INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Towards unsupervised phone and word segmentation using self-supervised vector-quantized neural networks

*Herman Kamper**, *Benjamin van Niekerk**

Department of E&E Engineering, Stellenbosch University, South Africa
`kamperh@sun.ac.za, benjamin.l.van.niekerk@gmail.com`

Abstract

We investigate segmenting and clustering speech into low-bitrate phone-like sequences without supervision. We specifically constrain pretrained self-supervised vector-quantized (VQ) neural networks so that blocks of contiguous feature vectors are assigned to the same code, thereby giving a variable-rate segmentation of the speech into discrete units. Two segmentation methods are considered. In the first, features are greedily merged until a prespecified number of segments are reached. The second uses dynamic programming to optimize a squared error with a penalty term to encourage fewer but longer segments. We show that these VQ segmentation methods can be used without alteration across a wide range of tasks: unsupervised phone segmentation, ABX phone discrimination, same-different word discrimination, and as inputs to a symbolic word segmentation algorithm. The penalized dynamic programming method generally performs best. While performance on individual tasks is only comparable to the state-of-the-art in some cases, in all tasks a reasonable competing approach is outperformed at a substantially lower bitrate.

Index Terms: unsupervised speech processing, phone segmentation, word segmentation, acoustic unit discovery, zero-resource.

1. Introduction

Methods for automatically learning phone- or word-like units from unlabelled speech audio could enable speech technology in severely low-resourced settings [1, 2] and could lead to new cognitive models of human language acquisition [3–5]. The goal in unsupervised representation learning of phone units is to learn features which capture phonetic contrasts while being invariant to properties like the speaker or channel. Early approaches focussed on learning continuous features [6–10]. In an attempt to better match the categorical nature of true phonetic units, more recent work has considered *discrete* representations [11–17]. One approach is to use a self-supervised neural network with an intermediate layer that quantizes features using a learned codebook [12]. While the discrete codes from such vector quantized (VQ) networks have given improvements in intrinsic phone discrimination tasks, they still encode speech at a much higher bitrate than true phone sequences [18].

As an example, the top of Figure 1 shows the code indices from a vector-quantized variational autoencoder (VQ-VAE) [19] overlaid on the input spectrogram. While there is some correspondence between the code assignments and the true phones (e.g. code 31 in both occurrences of [s]), and although there is some repetition of codes in adjacent frames (e.g. in [n] and [ae]), the input speech are often assigned to codes that are distinct from those of surrounding frames. This is not surprising since the VQ model is not explicitly encouraged to do so. The result is an encoding at a much higher bitrate (around 400 bits/sec) than that of true phone sequences (which is around 40 bits/sec [18, 20]).

In this paper we consider ways to constrain VQ models so that contiguous feature vectors are assigned to the same code, resulting in a low-bitrate segmentation of the speech into discrete units. We specifically compare two VQ segmentation methods. Both of these are based on a recent method for segmenting written character sequences [21]. The first method is a greedy approach, where the closest adjacent codes are merged until a set number of segments are reached. The second method allows for an arbitrary number of segments. A squared error between blocks of feature vectors and VQ codes are used together with a penalty term encouraging longer-duration segments. The optimal segmentation is found using dynamic programming.

We apply these two segmentation approaches using the encoders and codebooks of the two VQ models from [22]. The first is a type of VQ-VAE. The second is a vector-quantized contrastive predictive coding (VQ-CPC) model. The combination of these two models with the two segmentation approaches gives a total of four VQ segmentation models to consider. We evaluate these on four different tasks: unsupervised phone segmentation [23], ABX phone discrimination [24], same-different word discrimination [25], and as inputs to a symbolic word segmentation algorithm [1]. The last-mentioned is particularly important since the segmentation and clustering of word-like units remains a major but important challenge [5, 26].

