMM) sampler that jointly infers a codebook of global components and a mixture of clusters. The clusters are themselves mixtures that are defined over the codebook components. Split and merge samplers for modifying codebook components were complemented with split and merge samplers for modifying clusters. We introduced an additional switch sampler for cluster components to support and accelerate convergence, which has shown to be effective in experiments on real data. We demonstrated in the use case of unsupervised subword modeling on two separate data sets, that classes represented by a mixture of mixtures model reliably outperform the output of a standard DPMM. For both data sets and on all our evaluation metrics, the models inferred with our proposed DPMoMM sampler consistently achieved significant performance improvements of up to 13.5% relative. At the same time, considerably fewer classes that show longer average durations were sampled, a behavior that is desired for unsupervised subword modeling. Our experiments suggest that the inferred mixture of mixtures approximates the true underlying distributions of our experimental data much better than a standard DPMM. Lastly, parallelization of sampling steps allows the algorithm to work well even on larger data sets in higher data regimes. Although we originally developed the DPMoMM for improving unsupervised subword modeling, we think that our sampler will be useful not only for handling speech data, but for many tasks where classes are defined by multimodal distributions.

Acknowledgment

Part of this work was supported by JSPS KAKENHI Grant Numbers JP17H06101 and JP17K00237.

References

[1] T. S. Ferguson, "A Bayesian analysis of some nonparametric problems," *Annals of Statistics*, pp. 209–230, 1973.
[2] C. E. Antoniak, "Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems," *Annals of Statistics*, pp. 1152–1174, 1974.
[3] C. E. Rasmussen, "The infinite Gaussian mixture model," in *Advances in Neural Information Processing Systems*. Neural Information Processing Systems Foundation, 2000, pp. 554–560.
[4] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Sharing clusters among related groups: Hierarchical Dirichlet processes," in *Advances in Neural Information Processing Systems*. Neural Information Processing Systems Foundation, 2005, pp. 1385–1392.
[5] J. Chang and J. W. Fisher III, "Parallel sampling of HDPs using sub-cluster splits," in *Advances in Neural Information Processing Systems*. Neural Information Processing Systems Foundation, 2014, pp. 235–243.
[6] S. N. MacEachern, "Dependent nonparametric processes," in *Proceedings of the Section on Bayesian Statistical Science*. American Statistical Association, 1999, pp. 50–55.
[7] ——, "Dependent Dirichlet processes," *Unpublished manuscript, Department of Statistics, The Ohio State University*, pp. 1–40, 2000.
[8] M. Versteegh, R. Thiolliere, T. Schatz, X. N. Cao, X. Anguera, A. Jansen, and E. Dupoux, "The zero resource speech challenge 2015," in *Proceedings of Interspeech*. International Speech Communication Association, 2015, pp. 3169–3173.

[9] E. Dunbar, X. N. Cao, J. Benjumea, J. Karadyi, M. Bernard, L. Besacier, X. Anguerra, and E. Dupoux, "The zero resource speech challenge 2017," in *Proceedings of the Automatic Speech Recognition and Understanding Workshop*. Institute of Electrical and Electronics Engineers, 2017, pp. 323–330.

[10] H. Chen, C.-C. Leung, L. Xie, B. Ma, and H. Li, "Parallel inference of Dirichlet process Gaussian mixture models for unsupervised acoustic modeling: A feasibility study," in *Proceedings of Interspeech*. International Speech Communication Association, 2015, pp. 3189–3193.

[11] ——, "Multilingual bottle-neck feature learning from untranscribed speech," in *Proceedings of the Automatic Speech Recognition and Understanding Workshop*. Institute of Electrical and Electronics Engineers, 2017.

[12] M. Heck, S. Sakti, and S. Nakamura, "Unsupervised linear discriminant analysis for supporting DPGMM clustering in the zero resource scenario," in *Proceedings of the International Workshop on Spoken Language Technologies for Under-resourced Languages*, 2016, pp. 73–79.

[13] ——, "Supervised learning of acoustic models in a zero resource setting to improve DPGMM clustering." in *Proceedings of Interspeech*. International Speech Communication Association, 2016, pp. 1310–1314.

[14] ——, "Feature optimized DPGMM clustering for unsupervised subword modeling: A contribution to Zerospeech 2017," in *Proceedings of the Automatic Speech Recognition and Understanding Workshop*. Institute of Electrical and Electronics Engineers, 2017, pp. 740–746.

[15] L. R. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Prentice Hall, 1993.

[16] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-dependent modeling for acoustic-phonetic recognition of continuous speech," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, vol. 10. Institute of Electrical and Electronics Engineers, 1985, pp. 1205–1208.

