Interspeech 2018
2-6 September 2018, Hyderabad

Completely Unsupervised Phoneme Recognition by Adversarially Learning Mapping Relationships from Audio Embeddings

Da-Rong Liu, Kuan-Yu Chen, Hung-Yi Lee, Lin-shan Lee

National Taiwan University

{givebirthday, gary840212, tlkagkb93901106}@gmail.com, lslee@gate.sinica.edu.tw

Abstract

Unsupervised discovery of acoustic tokens from audio corpora without annotation and learning vector representations for these tokens have been widely studied. Although these techniques have been shown successful in some applications such as query-by-example Spoken Term Detection (STD), the lack of mapping relationships between these discovered tokens and real phonemes have limited the down-stream applications. This paper represents probably the first attempt towards the goal of completely unsupervised phoneme recognition, or mapping audio signals to phoneme sequences without phoneme-labeled audio data. The basic idea is to cluster the embedded acoustic tokens and learn the mapping between the cluster sequences and the unknown phoneme sequences with a Generative Adversarial Network (GAN). An unsupervised phoneme recognition accuracy of 36% was achieved in the preliminary experiments.

Index Terms: Phoneme Recognition, Unsupervised Learning, Generative Adversarial Network

1. Introduction

With the rapid development of deep learning, remarkable achievements in supervised speech recognition has been obtained [1, 2], but primarily relying on massive annotated data for model training, which is costly and labor requiring. In contrast, in the era of big data, huge quantities of audio corpora are available almost everywhere, but there is almost no way to annotate them or use them in the supervised paradigm. This is why unsupervised approaches of speech processing, including automatically discovering acoustic tokens and learning representations and linguistic structures from unlabeled audio corpora is attractive, desirable and critical [3].

In unsupervised discovery of acoustic tokens, the typical approach is to segment acoustically similar audio signal patterns followed by clustering the obtained repeated patterns in the corpora [4, 5, 6, 7]. These approaches were proved very useful in tasks such as query-by-example spoken term detection (STD) [8, 9, 10, 11].

On the other hand, in representation learning, efforts were made in trying to encode variable-length audio segments into vectors with fixed dimensionality [12]. These vector representations have been shown useful for many applications, such as speaker identification [13] and audio emotion classification [14], in addition to spoken term detection (STD) [15, 16, 17], because such vector representations can be applied to standard classifiers to determine the speaker, the emotion label, or whether the input query is included. For spoken document retrieval, by representing audio segments as vectors, search can be much more efficient compared to template matching over audio frames [12, 18, 19]. Deep learning approaches have been popularly used for such purposes, with LSTM-based sequence-to-sequence auto-encoder being a good example [20, 21]. By minimizing the reconstruction error for the input audio sequences, the embeddings for the audio segments can be extracted from the bottleneck layer of the model. It has been shown that vectors obtained in this way carry the information of phonetic structures for the audio segments [22, 23].

All the above approaches were not able to learn the mapping relationships between the human-defined phonemes and the automatically discovered acoustic tokens. The lack of this mapping relationships have seriously limited the applications obviously because the semantics and core information of the speech signal are carried by the phonemes. Although such automatically discovered acoustic tokens have been successfully transcribed into text with the help of some extra annotations [24], to our knowledge, no work has been reported to try to transcribe the discovered acoustic tokens directly into phonemes in an completely unsupervised way. Motivated by the very impressive work of neural machine translation (NMT) without parallel data [25, 26], with the concept of generative adversarial network (GANs) [27], mapping from these automatically discovered acoustic tokens into phonemes seems possible because it is also a kind of translation except on the acoustic level.

In this paper, we propose a completely unsupervised phoneme recognition framework, in which only unparalleled or unrelated speech utterances and text sentences are needed in model training. The audio signals are automatically segmented into acoustic tokens and encoded into representative vectors. These representative vectors are then clustered, and each speech utterance is represented as a cluster index sequence. A mapping relationship GAN is then developed, in which a generator transforms each cluster index sequence into a predicted phoneme sequence, and a discriminator is trained to distinguish the predicted phoneme sequence from the real phoneme sequences collected from text sentences. A phoneme recognition accuracy of 36% was achieved on TIMIT testing set for a model trained with TIMIT audio training set and unrelated text corpus.