On most metrics in the four tasks the combination of the VQ-VAE with the penalized dynamic programming approach is the best VQ segmentation method. Example output is shown in the middle of Figure 1. Compared to other existing methods, it does not achieve state-of-the-art performance in all four evaluation tasks. However, it achieves reasonable performance at a much lower bitrate than most existing methods. This is noteworthy since, while most of the other methods have been tailored to the respective tasks, a single VQ segmentation approach can be used without any alteration directly in a range of problems.

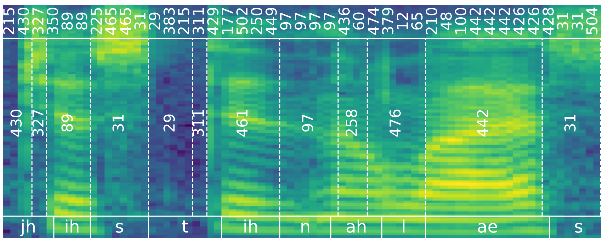

Figure 1: *The phrase 'just in the last' with the code indices from a VQ-VAE model (top) and the ground truth phones (bottom). The indices with the dashed segmentation (middle) is the result of applying duration-penalized dynamic programming segmentation on the VQ-VAE codes.*

*The two authors contributed equally.

http://dx.doi.org/10.21437/Interspeech.2021-50

2. Vector-quantized neural networks

A vector quantization (VQ) layer [19] consists of a trainable codebook $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_K\}$ of K distinct codes. In the forward pass, the layer discretizes a sequence of continuous feature vectors $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T)$ by mapping each $\mathbf{z}_t$ to it's nearest neighbour in the codebook, i.e. each $\mathbf{z}_t$ is replaced with $\mathbf{e}_k$, where $k = \arg\min_i ||\mathbf{z}_t - \mathbf{e}_i||^2$. The layer output is the resulting quantized sequence $(\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \ldots, \hat{\mathbf{z}}_T)$. Since the $\arg\min$ operator is not differentiable, gradients are approximated using the straight-through estimator [27] in the backward pass. The codebook is trained using an exponential moving average of the continuous features. A commitment cost is also added to encourage each $\mathbf{z}_t$ to commit to its selected code.

In this paper we use the pretrained encoders and codebooks of the two self-supervised VQ models from [22]. Both models' encoders take log-Mel spectrograms, downsamples it by a factor of two, and discretizes the feature vectors using a VQ layer with 512 codes. The first model is a type of vector-quantized variational autoencoder (VQ-VAE).[1] The VQ-VAE is trained to encode speech into a discrete latent space from which it reconstructs the original audio waveform using an autoregressive decoder [12]. The second model[2] combines vector quantization with contrastive predictive coding (VQ-CPC), similar to the models of [15, 28]. Using a contrastive loss [29], the VQ-CPC model is trained to identify future codes from among a set of negative examples drawn from other utterances.

3. VQ segmentation algorithms

The segmentation process consists of two steps. First, we extract a sequence of continuous feature vectors using a pretrained encoder. Next, we solve a constrained optimization problem to divide the continuous representation into segments. Figure 2 illustrates the steps, showing two possible segmentations. Each segment is assigned a representative vector chosen from the codebook. We can score the segmentations by summing the (squared) distances between the continuous feature vectors and the representative code of each segment (this corresponds to adding up the lengths of the arrowed lines in Figure 2). The objective would then be to find the segmentation that minimizes the summed distance. One problem with blindly following this approach is that the best segmentation will always put each $\mathbf{z}_t$ in its own segment, assigning it to the code closest to $\mathbf{z}_t$. This would just be the standard VQ layer (§2). Constraints are therefore required.

Formally, suppose sequence $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T)$ is segmented into disjoint contiguous segments $(s_1, s_2, \ldots, s_M)$. Each segment s_i is a subsequence of $\mathbf{z}_{1:T}$, consisting of $|s_i|$ feature vectors. These vectors are aggregated and the whole segment s_i is assigned to a single representative vector $\hat{\mathbf{z}}_{s_i}$, selected from the VQ codebook $\{\mathbf{e}_k\}_{k=1}^K$. So, as a concrete example using this notation, we might have $s_2 = \mathbf{z}_{3:5}$ with $\hat{\mathbf{z}}_{s_2} = \mathbf{e}_7$, meaning that all three vectors $(|s_2| = 3)$ in the second segment are assigned to the seventh code.