[17] K.-F. Lee, "On large-vocabulary speaker-independent continuous speech recognition," *Speech Communication*, vol. 7, no. 4, pp. 375–379, 1988.

[18] A. Waibel and K.-F. Lee, Eds., *Readings in speech recognition*. Morgan Kaufmann, 1990.

[19] M. Heck, S. Sakti, and S. Nakamura, "Iterative training of a DPGMM-HMM acoustic unit recognizer in a zero resource scenario," in *Proceedings of the Spoken Language Technology Workshop*. Institute of Electrical and Electronics Engineers, 2016, pp. 57–63.

[20] H. Z. Yerebakan, B. Rajwa, and M. Dundar, "The infinite mixture of infinite Gaussian mixtures," in *Advances in Neural Information Processing Systems*. Neural Information Processing Systems Foundation, 2014, pp. 28–36.

[21] J. Chang and J. W. Fisher III, "Parallel sampling of DP mixture models using sub-cluster splits," in *Advances in Neural Information Processing Systems*. Neural Information Processing Systems Foundation, 2013, pp. 620–628.

[22] X. Zhu, Z. Ghahramani, and J. Lafferty, "Time-sensitive Dirichlet process mixture models," School of Computer Science, Carnegie Mellon University, Tech. Rep., 2005.

[23] J. E. Griffin and M. J. Steel, "Order-based dependent Dirichlet processes," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 179–194, 2006.

[24] D. M. Blei and P. I. Frazier, "Distance dependent Chinese restaurant processes," *Journal of Machine Learning Research*, vol. 12, pp. 2461–2488, 2011.

[25] D. B. Dahl, "An improved merge-split sampler for conjugate Dirichlet process mixture models," Department of Statistics, University of Wisconsin Madison, Tech. Rep., 2003.

[26] S. Jain and R. M. Neal, "A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model," *Journal of Computational and Graphical Statistics*, vol. 13, no. 1, pp. 158–182, 2004.

[27] ——, "Splitting and merging components of a nonconjugate Dirichlet process mixture model," *Bayesian Analysis*, vol. 2, no. 3, pp. 445–472, 2007.

[28] O. Papaspiliopoulos and G. O. Roberts, "Retrospective Markov chain Monte Carlo methods for Dirichlet process hierarchical models," *Biometrika*, vol. 95, no. 1, pp. 169–186, 2008.

[29] S. Favaro and Y. W. Teh, "MCMC for normalized random measure mixture models," *Statistical Science*, pp. 335–359, 2013.

[30] H. Ishwaran and L. F. James, "Gibbs sampling methods for stick-breaking priors," *Journal of the American Statistical Association*, vol. 96, no. 453, pp. 161–173, 2001.

[31] J. Yang, R. C. Van Dalen, and M. J. Gales, "Infinite support vector machines in speech recognition," in *Proceedings of Interspeech*. International Speech Communication Association, 2013.

[32] J. Yang, R. C. Van Dalen, S.-X. Zhang, and M. J. Gales, "Infinite structured support vector machines for speech recognition," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. Institute of Electrical and Electronics Engineers, 2014, pp. 3320–3324.

[33] T. Schatz, V. Peddinti, F. Bach, A. Jansen, H. Hermansky, and E. Dupoux, "Evaluating speech features with the minimal-pair ABX task: Analysis of the classical MFC/PLP pipeline," in *Proceedings of Interspeech*. International Speech Communication Association, 2013, pp. 1781–1785.

[34] A. F. McDaid, D. Greene, and N. Hurley, "Normalized mutual information to evaluate overlapping community finding algorithms," *arXiv preprint arXiv:1110.2515*, 2011.

[35] M. A. Pitt, K. Johnson, E. Hume, S. Kiesling, and W. Raymond, "The Buckeye corpus of conversational speech: Labeling conventions and a test of transcriber reliability," *Speech Communication*, vol. 45, no. 1, pp. 89–95, 2005.

[36] N. J. De Vries, M. H. Davel, J. Badenhorst, W. D. Basson, F. De Wet, E. Barnard, and A. De Waal, "A smartphone-based ASR data collection tool for under-resourced languages," *Speech Communication*, vol. 56, pp. 119–131, 2014.

[37] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," in *Proceedings of the Automatic Speech Recognition and Understanding Workshop*. Institute of Electrical and Electronics Engineers, 2011.

[38] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *The Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[39] R. Haeb-Umbach and H. Ney, "Linear discriminant analysis for improved large vocabulary continuous speech recognition," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, vol. 1. Institute of Electrical and Electronics Engineers, 1992, pp. 13–16.

