



Discovering discrete subword units with Binarized Autoencoders and Hidden-Markov-Model Encoders

Leonardo Badino¹, Alessio Mereta¹, Lorenzo Rosasco^{2,3}

¹RBCS, Istituto Italiano di Tecnologia, Italy

²DIBRIS, Università degli Studi di Genova, Italy

³LCSL, Istituto Italiano di Tecnologia and Massachusetts Institute of Technology, USA

leonardo.badino@iit.it, alessio.mereta@gmail.com, lrosasco@mit.edu

Abstract

In this paper we address the problem of unsupervised learning of discrete subword units. Our approach is based on Deep Autoencoders (AEs), whose encoding node values are thresholded to subsequently generate a symbolic, i.e., 1-of-K (with $K = \text{No. of subwords}$), representation of each speech frame. We experiment with two variants of the standard AE which we have named Binarized Autoencoder and Hidden-Markov-Model Encoder. The first forces the binary encoding nodes to have a U-shaped distribution (with peaks at 0 and 1) while minimizing the reconstruction error. The latter jointly learns the symbolic encoding representation (i.e., subwords) and the prior and transition distribution probabilities of the learned subwords.

The ABX evaluation of the Zero Resource Challenge - Track 1 shows that a deep AE with only 6 encoding nodes, which assigns to each frame a 1-of-K binary vector with $K = 2^6$, can outperform real-valued MFCC representations in the across-speaker setting. Binarized AEs can outperform standard AEs when using a larger number of encoding nodes, while HMM Encoders may allow more compact subword transcriptions without worsening the ABX performance.

Index Terms: unsupervised acoustic modeling, auto-encoders, deep learning

1. Introduction

Although Automatic Speech Recognition is mostly a data-driven technology, typical ASR systems rely on linguistic resources, i.e., pronunciation lexicon and phonetically transcribed/annotated speech training data. Those resources are built upon a prescriptive inventory of subword units (typically the phone set). A variety of strategies has been proposed that rely on lower resources [1]. These range from grapheme-based ASR [2] which avoid learning a phone set, to strategies that explicitly discover an inventory of subword units.

Our approach falls in the latter category and is based on Autoencoder (AE) neural networks. The contribution of the present work consists of two novel methods that aim at removing some AE limitations for discrete subword learning.

Most of the proposed approaches to subword units generation are based on Gaussian Mixture Models (GMMs) [3, 4, 5, 6, 7, 8, 9]. For example, in [9] subwords are generated by iteratively splitting an initial single-state Hidden Markov Model with Gaussian output probabilities. In [6, 7] the unsupervised acoustic modeling problem is divided into three sub-tasks: segmentation, segment clustering and acoustic modeling of the clusters. In [7] the three sub-tasks are carried out at once using a Dirichlet Process mixture model.

In [10] we proposed an approach to subword learning based on AEs. Contrary to GMMs, AEs can identify non-linear dependencies between acoustic features and, consequently, distinctive phonetic properties that emerge from such dependencies. In [10] we showed that AE-extracted features are more discriminative than GMM-extracted features in a spoken query classification task.

AEs and more generally, neural network architectures have been proposed in low resource settings for ASR and related tasks [11, 12], although they do not explicitly aim at discovering discrete subwords.

In the present work we propose two variants to the standard AE which we have named Binarized Autoencoder and Hidden-Markov-Model Encoder. The first attempts to directly extract discrete binary features (i.e., with values $\{0, 1\}$) while minimizing the reconstruction error. The latter jointly learns the discrete representation (i.e., subwords) and the prior and transition distribution probabilities of the learned subwords. Both strategies have been tested on the ABX evaluation of the Zero Resource Challenge - Track 1 [13].

2. An Autoencoder-based approach to subword learning

2.1. Autoencoders

An AE consists of an encoding and a decoding part [14]. The encoder maps an input vector \mathbf{x} into an hidden/encoding representation \mathbf{h} :

$$\mathbf{h} = f_{\theta}(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (1)$$

where \mathbf{W} is a weight matrix, \mathbf{b} a bias vector and s is typically the logistic sigmoid function.