Let's use $E(\mathbf{z}_{1:T}, s_{1:M})$ to denote the error for a particular segmentation $s_{1:M}$ of the features $\mathbf{z}_{1:T}$. One option for the error function could be

$$E(\mathbf{z}_{1:T}, s_{1:M}) = \sum_{s_i \in s_{1:M}} \sum_{\mathbf{z}_j \in s_i} ||\mathbf{z}_j - \hat{\mathbf{z}}_{s_i}||^2. \qquad (1)$$

The problem with this error function, as noted above, is that the optimal strategy would simply be to place each $\mathbf{z}_t$ in its own segment. The solution is to add a penalty term that encourages longer but fewer segments:

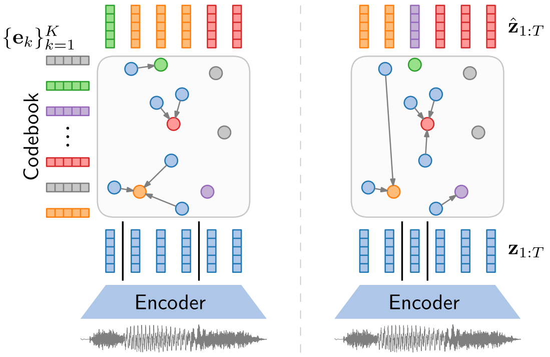

Figure 2: *Two potential segmentations of an utterance. The features in each segment are assigned to the same code. The left segmentation will have a smaller sum of (squared) distances between the features and their assigned codes than the right.*

$$E(\mathbf{z}_{1:T}, s_{1:M}) = \sum_{s_i \in s_{1:M}} \sum_{\mathbf{z}_j \in s_i} \left[||\mathbf{z}_j - \hat{\mathbf{z}}_{s_i}||^2 + \lambda \, \text{pen}(|s_j|) \right], \qquad (2)$$

where $\text{pen}(|s_j|)$ is a function penalizing the number of frames $|s_j|$ in a segment and λ is the penalty weight. Our objective is then to find the optimal segmentation $\arg\min_{s_{1:M}} E(\mathbf{z}_{1:T}, s_{1:M})$, and this can be accomplished using dynamic programming. Concretely, we define forward variables $\alpha_t \triangleq \min_{s_{1:M_t}} E(\mathbf{z}_{1:t}, s_{1:M_t})$ as the error for the optimal segmentation up to step t. This can be calculated recursively:

$$\alpha_t = \min_{j=1}^{t} \left\{ \alpha_{t-j} + \min_{k=1}^{K} \sum_{\mathbf{z}_i \in \mathbf{z}_{t-j+1:t}} \left[||\mathbf{z}_i - \mathbf{e}_k||^2 + \lambda \, \text{pen}(j) \right] \right\}.$$

We start with $\alpha_0 = 0$ and calculate α_t for $t = 1, \ldots, T-1$. We keep track of the optimal choice ($\arg\min$) for each α_t, and the overall optimal segmentation is then obtained by starting from the final position $t = T$ and moving backwards, repeatedly choosing the optimal boundary.

The above shortest-path (i.e. Viterbi) formulation is a generalization of the approach of [21]. Their starting point for the constraint on (1) is different, however. Instead of using a penalty term, they enforce a prespecified number of segments. I.e., while above M can take on an arbitrary number (depending on the penalty weight λ), they require a specific number of segments:

$$E(\mathbf{z}_{1:T}, s_{1:M}) = \sum_{s_i \in s_{1:M}} \sum_{\mathbf{z}_j \in s_i} ||\mathbf{z}_j - \hat{\mathbf{z}}_{s_i}||^2 \ \text{ s.t. } M = N, \qquad (3)$$

where N is the set number of segments.