2. Unsupervised phoneme recognition framework

Figure 1 depicts the proposed framework including three parts: Audio2Vec, K-means clustering, and matching relationship GAN. Audio2Vec transforms each audio utterance into a sequence of audio embeddings. Any unsupervised acoustic token discovery approach can be used here, not limited to the ones mentioned below. All the audio embeddings are then K-means clustered and each assigned a cluster index. The mapping relationship GAN then learns to transform the cluster indices into phonemes without supervision.

2.1. Audio2Vec

Each utterance $\mathbf{X}$ is first divided into a sequence of automatically discovered acoustic tokens $\mathbf{X} = (X^1, X^2, ..., X^M)$ in an unsupervised way, where M is the number of tokens in the utterance.

10.21437/Interspeech.2018-1800

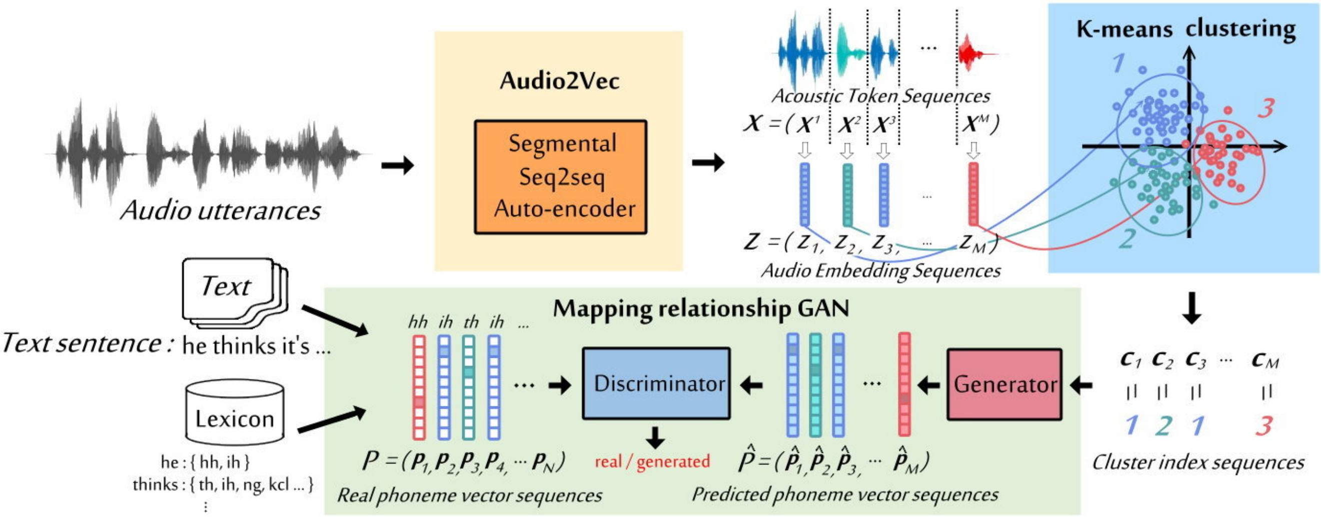
Audio utterances
Audio2Vec
Segmental Seq2seq Auto-encoder
Acoustic Token Sequences
X = (X¹ X² X³ ... X^M)
Z = (z₁, z₂, z₃, ... z_M)
Audio Embedding Sequences
K-means clustering
1
2
3
Text
Text sentence : he thinks it's ...
Lexicon
he : { hh, ih }
thinks : { th, ih, ng, kcl ... }
Mapping relationship GAN
hh ih th ih ...
Discriminator
real / generated
Generator
P = (P₁, P₂, P₃, P₄, ··· P_N)
Real phoneme vector sequences
P̂ = (P̂₁, P̂₂, P̂₃, ··· P̂_M)
Predicted phoneme vector sequences
c₁ c₂ c₃ ··· c_M
1 2 1 3
Cluster index sequences