The decoder maps back the hidden vector \mathbf{h} to a "reconstructed" input \mathbf{x}' :

$$\mathbf{x}' = g_{\theta'}(\mathbf{h}) = l(\mathbf{W}'\mathbf{h} + \mathbf{b}') \quad (2)$$

If the input data is assumed to be Gaussian distributed, as in the present work, l is typically an identity function and the AE is trained (through backpropagation) to minimize the squared error $E = \|\mathbf{x} - \mathbf{x}'\|^2$, which is equivalent to maximizing $\ln p(\mathbf{x}|\mathbf{x}'(\mathbf{x}))$. In the present work we use Deep, i.e. multi-layered, AEs which consist of more than one hidden layer. They are first pretrained using stacks of Restricted Boltzmann Machines and then finetuned using backpropagation [15].

2.2. Autoencoders for subword learning

Our basic AE-based approach is depicted in Figure 1. The encoder of the AE extracts a new representation \mathbf{h}_n

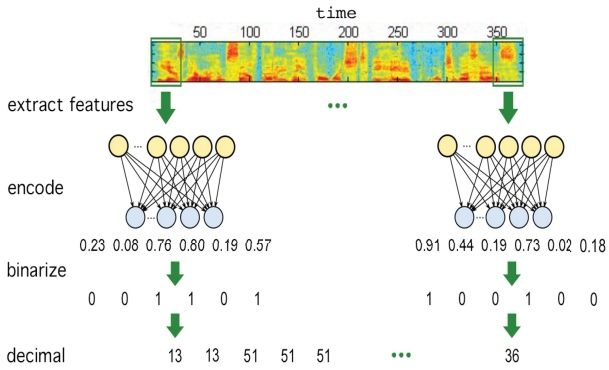


Figure 1: Overview of the AE-based approach to subword learning.

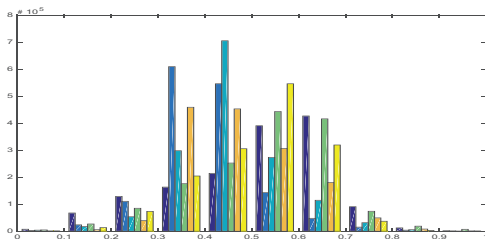


Figure 2: 10-bin histograms of the values of the 6 encoding nodes of a 5-hidden layer AE trained on Tsonga data.

for the n -th speech frame. The elements of $\mathbf{h}_n = [h_n(1), \dots, h_n(i), \dots, h_n(H)]$ (with $H = \text{No. of encoding nodes}$) are real-valued and lie in the $[0, 1]$ range. The $h_n(i)$ values can be seen as the probability of the i -th encoding node to activate (i.e., to take value 1).

Subsequently the real-valued \mathbf{h}_n is binarized. All elements of \mathbf{h}_n with values ≤ 0.5 are set to 0 while all the remaining elements are set to 1, resulting in the binary vector $\mathbf{b}_n = [b_n(1), \dots, b_n(i), \dots, b_n(H)]$. That produces $K = 2^H$ possible binary configurations/states. The states are the subwords learned by the AE. We can also represent a state/subword as a 1-of- K vector, \mathbf{z}_n . \mathbf{z}_n is a K -dimensional vector where its z_k element equals 1 and all the remaining elements equal 0 (in Figure 1 \mathbf{z}_n is represented by the decimal number k).

2.2.1. Binarized Autoencodes

After examination of the values taken by the encoding nodes of deep AEs trained on the Zero Resource Challenge - Track 1 data [13], we discovered that encoding node values tend to gather around the threshold 0.5 as shown in Figure 2 where histograms of the values of 6 encoding nodes are plotted. Such behaviour is unfortunate in that small variations around the threshold produce different subwords, if most of the encoding values are grouped around the threshold then different subwords can be hardly distinguishable.

To avoid or, at least limit, such behaviour we have developed two training strategies that should force the encoding nodes to tend towards a U-shaped distribution (with peaks at 0 and 1).

In the first strategy (henceforth Bin1) we add Gaussian noise to the input of each encoding unit, as proposed in [16].

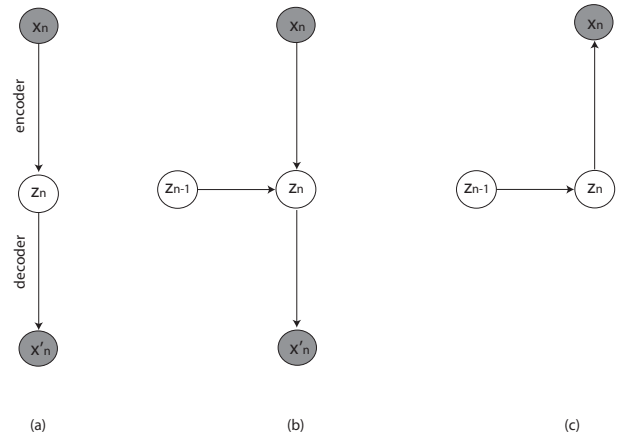


Figure 3: Graphical Model representation of (a) an Autoencoder, (b) an AE with first-order Markov dependencies between the encoding states \mathbf{z} , (c) a “HMM-Encoder”.

