GLU 2017 International Workshop on Grounding Language Understanding
25 August 2017, Stockholm, Sweden

Partitioning of Posteriorgrams using Siamese Models for Unsupervised Acoustic Modelling

Arvid Fahlström Myrman and Giampiero Salvi

KTH Royal Institute of Technology,
School of Computer Science and Communication,
Dept. of Speech, Music and Hearning, Stockholm, Sweden

{arvidfm, giampi}@kth.se

Abstract

Unsupervised methods tend to discover highly speaker-specific representations of speech. We propose a method for improving the quality of posteriorgrams generated from an unsupervised model through partitioning of the latent classes. We do this by training a sparse siamese model to find a linear transformation of the input posteriorgrams to lower-dimensional posteriorgrams. The siamese model makes use of same-category and different-category speech fragment pairs obtained by unsupervised term discovery. After training, the model is converted into an exact partitioning of the posteriorgrams. We evaluate the model on the minimal-pair ABX task in the context of the Zero Resource Speech Challenge. We are able to demonstrate that our method significantly reduces the dimensionality of standard Gaussian mixture model posteriorgrams, while still making them more robust to speaker variations. This suggests that the model may be viable as a general post-processing step to improve probabilistic acoustic features obtained by unsupervised learning.

Index Terms: zero-resource speech challenge, unsupervised learning, speech recognition

1. Introduction

In the area of unsupervised learning in speech technology, researchers have, for many decades, attempted to reduce the amount of handcrafted information needed to build speech recognition and synthesis systems. Many aspects of language acquisition have been covered, from finding sub-word units serving a similar role as pre-defined phonemes [1]–[5] to the discovery of recurrent patterns that may constitute word candidates [6]–[12]. Recently, Versteegh *et al.* introduced the Zero Resource Speech Challenge [13], with the goal of standardising these endeavours. Specifically, the first track, most relevant to this paper, involves finding speaker independent linguistic units from speech with no or weak supervision. The discovery of linguistic units follows two main approaches in the literature that we will refer to as frame-based and term discovery-based. In the first case, acoustic models are inferred directly from the acoustic features [14]–[21]. The second approach is to first segment the speech into syllable- or word-like units, and afterwards break these units into smaller subword units [7], [13], [19], [22]–[29].

Frame-based approaches: Varadarajan *et al.* [14] first define a one-state HMM, and then iteratively split and merge states depending on the data and according to a heuristic. The states of the final models (allophones), are then mapped into phonemes with the help of a separate model trained using labelled speech, making the method not fully unsupervised. Lee and Glass [15] use an infinite mixture model of three-state HMM-GMMs that performs segmentation and acoustic modelling jointly. Inference of the model is done using Gibbs sampling. A similar model but, without constraints on the topology of the HMMs was studied in [12]. Siu *et al.* [16] first use a segmental GMM (SGMM) to generate a transcription of the data and then iteratively train a standard HMM to improve the transcriptions. Note that the number of allowed states are here defined in advance.

Diverging from previous approaches which use temporal models, Chen *et al.* [17] perform standard clustering of speech frames using an infinite Gaussian mixture model. After training, the speech frames are represented as posteriorgrams, which have been shown to be more speaker-invariant than other features such as MFCCs [18]. Despite the simple approach, this turned out to be the overall best-performing model in the first track of the 2015 Zero Resource Speech Challenge [19]. Heck *et al.* [20] further improved on the model by performing clustering in two stages, with an intermediate supervised dimensionality reduction step using the clusters derived from the first clustering step as target classes. In [21] a siamese network [30] is used to create an embedding where speech frames close to each other are considered to belong to the same subword unit, while distant speech frames are said to differ.