Figure 1: *The proposed framework consists of three parts: Audio2Vec (yellow) divides the audio utterances into segments and obtains the audio embeddings for the segments; all audio embeddings are then clustered by K-means (blue) and assigned cluster indices; in mapping relationship GAN (green), the generator produces predicted phoneme vector sequences from cluster index sequences, and the discriminator is trained to distinguish the predicted and the real phoneme vector sequences collected from text sentences and the lexicon.*

Each acoustic token X^i is then transformed into an audio embedding $z_i \in \mathbb{R}^d$, where d is the dimensionality of the encoding space. So with Audio2Vec we encode the origin utterance into a vector sequence $Z = (z_1, z_2, ..., z_M)$, where $z_i \in \mathbb{R}^d$ is the audio embedding of the i^{th} token X^i. Many available approaches can be used here for this purpose, but below we assume the recently developed Segmental Audio Word2Vec is used [28]. In this approach, automatic segmentation of utterances into acoustic tokens and audio embedding of the acoustic tokens into vector representations can be jointly learned in a Segmental Sequence-to-sequence Autoencoder (SSAE). In SSAE, a segmentation gate is inserted into the previously developed Sequence-to-sequence Auto-encoder (SAE). The latter maps a variable-length audio segment into a fixed-length vector $z \in \mathbb{R}^d$ [23].

SSAE mentioned above consists of an RNN encoder and an RNN decoder. The RNN encoder reads the input sequence sequentially frame-by-frame and the hidden state of the RNN is updated accordingly. At the last frame of each acoustic token, the hidden state is taken as the audio embedding of the token and the hidden state is reset for the next token. In addition to learning to segment the tokens in an utterance, the RNN encoder and decoder are trained by minimizing the mean squared reconstruction loss. The whole training process requires no labeled data, but the audio embeddings are good representations for the tokens because based on them, the tokens can be reconstructed.

2.2. K-means clustering

In this stage, we collect all audio embeddings $z \in \mathbb{R}^d$ generated from all training audio data, and perform K-means to cluster them into K clusters. The choice of the parameter K will be discussed later on when discussing the experiments. We further map the audio embedding sequence $Z = (z_1, z_2, ..., z_M)$ for each utterance to a cluster index sequence $C = (c_1, c_2, ..., c_M)$, where $c_i = k \in [1, K]$ denotes the index for the cluster z_i belongs to. This clustering process reduces the difficult mapping between a high dimensional continuous embedding space and the phonemes to a simpler mapping between cluster indices and phonemes.

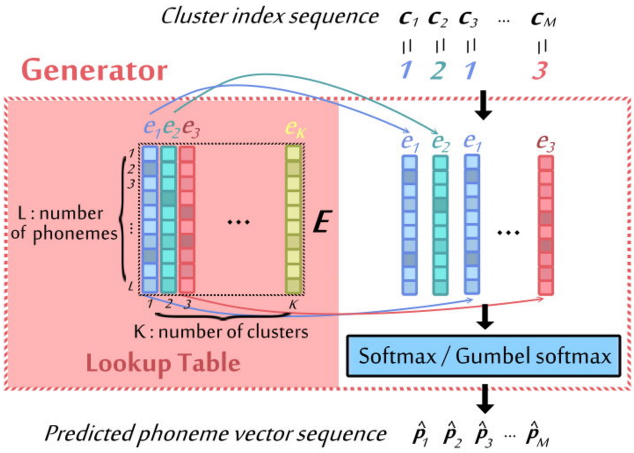

Figure 2: *The generator in the mapping relationship GAN consists of a $K \times L$ matrix E, which is a lookup table, where K is the number of clusters and L the number of phonemes. Given a cluster index $c_i = k \in [1, K]$, we pick up the corresponding column e_k in E by table lookup, and perform softmax or gumbel-softmax to obtain the predicted phoneme vector.*