Actually, if we set the penalty function in (2) to $\text{pen}(|s_j|) = 1 - |s_j|$, then it can be shown that the formulation in (2) is exactly the dual of (3). In our experiments we indeed use this penalty function—it gave reasonable results on development data and we did not experiment further. A more fundamental implementational difference here is that [21] doesn't actually optimise (3) directly. Instead they use an approximation where adjacent vectors are greedily merged until exactly N segments are obtained. The greedy approach has the advantage that a solution can be found in $\mathcal{O}(KT \log T)$ time,[3] while getting the exact solution using full dynamic programming takes $\mathcal{O}(KT^2N)$ time.

[1] https://github.com/bshall/ZeroSpeech

[2] https://github.com/bshall/VectorQuantizedCPC

[3] Our version of the greedy approach is slower than this, though, since we implement it with the help of an agglomerative clustering package.

4. Experimental setup

Both the VQ-VAE and VQ-CPC models (§2) are trained on the English training set from the *ZeroSpeech 2019 Challenge* [13], consisting of around 15 hours of speech from over 100 speakers. Depending on the task, evaluation is performed on the English test set from *ZeroSpeech 2019*, with around 30 minutes of speech, or on the Buckeye English development or test sets [30], each roughly 6 hours. There is no speaker overlap between these sets and the one used for training.

Hyperparameters for the VQ segmentation methods (§3) were tuned on Buckeye development data. The dynamic programming penalized method (denoted as 'DP penalized' in the tables and figures below) has one hyperparameter, λ, the duration penalty weight in (2). We set $\lambda = 3$ for the VQ-VAE and to $\lambda = 400$ for the VQ-CPC. For the greedy N-segmentation method (denoted as 'Greedy N-seg.' below), we need to specify the number of segments N that each utterance is segmented into. Instead of setting N directly, we rather set the required number of frames per segment: 3 gave the best development results.

We have four model combinations (VQ-VAE/VQ-CPC with greedy/DP penalized segmentation), which we evaluate on four different tasks. For the phone and word segmentation tasks below, we measure precision, recall and F-score of boundaries with a tolerance of 20 ms. We also measure over-segmentation (OS): how many more/fewer boundaries are proposed than the ground truth, e.g. an OS of –0.1 indicates we have 10% too few boundaries. F-score isn't always sensitive enough to the trade-off between recall and OS, which motivated the R-value metric [31]: it gives a perfect score (1) when a method has perfect recall (1) and perfect OS (0).

5. Experiments

5.1. Task 1: Phone segmentation

We perform phone segmentation experiments on Buckeye following the setup of [32]. In Table 1 we first compare the different VQ segmentation approaches to each other on development data. For the 'Merged' rows, repeated codes from the VQ models are simply collapsed. While this gives high recall, this is because of very high OS. Of the two VQ segmentation approaches, the dynamic programming variant (DP penalized) gives better results than the greedy version (Greedy N-seg.) in all cases except in OS on VQ-CPC codes. Using the VQ-VAE encoder and codebook is consistently better than using that of the VQ-CPC.

For testing, we therefore use the VQ-VAE with DP penalized segmentation. This is also the method in the middle of Figure 1. Table 2 compares this approach to existing methods. While our unsupervised approach does not perform as well as [32] on all metrics, it performs similarly or better than [33] and [34].

Table 1: *A comparison of different VQ segmentation algorithms for phone segmentation (%) on Buckeye development data. Segmentation of both VQ-CPC and VQ-VAE codes are shown.*

Model	VQ seg.	Prec.	Rec.	F	OS	R-val.
VQ-CPC:	Merged	29.7	**98.9**	45.7	232.6	–98.9
	Greedy N-seg.	51.5	65.6	57.7	27.4	56.2
	DP penalized	55.7	74.7	63.8	34.0	57.8
VQ-VAE:	Merged	32.0	98.3	48.3	207.2	–77.4
	Greedy N-seg.	57.1	72.9	64.0	27.8	61.2
	DP penalized	**66.4**	75.8	**70.8**	**14.1**	**72.4**