Term discovery-based approaches use unsupervised term discovery (UTD) to extract word-like segments that can guide the discovery of more stable sub-word units compared to purely frame-based approaches. The UTD is usually based on the segmental dynamic time warping (S-DTW) developed in [7]. In [23] an approximate version is introduced that reduces the complexity from $O(n^2)$ to $O(n \log n)$ time. This system also serves as the baseline for the second track of the Zero Resource Speech Challenge. The information from UTD is used in [24] to train term-specific HMMs. The states from each HMM are then clustered based on the similarity of their distributions, to form subword unit candidates. A related approach is taken in [22], where instead of HMM states, components from a GMM trained on speech frames are clustered based on co-occurence in pairs of fragments obtained from UTD. A neural network referred to as the ABnet and based on siamese networks [30] is introduced in [25]. The network takes a pair of speech frames as input, and adjusts its parameters so that the outputs are collinear if the inputs are known to correspond to the same subword unit, and orthogonal otherwise, using a cosine-based loss function. Thiolliere *et al.* [26] make use of this approach in the Zero Resource Speech Challenge, also incorporating UTD so as to make the whole process unsupervised, yielding competitive results [19]. Zeghidour *et al.* [27] experiment with supplying the ABnet with scattering spectrum features instead of filter bank features, showing that with the right features, a shallow architecture may outperform a deep architecture, especially when the amount of available data is low. Kamper *et al.* [28] use an autoencoder-like structure,

10.21437/GLU.2017-6

where a neural network is trained to "reconstruct" a frame given another frame known to be of the same type. Renshaw *et al.* [29] used this architecture in the Zero Resource Speech Challenge, albeit with a deeper decoder.

A limitation of term discovery-based approaches is that the UTD methods discussed here only discover a fraction of recurring patterns in the data, limiting the amount of available training data.

This work takes inspiration from the two so far most successful approaches, namely the clustering approach of [17] and the siamese network approach of [26]. We first cluster the data in an unsupervised manner using a GMM. We then improve the resulting posteriorgrams using information from UTD by mapping speaker- or context-specific classes to broader classes with a linear siamese model. This way we are able to take advantage of both the whole unlabelled data set, and the smaller set of fragments discovered by UTD. While the approach of partitioning posteriorgrams is reminiscent of [22], the major difference is that in place of direct clustering of classes, we are instead trying to maximise the similarity/dissimilarity between pairs of speech fragments, which only indirectly results in a partition of the classes. Our linear model also has the advantage of being more interpretable than deep networks like that of [26].

2. Method

The goal of our method is to find low-dimensional representations of posteriorgrams obtained by probabilistic unsupervised clustering such that the resulting representation is more invariant to speaker variation. The dimensionality reduction is done by partitioning the latent classes corresponding to the posteriorgram components. In order to do this in an unsupervised manner, we use information obtained by unsupervised term discovery (UTD) as in [23]. The result of the UTD is a set of clusters of speech fragments thought to be of the same category (e.g. word). We then generate a set of same-class and different-class frame pairs, each frame represented as a posteriorgram, by sampling and aligning fragment pairs as in [26].

We represent the input data as a set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ of N pairs of M-dimensional posteriorgrams, along with a set of indicators $\{c_i\}_{i=1}^N$ such that c_i is 1 if $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ belong to the same category, and 0 otherwise. We wish to transform the input to D-dimensional posteriorgrams such that two inputs $\boldsymbol{x}_i$, $\boldsymbol{y}_i$ are close in output space if $c_i = 1$, and distant otherwise. Our model is a simple linear transformation

$$f(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}, \quad \boldsymbol{W} \in \mathbb{R}^{M\times D}. \tag{1}$$