2.3. Mapping relationship GAN

Generative Adversarial Networks (GANs) [27, 29, 30] is used in this stage. To learn the mapping relationship, we need a collection of text sentences, which does not have to be related to the audio. Each of these text sentence is transformed into phoneme sequence based on a lexicon. Each phoneme is represented as a one-hot encoding vector p. Hence a phoneme sequence for a sentence is represented as a vector sequence $P = (p_1, p_2, ..., p_N)$, where N is the number of phonemes in the sentence. P is referred to as 'real phoneme vector sequence' below.

The mapping relationship GAN consists of a generator and a discriminator. The generator G is an $K \times L$ matrix $E =$

$\{e_1, e_2, ..., e_K\}$, where L is the number of phonemes, K the number of clusters, as in Figure 2. The i^{th} column of E, $e_i \in \mathbb{R}^L$, is intended to approximate the log probability of the i^{th} cluster of tokens over the L phonemes, so E is in fact a lookup table. Given a cluster index $c = k \in [1, K]$, the generator simply picks up the corresponding column of E, e_k, and performs a softmax function to produce a L-dimensional phoneme distribution $\hat{p}$. In this way, for each cluster index sequence C for an utterance, we can generate a sequence of vectors $\hat{P} = G(C) = (\hat{p}_1, \hat{p}_2, ..., \hat{p}_M)$, each vector $\hat{p}_i$ for a token X^i, referred to here as 'predicted phoneme vector sequence'. The discriminator D takes this predicted phoneme vector sequence $\hat{P}$ and outputs a scalar. The higher the scalar, the more possible it is a real phoneme vector sequence.

The discriminator is trained to distinguish between the predicted phoneme vector sequences $\hat{P}$ and the real phoneme vector sequences P, while the generator is trained to cheat the discriminator. The discriminator is trained by minimizing the loss below following the concept of WGAN [29],

$$\mathcal{L}_D = -(\mathbb{E}_{P \sim \mathbb{P}_r}[D(P)] - \mathbb{E}_{\hat{P}=G(C), C \sim \mathbb{P}_C}[D(\hat{P})]), \quad (1)$$

where $\mathbb{P}_r$ is the distribution of the real phoneme vector sequences from text, $\mathbb{P}_C$ the distribution of the cluster index sequences from audio data, and $\hat{P}$ the output of generator, $\hat{P} = G(C)$. Gradient penalty[31] was applied to penalize functions having high gradient norm or changing too rapidly. The generator loss is:

$$\mathcal{L}_G = -\mathbb{E}_{\hat{P}=G(C), C \sim \mathbb{P}_C}[D(\hat{P})]. \quad (2)$$

The generator and the discriminator are learned iteratively. At the end, the generator is supposed to map any cluster index sequence to a phoneme vector sequence which 'looks like' a real one. During testing, given an utterance, the cluster index sequence is generated and transformed into a phoneme vector sequence, and the phoneme with the highest score in each phoneme vector is taken as the phoneme recognition result.

In the generator, given a cluster index c, the predicted phoneme vector was obtained via softmax while the real phoneme vector is one-hot. An alternative way may be sampling from the phoneme distribution, but the sampling process is not differentiable. Therefore, we also try to use gumbel-softmax with straight-through estimator when training the generator [32, 33].

3. Experimental setup

3.1. Audio data

The audio data were from TIMIT acoustic-phonetic corpus [34] including broadband recordings of phonetically-balanced read speech, with 6300 utterances from 630 speakers. The train/test sets were split with 462/168 non-overlapping speakers in each set respectively. Each utterance came with manually time-aligned phonetic and word transcriptions, as well as a 16-bit, 16kHz speech waveform file. The 39-dim MFCCs were extracted with utterance-wise cepstral mean and variance normalization (CMVN) applied. To slightly simplify our work, the start silence and end silence were removed from each utterance according to oracle phoneme boundaries.