5.2. Task 2: ABX phone discrimination

Apart from segmentation performance, we want to see whether the resulting discrete sequences retain phonetic information, and whether it can do so at a low bitrate. For this we use the ABX phone discrimination task [24]. This tests whether triphone X is more similar to triphones A or B, where A and X are instances of the same triphone (e.g. 'cat'), while B differs in the middle phone (e.g. 'cut'). To measure speaker-invariance, A and B come from the same speaker, while X is taken from a different speaker. As a similarity metric, we use the average cosine distance along the dynamic time warping (DTW) alignment path over the segmented codes. ABX is reported as an aggregated error rate over all pairs of triphones in the test set.

Table 3 compares our approaches to some of the best submissions to *ZeroSpeech 2020* (a repeat of the 2019 challenge): the submission of [22] placed first on ABX followed by [16]. While VQ segmentation gives worse ABX than the unconstrained VQ-CPC and VQ-VAE, the bitrate is also dramatically reduced. Moreover, despite the drop, DP penalized segmentation still achieves better ABX than the closest competitor [16] (18.5% vs 20.2%) at roughly a third of the bitrate (106 vs 386).

An advantage of VQ segmentation is that the degree of compression can be controlled. Figure 3 illustrates the trade-off between phone discrimination performance and compression as the penalty weight parameter λ is varied (§4). The DP penalized VQ-VAE comes closer to the ideal of a 0% error rate at a 0 bitrate than submissions to either *ZeroSpeech 2019* or *2020*.

5.3. Task 3: Same-different word discrimination

In the same-different word discrimination task [25], we are given a pair of spoken words and we must decide whether they are examples of the same or different words. For every word pair in a

Table 2: *Phone segmentation results (%) on Buckeye test data for existing methods and VQ segmentation.*

Model	Prec.	Rec.	F	OS	R-val.
Unsupervised:					
Next-frame prediction [33]	69.3	65.1	67.2	–6.1	72.1
GRU gate activation [34]	69.6	72.6	71.0	–4.1	74.8
Self-sup. contrastive [32]	**75.8**	76.9	76.3	**–1.4**	**79.7**
Our VQ-VAE: DP penalized	70.8	**85.6**	**77.5**	20.9	74.8
Supervised:					
LSTM [35]	87.8	83.3	85.5	5.4	87.2
LSTM structured loss [36]	85.4	89.1	87.2	–4.1	88.8

Table 3: *ABX phone discrimination error rates and bitrates (bits/sec) on the ZeroSpeech 2019 English data.*

Model	ABX (%)	Bitrate
Unsupervised:		
MFCCs [14]	22.7	1738
Instance normalized WaveNet AE [16]	20.2	386
VQ-CPC [22]	**13.4**	421
VQ-VAE [22]	14.0	412
Our VQ-VAE: Greedy N-seg.	23.1	142
Our VQ-VAE: Duration penalized	18.5	**106**
Supervised:		
ASR output [18]	29.9	38

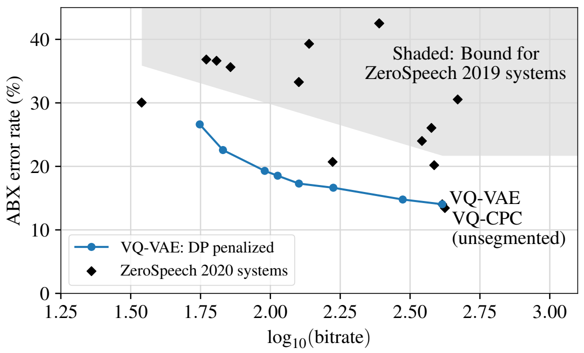

Figure 3: *ABX error rate as a function of bitrate on the ZeroSpeech 2019 English data. For the DP penalized method, bitrates are varied by changing the duration penalty weight λ.*

Table 4: *Average precision (AP, %) and bitrates (bits/sec) achieved in same-different word discrimination on Buckeye data. Only a selection of models are applied to the test data.*