In order to ensure that the output is a probability distribution, we need to constrain $\boldsymbol{W}$ so that each element is positive, and the elements of each row sum to 1. This is done by costructing the model as follows:

$$\boldsymbol{V} \in \mathbb{R}^{M\times D} \tag{2}$$
$$\widetilde{\boldsymbol{W}} = |\boldsymbol{V}| \tag{3}$$
$$\boldsymbol{W} = \widetilde{\boldsymbol{W}} \oslash \left(\widetilde{\boldsymbol{W}}\mathbf{1}_D\mathbf{1}_D^T\right) \tag{4}$$
$$f(\boldsymbol{x}; \boldsymbol{V}) = \boldsymbol{x}\boldsymbol{W} \tag{5}$$

where $\mathbf{1}_D$ is a column vector of D ones and $\mathbf{1}_D^T$ its transpose, $|\cdot|$ denotes the element-wise absolute value, and $\oslash$ denotes element-wise division. Note that the function of equation (4) is to normalise the rows of $\boldsymbol{W}$ to sum to one. This formulation makes it possible to optimise the model while ensuring that the constraints on $\boldsymbol{W}$ hold, by performing gradient descent with respect to $\boldsymbol{V}$.

To encourage the model to place points belonging to the same class close together in the output space, we use the siamese paradigm of [25], [26]. Let $B_1 = \{i \in B : c_i = 1\}$ be the subset of same-class pairs in the current minibatch, and $B_0 = \{i \in B : c_i = 0\}$ the subset of different-class pairs. Additionally, let $\hat{\boldsymbol{x}}_i = f(\boldsymbol{x}_i; \boldsymbol{V})$ and $\hat{\boldsymbol{y}}_i = f(\boldsymbol{y}_i; \boldsymbol{V})$. We then define the loss function over a minibatch B as

$$\begin{aligned} L_{\text{JS}}(\boldsymbol{V}; B) = &\frac{1}{(\alpha+1)|B_1|}\sum_{i\in B_1}\overbrace{\sqrt{\text{JS}(\hat{\boldsymbol{x}}_i||\hat{\boldsymbol{y}}_i)}}^{\text{same-class}} \\ &+ \frac{\alpha}{(\alpha+1)|B_0|}\sum_{i\in B_0}\underbrace{\left(1-\sqrt{\text{JS}(\hat{\boldsymbol{x}}_i||\hat{\boldsymbol{y}}_i)}\right)}_{\text{different-class}}, \end{aligned} \tag{6}$$

where α is a hyperparameter determining how much to weight the different-class loss over the same-class loss, and $\text{JS}(\boldsymbol{x}||\boldsymbol{y})$ is the Jensen-Shannon (JS) divergence defined as

$$\text{JS}(\boldsymbol{x}||\boldsymbol{y}) = \frac{1}{2}\text{KL}(\boldsymbol{x}||\boldsymbol{m}) + \frac{1}{2}\text{KL}(\boldsymbol{y}||\boldsymbol{m}), \tag{7}$$

where $\text{KL}(\boldsymbol{x}||\boldsymbol{y})$ is the Kullback-Leibler (KL) divergence, and $\boldsymbol{m} = (\boldsymbol{x}+\boldsymbol{y})/2$. Thus, we attempt to minimise the JS divergence between same-class outputs, while maximising the divergence between different-class outputs.

The choice of a statistical distance as the loss function is motivated by the fact that the output of the model is a probability distribution. The JS divergence was chosen over the KL divergence because it is symmetric, always defined, and bounded between 0 and 1 (when using the base-2 logarithm), making it more appropriate for maximising the divergence between different-class outputs. Additionally, the square root of the JS divergence, used here, is a metric satisfying the triangle inequality [31].

2.1. Entropy penalty

To ensure the interpretability of the output, we add a penalty term that attempts to minimise the entropy, i.e. the spread of the probability mass, in the output distribution. We use the normalised entropy, defined as

$$\hat{H}(\boldsymbol{x}) = -\frac{1}{\log_2 D}\sum_{i=1}^{D} x_i \log_2 x_i. \tag{8}$$