3.2. Lexicon and text data

We built a lexicon containing all the words in the text of TIMIT training data. The text sentences were from the English monolingual data of WMT'16 [35], which was extracted from various online news published in 2015, containing roughly 27 millions of sentences. We only used thirty thousands sentences for which all words are in the above lexicon for training.

3.3. Experimental setting

All models were trained with stochastic gradient descent using a mini-batch size of 128, and Adam optimization technique with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. For Sequence-to-sequence Auto-Encoder, one-layer LSTM with 512 hidden units was used for both encoder and decoder. Training was set to 300 epoch with learning rate 5×10^{-4}. The cluster number for K-means ranged from 50 to 1000 and will be discussed below. The discriminator was a two-layer 1D CNN. The first layer concatenated 4 different kernel size: 3,5,7 and 9, each with 256 channels. The second layer was a convolution layer with kernel size 3 and 1024 channels. We used Leaky relu as the activation function. The learning rate of generator and discriminator were set to 0.01 and 0.001 respectively. Every training iteration consisted of 3 discriminator updates and a single generator update. The Gradient penalty ratio of the discriminator was set to 10 [29]. In gumbel-softmax, the inverse temperature was set to 0.9 without annealing [36].

In the preliminary experiments, we didn't use SSAE mentioned above, but took the oracle phoneme boundaries provided in TIMIT for limit of time. This is the only annotaion we used. The rest of the process were competely unsupervised. Automatically segmenting the utterances will be our future work. This gave about 160 thousands and 60 thousands segments in training/testing sets respectively. We trained all the models on the training set, and tested on both training and testing sets. The evaluation metrics is phoneme accurary.

4. Experimental results

4.1. Analysis on number of clusters

We first varied the cluster number K to be 50, 100, 200, 300, 500, 800 and 1000. The results in Figure 3 (A) are those for the reference phoneme sequences from TIMIT training set taken as the real phoneme sequences. So the text sentences are the transcriptions of the audio utterances (or extremely matched), not aligned though. In Figure 3 (B), the phoneme sequences generated from WMT'16 were used as real phoneme sequences, or unrelated to the audio utterances. We obtained the correct phoneme of each segmented token based on the reference transcriptions, from which we defined the majority phoneme of each cluster. The green curves in both Figure 3 (A) and (B) are the cluster purity, which is the percentage of the majority phonemes in all clusters. This is the upper bound of the phoneme accuracy, if all segments in a cluster were mapped to its majority phoneme. Results for both softmax and gumbel softmax are shown for both training and testing sets.

In Figure 3 (A), the performance increased with the cluster number. The best performance (52.49%) was achieved for 1000 clusters, very close to the upper bound or cluster purity of 59.02%. Because the two domains were extremely matched, the mapping relationship GAN could identify the mapping relationship easily, so the performance was primarily determined by the cluster purity, which increased with the cluster number. In Figure 3 (B), there was a much wider gap between the accuracy and the upper bound. For 500 or more clusters (roughly 10 or more clusters for each phoneme in average), the mapping relationship became difficult to learn even though the cluster purity was higher for more clusters. The best performance happened around 300 clusters.

For all cases in Figure 3 (A) and (B), gumbel-softmax outperformed softmax. The real phoneme sequences are all one-hot,