	Development		Test	
Model	AP	Bitrate	AP	Bitrate
MFCCs	36.8	1721	35.9	1746
CAE [37]	47.4	1721		
Siamese [38, 39]	38.2	1721		
CTriamese [9]	**50.4**	1721	47.2	1746
VQ-CPC [22]	45.6	429		
VQ-VAE [22]	**50.4**	410	**47.8**	411
Our VQ-VAE: Merged	47.9	310		
Our VQ-VAE: Greedy N-seg.	25.7	135		
Our VQ-VAE: DP penalized	35.7	**118**	34.5	**118**

test set of pre-segmented words, the DTW distance is calculated using the feature representation under evaluation. Two words can then be classified as being of the same or different type based on some threshold, and a precision-recall curve is obtained by varying the threshold. The area under this curve is used as final evaluation metric, referred to as the average precision (AP).

Table 4 compares our approaches to existing unsupervised methods. We again see that VQ segmentation trades off word discrimination performance for compression, e.g. the DP penalized VQ-VAE achieves a similar AP to MFCCs at less than a tenth of the bitrate. Also note that the results for the unsegmented VQ-CPC and VQ-VAE given here are new—they have never been applied to this task. This is therefore the first time that it is shown that the unsegmented VQ-VAE achieve state-of-the-art word discrimination scores (47.8% on test) at a much lower bitrate than the current best approach [9] (411 vs 1746 on test).

5.4. Task 4: Towards word segmentation

Inspired by [1], we consider how word segmentation can be performed on top of VQ segmentation: code indices are provided as input to a symbolic word segmentation algorithm, normally used for segmenting phonemic or character sequences without word spaces. We use the adaptor grammar (AG) [40]. Since this is an initial experiment, we report development results. We also performed initial experiments with the unigram Dirichlet process model of [41] and a method based on thresholding the transition probabilities between code indices [42]. The thresholding approach performed worst while the Dirichlet process and AG performed similarly. We only report results for the latter.

Table 5: *Word boundary segmentation results (%) on Buckeye development data for existing methods and VQ segmentation code indices segmented with an adaptor grammar.*

Model	Prec.	Rec.	F	OS	R-val.
Existing methods:					
ES-KMeans [43]	30.7	18.0	22.7	**−41.2**	**39.7**
BES-GMM [44]	**31.7**	13.8	19.2	−56.6	37.9
Treating VQ seg. as word seg.:					
VQ-CPC: DP penalized	15.5	**81.0**	26.1	421.4	−266.6
VQ-VAE: DP penalized	15.8	68.1	25.7	330.9	−194.5
Adaptor grammar on top of:					
VQ-CPC: DP penalized	18.2	54.1	**27.3**	196.4	−86.5
VQ-VAE: DP penalized	16.4	56.8	25.5	245.2	−126.5

Table 5 shows word boundary segmentation scores for existing approaches (top); VQ segmentation where the code sequences are treated as words themselves (middle); and when the VQ segmentation code indices are segmented with an AG (bottom). While the VQ segmentation approaches (middle) achieve higher word boundary F-scores, they heavily over-segment, achieving much worse R-value scores than previous methods (top). When segmenting the code indices with an AG (bottom), we see that OS does improve compared to just treating the segmented codes as words (middle). However, our best approach (AG on top of VQ-CPC DP penalized indices) still heavily over-segments. Word token F-scores, where both the beginning and end boundaries need to be correctly predicted without an intermediate boundary, is very poor (in the order of 4%, not shown in the table). This is similar to the results in [1] (which was performed on a different dataset). As in [1], we conclude that a more integrated approach is required where word segmentation is (potentially) performed jointly with VQ segmentation.