In order to prevent the added noise from disturbing backpropagation based fine-tuning we use a “deterministic” noise, where sampled noise values are fixed in advance for each training example and do not change during training. The rationale behind such a strategy is that it encourages the input received from the previous layer to be large and positive for some examples and large and negative for others, so that similar examples can produce similar activations despite the noise.

In the second strategy (henceforth Bin2) we explicitly penalize the activations of the encoding nodes that are near 0.5. To do so, we add a binary entropy term to the cost function for each i -th encoding node:

$$g(h(i)) = -h(i) \ln h(i) - (1 - h(i)) \ln(1 - h(i)) \quad (3)$$

In this way the activations are encouraged to stay near 0 or 1.

2.3. Hidden-Markov-Model Encoders

An approach based on AE ignores the sequential nature of speech and inter-subword dependencies. On a more practical side, taking into account inter-subword dependencies may help to reduce inconsistent subword transcriptions, i.e., examples of the same word type can have largely different subword transcriptions because of very frequent subword transitions at the frame level. The problem of inconsistent transcriptions might not only affect the ABX evaluation but can also be critical if we want to build a grapheme-to-subword mapping, which is necessary for an ASR based on automatically learned subwords. Note that one of the causes of very frequent subword transitions can be the threshold problem mentioned above, which is addressed by the Binarized AEs.

In [10] we smoothed subword transitions by introducing a subword transition penalty which increases the expected subword dwell time. Here we propose a different approach where we jointly fine-tune the AE parameters (more specifically the encoding parameters) and learn subword transition probabilities and subword prior probabilities.

To accomplish that we introduce first-order Markov dependencies between subwords \mathbf{z}_k . Figure 3a shows a Graphical Model representation of an AE. Introducing first-order Markov dependencies results in the graphical model of Figure 3b, which

resembles an input-output HMM with no connections between the two observed variables \mathbf{x} and \mathbf{x}' (which are actually identical, being \mathbf{x}' a copy of \mathbf{x}). Such graphical model makes the probability of \mathbf{z}_n to be dependent on both \mathbf{z}_{n-1} and \mathbf{x}_n . That would require to train new elements, e.g., neural nets as in the input-output HMM. We wanted to have an explicit form for $p(\mathbf{z}_n|\mathbf{z}_{n-1})$ and keep a simpler model like the one depicted in Figure 3c, an HMM where output probabilities are determined by the AE, specifically by its encoding part.

The HMM can be trained using Expectation-Maximization (EM) [17] which maximizes:

$$Q(\phi, \phi^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \phi^{old}) \ln p(\mathbf{X}, \mathbf{Z}|\phi) \quad (4)$$

where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \dots, \mathbf{x}_N\}$, $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n, \dots, \mathbf{z}_N\}$ and ϕ represents all the learning parameters.

$Q(\phi, \phi^{old})$ can be written as ([18]):

$$\begin{aligned} \sum_{k=1}^K \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^N \sum_{j=1}^K \sum_{k=1}^K \xi(z_{n-1,j}, z_{nk}) \ln A_{j,k} \\ + \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln p(\mathbf{x}_n | z_{nk}, \theta) \end{aligned} \quad (5)$$

where $\pi_k = p(z_{1k} = 1)$, $A_{jk} = p(z_{nk} = 1 | z_{n-1,j} = 1) \forall n$, $\gamma(z_{nk}) = p(z_{nk} = 1 | \mathbf{X}, \phi^{old})$, $\xi(z_{n-1,j}, z_{nk}) = p(z_{n-1,j} = 1, z_{nk} = 1 | \mathbf{X}, \phi^{old})$.

The only difference with the maximization of Q of a HMM that uses output Gaussian density probabilities resides in the maximization of the last term w.r.t. the learning parameters θ .