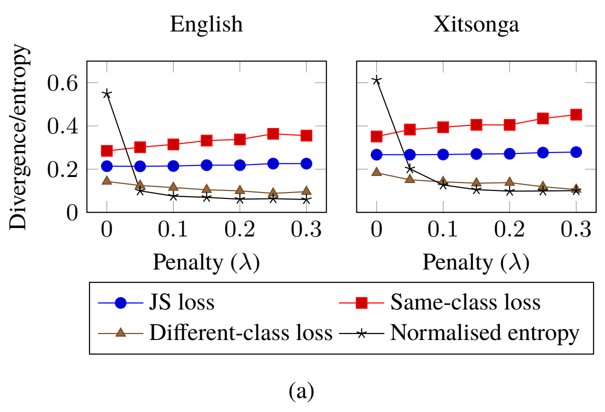
The normalisation ensures that the entropy is always bounded between 0 and 1, regardless of the number of outputs D of the model. Over a minibatch B, the entropy loss function is given as

$$\begin{aligned} L_{\text{H}}(\boldsymbol{V}; B) = & \\ &\frac{1}{2|B|}\sum_{i\in B}\left(\hat{H}(f(\boldsymbol{x}_i;\boldsymbol{V})) + \hat{H}(f(\boldsymbol{y}_i;\boldsymbol{V}))\right) \end{aligned} \tag{9}$$

The entropy penalty implicitly encourages sparsity in $\boldsymbol{W}$, as the only way to avoid spreading the probability mass across several outputs is for each row of $\boldsymbol{W}$ to only contain a single element close to 1. In summary, the complete loss over a minibatch B is as follows:

$$L(\boldsymbol{V}; B) = L_{\text{JS}}(\boldsymbol{V}; B) + \lambda L_{\text{H}}(\boldsymbol{V}; B) \tag{10}$$

where λ is a hyperparameter.



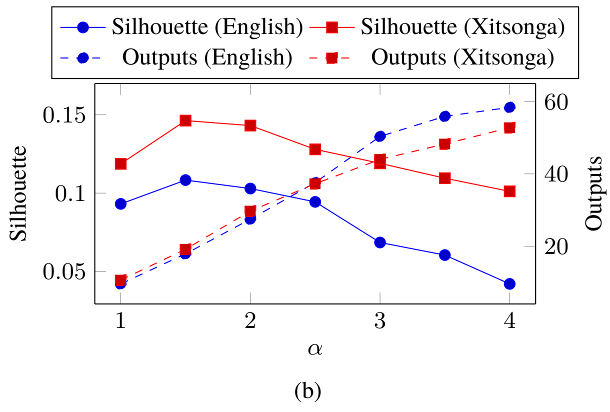


Figure 1: ***Left:*** *Normalised entropy of the output distribution and the combined, same-class and different-class Jensen-Shannon losses, averaged over the validation set, as a function of the entropy penalty hyperparameter λ for the English and Xitsonga data sets.* ***Right:*** *Silhouette for different weightings α of the same-class and different-class losses. Also shown is a heuristically calculated estimate of the number of outputs used by the model for different α.*

2.2. Binarising the model

As the resulting model is sparse, we can construct an exact partition of the input classes. We do this by setting the largest element in each row in $\boldsymbol{W}$ to 1, and the remaining elements to 0, resulting in a binary $\boldsymbol{W}$. An optional further processing step is to binarise the output distribution by setting the largest output to 1 and the rest to 0; this can be thought of as taking the argmax of the output distribution.

3. Experiments

3.1. Data

To test our method we use the same data as the 2015 Zero Resource Speech Challenge. The challenge makes use of two corpora: The Buckeye corpus of conversational English [32] and the NCHLT speech corpus of read Xitsonga [33]. For the challenge only a subset of the data is used, consisting of 12 speakers for a total of 5 hours of data for the Buckeye corpus, and 24 speakers for a total of 2.5 hours of data for the NCHLT Xitsonga corpus. Additionally provided is voice activity information indicating segments containing clean speech, as well as labels indicating the identity of the speaker.