6. Discussion and conclusion

We considered methods for constraining the outputs of vector-quantized (VQ) neural networks to produce a discrete low-bitrate segmentation of unlabelled speech. The experimental results on four different tasks show that our VQ segmentation method gives reasonable performance at a much lower bitrate than existing methods. While we found that the segmented codes from a VQ-VAE is normally slightly better than that from a VQ-CPC, it is worth noting that the latter is typically faster to train, since it does not require waveform reconstruction. Another practical consideration is that, while duration-penalized segmentation with dynamic programming typically outperformed the greedy approach where the number of segments are specified, the former has a more sensitive duration-weight hyperparameter, which could make the latter more useful in practice. In future work, we would like to further investigate whether the resulting discrete units match true phonetic units. We would also like to see whether VQ segmentation can be performed as part of a VQ network, rather than using pretrained encoders and codebooks.

7. Acknowledgements

We would like to thank Felix and Yossi[2] for sharing experimental details [32]. This work is supported by the National Research Foundation of South Africa (grant no. 120409), a Google Faculty Award, and a Google Africa PhD Fellowship.

8. References

[1] A. Jansen *et al.*, "A summary of the 2012 JHU CLSP workshop on zero resource speech technologies and models of early language acquisition," in *Proc. ICASSP*, 2013.

[2] R. Menon, H. Kamper, E. Van Der Westhuizen, J. Quinn, and T. R. Niesler, "Feature exploration for almost zero-resource ASR-free keyword spotting using a multilingual bottleneck extractor and correspondence autoencoders," in *Proc. Interspeech*, 2019.

[3] O. J. Räsänen, "Computational modeling of phonetic and lexical learning in early language acquisition: Existing models and future directions," *Speech Commun.*, vol. 54, pp. 975–997, 2012.

[4] E. Dupoux, "Cognitive science in the era of artificial intelligence: A roadmap for reverse-engineering the infant language-learner," *Cognition*, vol. 173, pp. 43–59, 2018.

[5] C. Shain and M. Elsner, "Acquiring language from speech by learning to remember and predict," in *Proc. CoNLL*, 2020.

[6] N. Zeghidour, G. Synnaeve, N. Usunier, and E. Dupoux, "Joint learning of speaker and phonetic similarities with Siamese networks," in *Proc. Interspeech*, 2016.

[7] M. Heck, S. Sakti, and S. Nakamura, "Learning supervised feature transformations on zero resources for improved acoustic unit discovery," *IEICE T. Inf. Syst.*, vol. 101, no. 1, pp. 205–214, 2018.

[8] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," in *Proc. Interspeech*, 2019.

[9] P.-J. Last, H. A. Engelbrecht, and H. Kamper, "Unsupervised feature learning for speech using correspondence and siamese networks," *IEEE Signal Proc. Let.*, vol. 27, pp. 421–425, 2020.

[10] R. Algayres, M. S. Zaiem, B. Sagot, and E. Dupoux, "Evaluating the reliability of acoustic speech embeddings," in *Proc. Interspeech*, 2020.

[11] L. Badino, A. Mereta, and L. Rosasco, "Discovering discrete subword units with binarized autoencoders and hidden-Markov-model encoders," in *Proc. Interspeech*, 2015.

[12] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using WaveNet autoencoders," *IEEE Trans. Audio, Speech, Language Process.*, vol. 27, no. 12, pp. 2041–2053, 2019.

[13] E. Dunbar *et al.*, "The Zero Resource Speech Challenge 2019: TTS without T," in *Proc. Interspeech*, 2019.

[14] R. Eloff *et al.*, "Unsupervised acoustic unit discovery for speech synthesis using discrete latent-variable neural networks," in *Proc. Interspeech*, 2019.

[15] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," in *Proc. ICLR*, 2020.

[16] M. Chen and T. Hain, "Unsupervised acoustic unit representation learning for voice conversion using WaveNet auto-encoders," in *Proc. Interspeech*, 2020.

[17] Y.-A. Chung, H. Tang, and J. Glass, "Vector-quantized autoregressive predictive coding," in *Proc. Interspeech*, 2020.

[18] E. Dunbar *et al.*, "The Zero Resource Speech Challenge 2020: Discovering discrete subword and word units," in *Proc. Interspeech*, 2020.