Using the Bayesian rule we aim at finding the encoder parameters θ that maximize:

$$\begin{aligned} \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln p(\mathbf{x}_n | z_{nk}, \theta) \\ \simeq \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln \frac{p(z_{nk} | \mathbf{x}_n, \theta)}{p(\mathbf{z}_n | \phi^{old})} p(\mathbf{x}_n) \\ = \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln \frac{p(z_{nk} = 1 | \mathbf{h}_n(\mathbf{x}_n), \theta)}{p(\mathbf{z} | \phi^{old})} p(\mathbf{x}_n) \end{aligned} \quad (6)$$

where $p(\mathbf{z}_n) = p(\mathbf{z}) \forall n$ and we considered the $p(\mathbf{z})$ computed after the Expectation step (thus it is a constant in the Maximization step). $p(\mathbf{x}_n) (= p(\mathbf{h}_n))$ does not affect maximization w.r.t. θ . We focus on $\sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln p(z_{nk} = 1 | \mathbf{h}_n, \theta)$ and ignore for the moment $p(\mathbf{z})$ to simplify equations.

Assuming that the probabilities of activation $h(i)$ are independent then:

$$p(z_k = 1 | \mathbf{h}) = \prod_i^H h^{b_k(i)}(i) (1 - h(i))^{(1 - b_k(i))} \quad (7)$$

where we have removed the index n to simplify notation. Combining the last two equations:

$$\begin{aligned} \sum_{k=1}^K \gamma(z_k) \ln p(z_k = 1 | \mathbf{h}) \\ = \sum_{k=1}^K \gamma(z_k) \sum_i^H b_k(i) \ln h(i) + (1 - b_k(i)) \ln(1 - h(i)) \\ = \sum_i^H E_{\gamma(\mathbf{z})}[b(i)] \ln h(i) + (1 - E_{\gamma(\mathbf{z})}[b(i)]) \ln(1 - h(i)) \end{aligned} \quad (8)$$

where $E_{\gamma(\mathbf{z})}[b(i)]$ is the expected value of $b(i)$ w.r.t. $\gamma(\mathbf{z})$.

Reintroducing n and dividing $\gamma(z_{nk})$ by $p(\mathbf{z})$ we have:

$$\sum_n^N \sum_i^H E_{\gamma(\mathbf{z}_n)}[b_n(i)] \ln h_n(i) + (1 - E_{\gamma(\mathbf{z}_n)}[b_n(i)]) \ln(1 - h_n(i)) \quad (9)$$

where $\hat{\gamma}(\mathbf{z}_n)$ is the scaled version of $\gamma(\mathbf{z}_n)$. Equation 9 is the error function that the encoder part of the AE has to minimize. We can easily compute the gradient for backpropagation as it is a negative cross-entropy function.

Using the decoder to maximize the last term of equation 5 would be much more time consuming as it requires to compute $E_{\gamma(\mathbf{z}_n)}[x'_n(i)]$ every time that the decoding parameters are updated.

Finally note that the HMM-based fine-tuning of the encoder is applied only after having trained the AE as described in section 2.1.

3. Experimental setup

The acoustic input to most of the AEs was a 60 ($20 + \Delta + \Delta\Delta$) mel-scaled filterbank coefficients (fbanks) vector extracted from speech signal previously segmented into 25 ms Hamming windows sampled every 10 ms. The input variables were normalized to have 0 mean and 1 standard deviation. AEs had 5 hidden layers with either a 100-50-6-50-100 or a 100-50-12-50-100 architecture ($H=6$ or 12). 6 was chosen because 2^6 is close to the cardinality of a phone set, while 12 was selected to compare real-valued representations with the MFCCs baseline of the Zero Resources Challenge. AEs with a smaller number of hidden layers produced poorer results in the ABX.

Concerning the HMM Encoder we first trained a (standard) AE and then changed its parameters within a maximum of 10 iterations of the EM algorithm. At iteration 1 the prior distribution $p(\mathbf{z} | \phi^{old})$ was initialized as a uniform distribution. That results in a non-scaled $p(z_{nk} = 1 | \mathbf{h}_n, \theta)$ which (initially) increases the number of the most probable subwords and largely reduces the number of subword transitions within an utterance. During training we always observed the loglikelihood increasing at each iteration, however the best ABX scores were achieved after 2/3 iterations.

Experiments were carried out on the Tsonga and English datasets of the Zero Resource Speech Challenge 2015.

4. Results

For each system we ran 3 ABX evaluations (with cosine similarity) depending on the encoding representation: Real Valued (RV) based on \mathbf{h} , Soft evaluation based on \mathbf{b} and Hard evaluation based on \mathbf{z} . The latter evaluation is named Hard as frame similarity is not null only when subwords are identical, while in the Soft evaluation shared binary values (which we can consider as shared properties) between subwords are taken into account. The Soft evaluation is particularly useful when K is large and similarity between 1-of- K vectors is rarely different from 0. Finally, note that the H real-valued is a lower dimensionality ($H \ll K = 2^H$ for $H > 6$) representation of the posteriors of the K subwords.