MFCC features were extracted from the data using a frame window length of 25 ms which was shifted 10 ms for each frame, an FFT resolution of 512 frequency steps, and 40 mel-spaced triangular filter banks. 13 coefficients with both delta and delta-delta features were used. The MFCCs corresponding to segments with voice activity were clustered using an implementation of a Gaussian mixture model (GMM) provided by scikit-learn [34]. The GMM was trained using the expectation maximisation algorithm, using $M = 1024$ Gaussians with diagonal covariance matrices, for a maximum of 200 iterations. After training, posteriorgrams were calculated for each frame.

The unsupervised term discovery yielded 6512 fragments and 3149 clusters for the Buckeye corpus, and 3582 fragments and 1782 clusters for the NCHLT Xitsonga corpus[1]. 70% of the same-class and different-class fragment pairs were used for training, with the remaining pairs used for validation to determine when to interrupt the training of the models.

[1]The cluster files used in this work were generously provided by Roland Thiollière [26] and were generated by Aren Jansen [23].

3.2. Model implementation

We used $D = 64$ outputs for all models. The models were trained using AdaMax [35] with the recommended default parameters. All frames used for training were shuffled once at the start of training, and a minibatch size of 1000 frame pairs was used. The models were trained until no improvement had been observed on the held-out validation set for 15 epochs, where one epoch is defined as one complete scan over the training data.

All network models were implemented in Python 3.5 using Theano [36] for automatic differentiation and GPU acceleration, librosa [37] for feature extraction, scikit-learn [34] for various utilities, and numba [38] for accelerating the dynamic time warping code. The training was performed on a GeForce GTX TITAN with 6 GB VRAM and 12 Intel i7-5930K cores clocked at 3.50 GHz, with 64 GB RAM.

3.3. Tuning the hyperparameters

Figure 1a shows the Jensen-Shannon and entropy losses after convergence as a function of λ, with α fixed at 1. We choose $\lambda = 0.1$, as little improvement of the entropy is seen for larger values. The fact that such a low value suffices to minimise the entropy suggests that the entropy is easy to optimise for.

To find an optimal α we make use of the clusters discovered by the UTD system, choosing the α that maximises the separation of the clusters. We use the silhouette [39] as a cluster separation measure, taking the distance between individual fragments to be the DTW score, with the symmetrised KL divergence as the frame-based distance, where each frame is represented as the output of the model being evaluated. The silhouette for different α with λ fixed at 0.1, calculated on a subset of 1000 clusters, can be seen in figure 1b. The optimal value is found to be $\alpha = 1.5$ for both data sets.

3.4. Model evaluation

We train two models, one with $\alpha = 1$ and the other with $\alpha = 1.5$ to measure the influence of reweighting the losses. Both models are trained with an entropy penalty hyperparameter of $\lambda = 0.1$. We additionally construct an exact partition using the latter model, by binarising the weight matrix $\boldsymbol{W}$. Finally, we also evaluate how much performance is retained when further binarising the output of the model with binary $\boldsymbol{W}$.

Table 1: *ABX and silhouette results for the models described in section 3.4.*

		English			Xitsonga		
	Model	Silhouette	Within	Across	Silhouette	Within	Across
Baseline	GMM posteriors	0.008	12.3	23.8	0.066	11.4	23.2
Proposed models	Real $\boldsymbol{W}$ ($\alpha = 1$)	0.093	14.1	21.2	0.119	15.8	25.1
	Real $\boldsymbol{W}$ ($\alpha = 1.5$)	0.108	12.8	19.8	0.146	14.0	23.2
	Binary $\boldsymbol{W}$	0.124	12.0	19.3	0.170	12.7	21.9
	Binary output	0.010	16.5	24.6	0.014	19.4	29.2
Related models	Deep JS	-0.058	18.1	25.6	0.150	17.5	23.5
	Deep coscos^2 (own implementation)	0.175	12.0	19.7	0.298	11.8	19.2
	Deep coscos^2 [26]	—	12.0	17.9	—	11.7	16.6
	DPGMM + LDA [20]	—	10.6	16.0	—	8.0	12.6