[19] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proc. NeurIPS*, 2017.

[20] C. Coupé, Y. M. Oh, D. Dediu, and F. Pellegrino, "Different languages, similar encoding efficiency: Comparable information rates across the human communicative niche," *Sci. Adv.*, vol. 5, no. 9, 2019.

[21] J. Chorowski *et al.*, "Unsupervised neural segmentation and clustering for unit discovery in sequential data," in *NeurIPS PGR Workshop*, 2019.

[22] B. van Niekerk, L. Nortje, and H. Kamper, "Vector-quantized neural networks for acoustic unit discovery in the zerospeech 2020 challenge," in *Proc. Interspeech*, 2020.

[23] O. Räsänen, "Basic cuts revisited: Temporal segmentation of speech into phone-like units with statistical learning at a prelinguistic level," in *Proc. CogSci*, 2014.

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

[25] M. A. Carlin, S. Thomas, A. Jansen, and H. Hermansky, "Rapid evaluation of speech representations for spoken term discovery," in *Proc. Interspeech*, 2011.

[26] O. Räsänen and M. A. C. Blandón, "Unsupervised discovery of recurring speech patterns using probabilistic adaptive metrics," in *Proc. Interspeech*, 2020.

[27] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[28] G. Hadjeres and L. Crestel, "Vector quantized contrastive predictive coding for template-based music generation," *arXiv preprint arXiv:2004.10120*, 2020.

[29] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[30] 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 Commun.*, vol. 45, no. 1, pp. 89–95, 2005.

[31] O. J. Räsänen, U. K. Laine, and T. Altosaar, "An improved speech segmentation quality measure: the r-value," in *Proc. Interspeech*, 2009.

[32] F. Kreuk, J. Keshet, and Y. Adi, "Self-supervised contrastive learning for unsupervised phoneme segmentation," in *Proc. Interspeech*, 2020.

[33] P. Michel, O. Räsänen, R. Thiolliere, and E. Dupoux, "Blind phoneme segmentation with temporal prediction errors," *arXiv preprint arXiv:1608.00508*, 2016.

[34] Y.-H. Wang, C.-T. Chung, and H.-y. Lee, "Gate activation signal analysis for gated recurrent neural networks and its correlation with phoneme boundaries," in *Proc. Interspeech*, 2017.

[35] J. Franke, M. Mueller, F. Hamlaoui, S. Stueker, and A. Waibel, "Phoneme boundary detection using deep bidirectional LSTMs," in *in Proc. Speech Commun. ITG Symposium*. VDE, 2016, pp. 1–5.

[36] F. Kreuk, Y. Sheena, J. Keshet, and Y. Adi, "Phoneme boundary detection using learnable segmental features," in *Proc. ICASSP*, 2020.

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

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

[39] N. Zeghidour, G. Synnaeve, M. Versteegh, and E. Dupoux, "A deep scattering spectrum-deep Siamese network pipeline for unsupervised acoustic modeling," in *Proc. ICASSP*, 2016.

[40] M. Johnson, T. L. Griffiths, and S. J. Goldwater, "Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models," in *Proc. NIPS*, 2006.

[41] S. J. Goldwater, T. L. Griffiths, and M. Johnson, "A Bayesian framework for word segmentation: Exploring the effects of context," *Cognition*, vol. 112, no. 1, pp. 21–54, 2009.

[42] A. Saksida, A. Langus, and M. Nespor, "Co-occurrence statistics as a language-dependent cue for speech segmentation," *Devel. Sci.*, vol. 20, no. 3, p. e12390, 2017.

[43] H. Kamper, K. Livescu, and S. Goldwater, "An embedded segmental K-means model for unsupervised segmentation and clustering of speech," in *Proc. ASRU*, 2017.

[44] H. Kamper, A. Jansen, and S. Goldwater, "A segmental framework for fully-unsupervised large-vocabulary speech recognition," *Comput. Speech Lang.*, vol. 46, pp. 154–174, 2017.