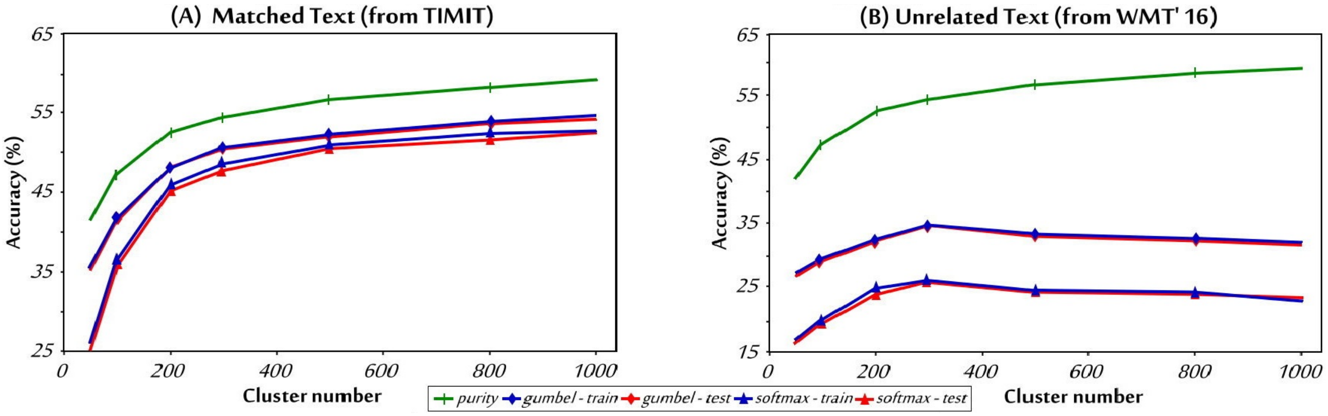

Figure 3: *Phoneme accuracy for different numbers of clusters. Cluster purity (green curves) are the upper bounds.*

Table 1: *Comparison of different methods on training and testing data.*

Methods			Accuracy	
			Train	Test
(a) Random			2.56	2.56
(b) Most Frequent Phoneme			5.12	4.83
proposed	Matched (Figure 3 (A))	(c) Softmax	48.58	48.23
		(d) Gumbel	50.84	50.33
	Unrelated (Figure 3 (B))	(e) Softmax	25.78	25.68
		(f) Gumbel	34.66	34.12
	Unrelated (emsemble)	(g) Softmax	31.35	31.21
		(h) Gumbel	**36.27**	**36.05**

but when using softmax, the generator naturally tried to make the predicted phoneme vectors close to one-hot to cheat the discriminator. But this is weird because a cluster mapped to a specific phoneme only when the cluster purity is 100%. In contrast, with gumbel-softmax, the predicted phoneme vector was to be transformed towards one-hot. So the generator did not need to make the distribution very sharp, thus could concentrate more on making the predicted phoneme vector close to the real distributions. Gumbel-softmax also made the training about 5 times faster.

4.2. Unsupervised phoneme recognition accuracy

The unsupervised phoneme recognition accuracy for 300 clusters in Figure 3 is listed in Table 1. Random baseline (row(a)) guessed uniformly from the 39 phoneme, while the most frequent phoneme baseline (row (b)) simply guessed the most frequent phoneme,/ix/,for all segments. Rows (c) to (f) are the results for the proposed approach for extremely matched data(Figure 3 (A)) and unrelated data(Figure 3 (B)) using softmax and gumbel-softmax It is clear that even with unrelated text the proposed approach significantly outperformed the trivial baselines (rows (e), (f) v.s. (a), (b)). For unrelated data, we also integrated the output of 6 models with slightly different parameters in the generator and discriminator by majority vote for both softmax and gumbel-softmax (rows (g), (h)). Obviously this improved the performance (rows (g) v.s. (e), (h) v.s. (f)), giving an accuracy exceeding 36%. In all cases the training and testing results were very close. For unsupervised learning of phoneme characteristics, whether the model has seen the data is not important.

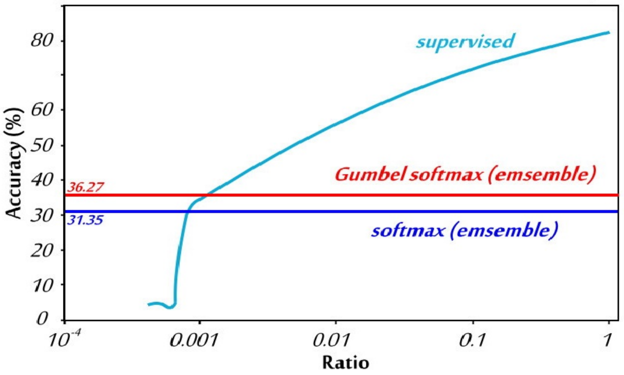

Figure 4: *Comparison of unsupervised learning with unrelated text and supervised learning with different amounts of training data.*