[40] R. A. Gopinath, "Maximum likelihood modeling with Gaussian distributions for classification," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. Institute of Electrical and Electronics Engineers, 1998, pp. 661–664.

[41] M. J. Gales, "Semi-tied covariance matrices for hidden Markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 272–281, 1999.

[42] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, "A compact model for speaker-adaptive training," in *Proceedings of the International Conference on Spoken Language*. Institute of Electrical and Electronics Engineers, 1996, pp. 1137–1140.

[43] M. J. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech and Language*, vol. 12, no. 2, pp. 75–98, 1998.

Michael Heck received his diploma degree in computer science from Karlsruhe Institute of Technology (KIT), Germany, in 2012. He was granted the "Baden-Württemberg" scholarship and was supported by the Global Initiatives Program for a stay at Nara Institute of Science and Technology (NAIST), Japan, during his thesis work. Between 2013-2015, he worked as research assistant at the Interactive Systems Labs at KIT on unsupervised training and adaptation of acoustic models for automatic speech recognition. He was involved in several speech recognition evaluations and international programs including Quaero, Babel and EU-BRIDGE. While at KIT, he was a regular visiting researcher at the Augmented Human Communication Laboratory (AHC Lab) at NAIST. Since 2015, he is a research assistant and doctoral course student at AHC Lab as recipient of the NAIST international scholarship. In 2017, he was a research intern at IBM Research – Tokyo. His research interests include automatic speech recognition, unsupervised learning and the zero resource scenario.

Sakriani Sakti received her B.E. degree in Informatics (cum laude) from Bandung Institute of Technology, Indonesia, in 1999. In 2000, she received DAAD-Siemens Program Asia 21st Century Award to study in Communication Technology, University of Ulm, Germany, and received her MSc degree in 2002. During her thesis work, she worked with Speech Understanding Department, DaimlerChrysler Research Center, Ulm, Germany. Between 2003-2009, she worked as a researcher at ATR SLC Labs, Japan, and during 2006-2011, she worked as an expert researcher at NICT SLC Groups, Japan. While working with ATR-NICT, Japan, she continued her study (2005-2008) with Dialog Systems Group University of Ulm, Germany, and received her PhD degree in 2008. She was actively involved in collaboration activities such as Asian Pacific Telecommunity Project (2003-2007), A-STAR and U-STAR (2006-2011). In 2009-2011, she served as a visiting professor of Computer Science Department, University of Indonesia (UI), Indonesia. From 2011, she has been an assistant professor at the Augmented Human Communication Laboratory, NAIST, Japan. She served also as a visiting scientific researcher of INRIA Paris-Rocquencourt, France, in 2015-2016, under "JSPS Strategic Young Researcher Overseas Visits Program for Accelerating Brain Circulation". She is a member of JNS, SFN, ASJ, ISCA, IEICE and IEEE. Her research interests include statistical pattern recognition, speech recognition, spoken language translation, cognitive communication, and graphical modeling framework.

Satoshi Nakamura is Professor at the Graduate School of Information Science, Nara Institute of Science and Technology, Japan, Honorarprofessor of Karlsruhe Institute of Technology, Germany, and ATR Fellow. He received his B.S. from Kyoto Institute of Technology in 1981 and Ph.D. from Kyoto University in 1992. He was Associate Professor of Graduate School of Information Science at Nara Institute of Science and Technology in 1994-2000. He was Director of ATR Spoken Language Communication Research Laboratories in 2000-2008 and Vice president of ATR in 2007-2008. He was Director General of Keihanna Research Laboratories and the Executive Director of Knowledge Creating Communication Research Center, National Institute of Information and Communications Technology, Japan in 2009-2010. He is currently Director of Augmented Human Communication laboratory and a full professor of Graduate School of Information Science at Nara Institute of Science and Technology. He is interested in modeling and systems of speech-to-speech translation and speech recognition. He is one of the leaders of speech-to-speech translation research and has been serving for various speech-to-speech translation research projects in the world including C-STAR, IWSLT and A-STAR. He received Yamashita Research Award, Kiyasu Award from the Information Processing Society of Japan, Telecom System Award, AAMT Nagao Award, Docomo Mobile Science Award in 2007, ASJ Award for Distinguished Achievements in Acoustics. He received the Commendation for Science and Technology by the Minister of Education, Science and Technology, and the Commendation for Science and Technology by the Minister of Internal Affairs and Communications. He also received LREC Antonio Zampoli Award 2012. He has been Elected Board Member of International Speech Communication Association, ISCA, since June 2011, IEEE Signal Processing Magazine Editorial Board Member since April 2012, IEEE SPS Speech and Language Technical Committee Member since 2013, and IEEE Fellow since 2016.