To compare to the shallow models, and to get an idea of how the JS loss performs in general, we also build a deep network with two hidden layers of 500 sigmoid units each, with 64 softmax outputs. The network is trained using the JS loss with $\alpha = 1$. As softmax outputs are naturally sparse, we do not enforce any entropy penalty. For comparison we train the same architecture, albeit with 100 sigmoid outputs instead, using the coscos^2 loss of Synnaeve *et al.* [25]. This is the architecture used by Thiolliere *et al.* [26]. In place of posteriorgrams we use log-scale outputs of 40 mel-scaled filter banks, normalised to have zero mean and unit variance over the whole data set and with a context of 3 frames on both sides, for a total of 280 values as input to the deep networks.

We evaluate the models on the minimal-pair ABX task [40] using the toolkit provided for the Zero Resource Speech Challenge [13]. For the models with continuous output, the frame-based metric is chosen as the symmetrised Kullback-Leibler divergence (with the model output normalised as necessary). For the model with binary output, however, we use a distance of 0 for identical and 1 for non-identical vectors.

4. Results

The results of the ABX evaluation are shown in table 1, along with the silhouette for each model. The silhouette is calculated as in section 3.3; higher is better. The ABX scores are shown as the percentage of ABX triples for which the model answered incorrectly; lower is better. We show results for both the within-speaker and across-speaker ABX tasks.

We can see that in general, the silhouette seems to be indicative of the relative performance of the models on the ABX task, with well-performing models having a higher silhouette score. Among the shallow models, we see that rebalancing the same-class and different-class losses results in significant gains, with binarisation of the weights further improving the results. Unsuprisingly however, binarising the output as well severely worsens the results, likely due to too much information being discarded. We find that while the models perform worse than the current state-of-the-art [20], especially for Xitsonga, they were generally able to improve on the input posteriorgrams, especially for the across-speaker task.

The resulting shallow models are very sparse, with the average row-wise maximum weight of $\boldsymbol{W}$ being 0.991 for English and 0.929 for Xitsonga, for $\alpha = 1.5$. This also results in only a subset of the available outputs being used, with 33 outputs receiving any probability mass when binarising the English model; for Xitsonga 35 outputs were used.

The deep model performs poorly when trained with the Jensen-Shannon loss, despite a similar architecture performing well when trained with the coscos^2 loss. Inspecting the average output of the deep model over the English data set, we found that only 6 outputs are actually used by the model. This suggests that the JS loss is more sensitive than the coscos^2 loss when it comes to balancing the same-class and different-class losses. Note that we were unable to replicate the results of Thiolliere *et al.* [26] using the coscos^2 loss.

5. Conclusions

A linear model for partitioning of posteriorgrams was introduced and applied to unsupervised learning of linguistic units in speech. The model was shown to be able to reduce the dimensionality of GMM posteriorgrams from 1024 to below 40, while simultaneously improving the across-speaker robustness. The model does not depend on the GMM, however, as it is able to take posteriorgrams generated from any probabilistic model as input, the only requirement being that the underlying true classes are disentangled in the input representation.

While the model depends on two hyperparameters, the hyperparameter search is alleviated somewhat by the ease of training the linear model. Additionally, the entropy penalty was shown to be easy to optimise for. The silhouette was shown to be indicative of ABX performance, enabling hyperparameter search without making use of the gold transcription.

The resulting model is sparse, easily interpretable, and robust to overfitting as a result of the low number of parameters and the regularisation imposed by the entropy penalty. This entropy penalty also results in only a subset of the outputs being used, making the model insensitive to the total number of available outputs. However, the Jensen-Shannon loss function used is sensitive to the balancing of the same-class and different-class losses, making it particularly unsuitable for deep architectures.