Tables 1 and 2 show the results of the ABX tests on Tsonga and English data respectively. One of the most remarkable results is shown in the Tsonga table where a standard AE with $H = 6$, corresponding to $K=64$ discrete subwords, and its HMM-Encoder variant, outperform (Hard evaluation) the real valued baseline in the across speakers setting.

System	Within			Across		
	RV	Soft	Hard	RV	Soft	Hard
MFCCs	19.1	-	-	33.8	-	-
AE6	14.1	17.9	21.5	23.6	27.7	32.4
AE6 Bin2	27.9	36.1	35.9	33.2	38.9	38.7
HAE6	-	-	22.5	-	-	32.0
AE12	16.0	21.6	33.4	27.4	31.9	43.1
AE12 Bin1	17.1	18.0	32.0	20.2	26.4	40.8
AE12 Bin2	21.9	25.2	25.4	32.1	33.5	34.6

Table 1: *ABX error rates within and across speakers on Tsonga data. MFCCs is the Challenge baseline consisting of 13 MFCCs.*

System	Within			Across		
	RV	Soft	Hard	RV	Soft	Hard
MFCCs	15.6	-	-	28.1	-	-
AE6	17.3	19.7	23.1	26.3	27.7	32.2
HAE6	-	-	30.2	-	-	35.3
AE12	16.7	20.2	31.0	26.8	30.1	40.3
AE12 Bin1	19.6	19.7	30.1	29.1	28.7	40.2
AE12 Bin2	23.0	24.4	25.2	30.5	31.5	32.6

Table 2: *ABX error rates on English data.*

Comparing standard AEs vs. Binarized AEs, standard AEs with 6 encoding nodes often outperform any other AE. Binarized AEs perform poorly when $H = 6$. However when $H = 12$ the Binarized version almost always outperforms its standard counterpart (especially when representations are discrete). A possible explanation is that Binarized AEs do remove a lot information and when using only 6 nodes such reduction is extreme. Additionally, when $H = 6$ the Bin2 strategy completely fails to produce U-shaped distributions at the encoding level and some units saturate. With 12 encoding nodes, Binarized AEs do not present bell-shaped distributions centered at 0.5 (as in Figure 2) and can produce U-shaped distributions (Figure 4). That explains why Binarized AEs exhibit smaller performance decrease than standard AEs when moving from RV to Soft evaluation.

System	Dataset	AST	ABX	
			Within	Across
AE6	SampleEnglish	50	32.7	33.4
HAE6	SampleEnglish	69	34.6	33.4
AE6	English	27929	23.1	32.2
HAE6	English	16652	30.2	35.3
AE6	Tsonga	150	21.5	32.4
HAE6	Tsonga	72	22.5	32.0

Table 3: *Comparison between AE and HMM-Encoder. AST is the average number of subword transitions within the data files of the Zero Resource Challenge.*

Table 3 compares standard AEs and HMM-Encoders. Despite HMM-Encoders drastically reduce the average number of subword transitions they can even perform slightly better than AEs (in Tsonga). We consider this as an interesting result as in [10] more “compact” subword transcriptions always resulted in poorer performance in a word classification task. As mentioned above, more compact transcriptions are advantageous for different reasons, e.g. they can facilitate the letter-to-subword

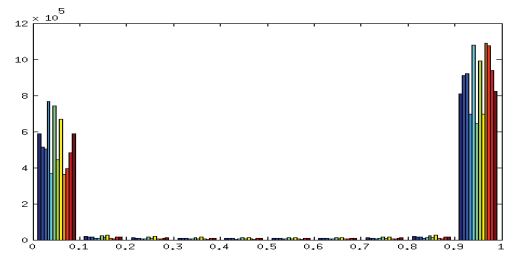


Figure 4: *10-bin histograms of the values of 12 encoding nodes of a Bin1 AE trained on Xitsonga data.*

mapping.

Finally we wanted to investigate what phonetic distinctive features a deep AE can learn. We trained a standard AE6 on the single-speaker mngu0 dataset [19] and computed a co-occurrence matrix (with bi-clustering as in [12]) between the learned subwords and phones. Figure 5 shows that a deep AE can effectively learn distinctive features such as $[\pm \text{vowel}]$, $[\pm \text{nasal}]$, $[\pm \text{plosive}]$, $[\pm \text{fricative}]$.