We believe that the disparity in performance between English and Xitsonga may be due to the lower number of speech fragment pairs obtained through unsupervised term discovery for Xitsonga. Further investigation is needed to assess why our model is more adversely affected by this than deeper models.

6. Acknowledgements

The GeForce GTX TITAN used for this research were donated by the NVIDIA Corporation. This research was partly supported by the CHIST-ERA project IGLU.

7. References

[1] C.-H. Lee, F. Soong, and B.-H. Juang, "A segment model based approach to speech recognition," in *Proc. of IEEE ICASSP*, 1988.

[2] T. Svendsen, K. Paliwal, E. Harborg, and P. Husoy, "An improved sub-word based speech recognizer," in *Proc. of IEEE ICASSP*, 1989.

[3] M. Bacchiani, M. Ostendorf, Y. Sagisaka, and K. Paliwal, "Design of a speech recognition system based on acoustically derived segmental units," in *Proc. of IEEE ICASSP*, 1996.

[4] M. Huijbregts, M. McLaren, and D. Van Leeuwen, "Unsupervised acoustic sub-word unit detection for query-by-example spoken term detection," in *Proc. of IEEE ICASSP*, 2011.

[5] P. O'Grady, "Discovering speech phones using convolutive non-negative matrix factorisation with a sparseness constraint," *Neurocomputing*, vol. 72, no. 1-3, pp. 88–101, 2008.

[6] O. Räsänen, "A computational model of word segmentation from continuous speech using transitional probabilities of atomic acoustic events," *Cognition*, vol. 120, no. 2, pp. 149–176, 2011.

[7] A. S. Park and J. R. Glass, "Unsupervised pattern discovery in speech," *IEEE Trans. on Audio, Speech, and Lang. Proc.*, vol. 16, no. 1, pp. 186–197, 2008.

[8] G. Aimetti, R. K. Moore, and L. ten Bosch, "Discovering an optimal set of minimally contrasting acoustic speech units: A point of focus for whole-word pattern matching," in *Proc. of Interspeech*, 2010.

[9] V. Stouten, K. Demuynck, and H. van Hamme, "Discovering phone patterns in spoken utterances by non-negative matrix factorization," *IEEE Signal Processing Lett.*, vol. 15, pp. 131–134, 2008.

[10] J. Driesen, L. ten Bosch, and H. van Hamme, "Adaptive non-negative matrix factorization in a computational model of language acquisition," in *Proc. of Interspeech*, 2009.

[11] N. Vanhainen and G. Salvi, "Word discovery with beta process factor analysis," in *Proc. of Interspeech*, 2012.

[12] ——, "Pattern discovery in continuous speech using block diagonal infinite hmm," in *Proc. of IEEE ICASSP*, 2014.

[13] M. Versteegh, R. Thiolliere, T. Schatz, *et al.*, "The Zero Resource Speech Challenge 2015," in *Proc. of Interspeech*, 2015.

[14] B. Varadarajan, S. Khudanpur, and E. Dupoux, "Unsupervised learning of acoustic sub-word units," in *Proc. of ACL, Short Papers*, 2008.

[15] C.-y. Lee and J. Glass, "A nonparametric Bayesian approach to acoustic model discovery," in *Proc. of ACL, Long Papers*, 2012.

[16] M.-h. Siu, H. Gish, A. Chan, *et al.*, "Unsupervised training of an HMM-based self-organizing unit recognizer with applications to topic classification and keyword discovery," *Computer Speech and Language*, vol. 28, no. 1, pp. 210–223, 2014.

[17] H. Chen, C.-C. Leung, L. Xie, *et al.*, "Parallel inference of Dirichlet process Gaussian mixture models for unsupervised acoustic modeling: A feasibility study," in *Proc. of Interspeech*, 2015.

[18] Y. Zhang and J. R. Glass, "Towards multi-speaker unsupervised speech pattern discovery," in *Proc. of IEEE ICASSP*, 2010.