5. Conclusions

In this paper we address the problem of unsupervised learning of discrete subwords using neural network Autoencoders. We propose two strategies, namely Binarized Autoencoders and Hidden-Markov-Model Encoders. The first attempts to directly extract subwords, in the form of binary vectors, while minimizing the reconstruction error. The latter jointly learns the discrete subwords and the prior and transition distribution probabilities of the learned subwords. Results show that Binarized AEs can outperform standard AEs when using a sufficiently large number of encoding nodes (e.g., 12), while HMM Encoders can allow more compact subword transcriptions without worsening (or even improving) the ABX performance.

6. Acknowledgements

The authors acknowledge the support of the European Commission project POETICON++ (grant agreement 288382).

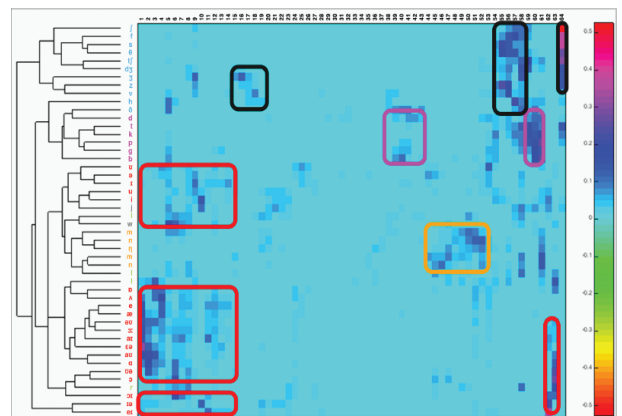


Figure 5: *Co-occurrence matrix between phones and the automatically discovered subwords on the mngu0 dataset.*

7. References

- [1] J. Glass, "Towards unsupervised speech processing," in *Proceedings of ISSPA, Montreal, Canada*, 2012.
- [2] M. Killer, S. Stuker, and T. Schultz, "grapheme based speech recognition," in *Proceedings of Eurospeech, Geneva, Switzerland*, 2003.
- [3] M. Bacchiani and M. Ostendorf, "Joint lexicon, acoustic unit inventory and model design," *Speech Communication*, vol. 29, pp. 99–114, 1999.
- [4] R. Singh, B. Raj, and M. Stern, "Automatic generation of subword units for speech recognition systems," *IEEE T-ASLP*, vol. 10, no. 2, pp. 89–99, 2002.
- [5] A. Jansen and K. Church, "Towards unsupervised training of speaker independent acoustic models," in *Proceedings of INTER-SPEECH*, 2011, pp. 1693–1696.
- [6] A. Garcia and H. Gish, "Keyword spotting of arbitrary words using minimal speech resources," in *Proceedings of ICASSP*, 2006.
- [7] C. Lee and J. Glass, "A nonparametric bayesian approach to acoustic model discovery," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 2012.
- [8] C. Lee, Y. Zhang, and J. Glass, "Joint learning of phonetic units and word pronunciations for asr,," in *Proceedings of EMNLP*, 2013.
- [9] B. Varadarajan, S. Khudanpur, and E. Dupoux, "Unsupervised learning of acoustic sub-word units," in *Proceedings of ACL-08:HLT Short Papers*, 2008, pp. 162–168.
- [10] L. Badino, C. Canevari, L. Fadiga, and G. Metta, "An auto-encoder based approach to unsupervised learning of subword units," in *Proceedings of ICASSP*, 2014.
- [11] H. Kamper, M. Elsner, A. Jansen, and S. Goldwater, "Unsupervised neural network based feature extraction using weak top-down constraints," in *Proceedings of ICASSP*, 2015.
- [12] G. Synnaeve, T. Schatz, and E. Dupoux, ". phonetics embedding learning with side information." in *Proceedings of IEEE SLT*, 2014.
- [13] The zero resource speech challenge. [Online]. Available: <http://www.lscp.net/persons/dupoux/bootphon/zerospeech2014/website/>
- [14] D. Plaut and G. Hinton, "Learning sets of filters using back-propagation," *Computer Speech and Language*, vol. 2, pp. 35–61, 1987.
- [15] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [16] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.
- [17] A. P. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1997.
- [18] C. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.
- [19] K. Richmond, P. Hoole, and S. King, "Announcing the electromagnetic articulography (day 1) subset of the mngu0 articulatory corpus," in *Proceedings of Interspeech*, 2011.