[19] M. Versteegh, X. Anguera, A. Jansen, and E. Dupoux, "The Zero Resource Speech Challenge 2015: Proposed approaches and results," *Procedia Computer Science*, vol. 81, pp. 67–72, 2016.

[20] M. Heck, S. Sakti, and S. Nakamura, "Unsupervised linear discriminant analysis for supporting DPGMM clustering in the zero resource scenario," *Procedia Computer Science*, vol. 81, pp. 73–79, 2016.

[21] G. Synnaeve and E. Dupoux, "A temporal coherence loss function for learning unsupervised acoustic embeddings," *Procedia Computer Science*, vol. 81, pp. 95–100, 2016.

[22] A. Jansen, S. Thomas, and H. Hermansky, "Weak top-down constraints for unsupervised acoustic model training.," in *Proc. of IEEE ICASSP*, 2013.

[23] A. Jansen and B. Van Durme, "Efficient spoken term discovery using randomized algorithms," in *Proc. of IEEE ASRU*, 2011.

[24] A. Jansen and K. Church, "Towards unsupervised training of speaker independent acoustic models.," in *Proc. of Interspeech*, 2011.

[25] G. Synnaeve, T. Schatz, and E. Dupoux, "Phonetics embedding learning with side information," in *Proc. of IEEE SLT*, 2014.

[26] R. Thiolliere, E. Dunbar, G. Synnaeve, *et al.*, "A hybrid dynamic time warping-deep neural network architecture for unsupervised acoustic modeling," in *Proc. of Interspeech*, 2015.

[27] N. Zeghidour, G. Synnaeve, M. Versteegh, and E. Dupoux, "A deep scattering spectrum–deep siamese network pipeline for unsupervised acoustic modeling," in *Proc. of IEEE ICASSP*, 2016.

[28] H. Kamper, M. Elsner, A. Jansen, and S. Goldwater, "Unsupervised neural network based feature extraction using weak top-down constraints," in *Proc. of IEEE ICASSP*, 2015.

[29] D. Renshaw, H. Kamper, A. Jansen, and S. Goldwater, "A comparison of neural network methods for unsupervised representation learning on the Zero Resource Speech Challenge," in *Proc. of Interspeech*, 2015.

[30] J. Bromley, I. Guyon, Y. LeCun, *et al.*, "Signature verification using a "siamese" time delay neural network," *Int. J. Pattern Recogn. Artif. Intell.*, vol. 7, no. 4, pp. 669–688, 1993.

[31] D. M. Endres and J. E. Schindelin, "A new metric for probability distributions," *IEEE Trans. on Inf. Theory*, vol. 49, no. 7, pp. 1858–1860, 2003.

[32] M. Pitt, L. Dilley, K. Johnson, *et al.*, *Buckeye corpus of conversational speech (2nd release)*, Online, Columbus, OH: Department of Psychology, Ohio State University (Distributor), 2007.

[33] E. Barnard, M. H. Davel, C. J. van Heerden, *et al.*, "The NCHLT speech corpus of the South African languages.," in *Proc. of SLTU*, 2014.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[35] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of ICLR*, 2015.

[36] R. Al-Rfou, G. Alain, A. Almahairi, *et al.*, "Theano: a Python framework for fast computation of mathematical expressions," *ArXiv e-prints*, vol. abs/1605.02688, 2016.

[37] B. McFee, M. McVicar, O. Nieto, *et al.* (2017). Librosa 0.5.0, [Online]. Available: `https://github.com/librosa/librosa`.

[38] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A LLVM-based Python JIT compiler," in *Proc. of LLVM-HPC*, 2015.

[39] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[40] T. Schatz, V. Peddinti, F. Bach, *et al.*, "Evaluating speech features with the minimal-pair ABX task: Analysis of the classical MFC/PLP pipeline," in *Proc. of Interspeech*, 2013.