4.3. Comparison with supervised approaches

Here we wish to find out the accuracy we achieved can be obtained with how much of labeled data in supervised apprroach. We used the labeled TIMIT training data to train a recurrent model to map each segment to its phoneme. We used a one-layer LSTM network with 512 hidden units with relu, cross-entropy as loss function. The results are in Figure 4, where the horizontal axis is the labeled data ratio, where 1.0 at the right end means we used exactly the same training data except all of them were labeled, and 0.01 means we only used 1% of the data. Serious over-fitting occured at around 0.0007 of the ratio for the supervise model. We see the proposed unsupervised model with ensemble (the red line) exceeded the supervised model using 0.001 (0.1%) of the training data but labeled. There is still a long way to go.

5. Conclusions and future works

In this work we proposed a framework to achieve compelety unsupervised phoneme recognition without parallel data. We demonstrate that the proposed unsupervised framework can achieve roughly comparable performance on supervised learning with 0.1% of training data. There is still a long way to go along this direction, but here we show it is feasible to recognize phonemes without any labeled data.[1]

[1] Thanks for the sponsorship of Ministry of Science and Technology of Taiwan.

6. References

[1] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in *Advances in neural information processing systems*, 2015, pp. 577–585.

[2] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, K. Gonina *et al.*, “State-of-the-art speech recognition with sequence-to-sequence models,” *arXiv preprint arXiv:1712.01769*, 2017.

[3] H. Kamper, A. Jansen, and S. Goldwater, “A segmental framework for fully-unsupervised large-vocabulary speech recognition,” *Computer Speech & Language*, vol. 46, pp. 154–174, 2017.

[4] J. Glass, “Towards unsupervised speech processing,” in *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, July 2012, pp. 1–4.

[5] A. S. Park and J. R. Glass, “Unsupervised pattern discovery in speech,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 186–197, 2008.

[6] J. Driesen and H. Van hamme, “Fast word acquisition in an NMF-based learning framework,” in *ICASSP*, 2012.

[7] H. Wang, C.-C. Leung, T. Lee, B. Ma, and H. Li, “An acoustic segment modeling approach to query-by-example spoken term detection,” in *ICASSP*, 2012.

[8] V. Lyzinski, G. Sell, and A. Jansen, “An evaluation of graph clustering methods for unsupervised term discovery,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[9] Y. Zhang, R. Salakhutdinov, H.-A. Chang, and J. Glass, “Resource configurable spoken query detection using deep boltzmann machines,” in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 5161–5164.

[10] C.-T. Chung and L.-S. Lee, “Unsupervised discovery of structured acoustic tokens with applications to spoken term detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 394–405, 2018.

[11] C.-T. Chung, C.-Y. Tsai, C.-H. Liu, and L.-S. Lee, “Unsupervised iterative deep learning of speech features and acoustic tokens with applications to spoken term detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 10, pp. 1914–1928, 2017.

[12] K. Levin, K. Henry, A. Jansen, and K. Livescu, “Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings,” in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 410–415.

[13] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, “Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification,” in *Tenth Annual conference of the international speech communication association*, 2009.

[14] B. Schuller, S. Steidl, and A. Batliner, “The interspeech 2009 emotion challenge,” in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[15] H.-y. Lee and L.-s. Lee, “Enhanced spoken term detection using support vector machines and weighted pseudo examples,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 6, pp. 1272–1284, 2013.

[16] I.-F. Chen and C.-H. Lee, “A hybrid hmm/dnn approach to keyword spotting of short words.” in *INTERSPEECH*, 2013, pp. 1574–1578.

[17] A. Norouzian, A. Jansen, R. C. Rose, and S. Thomas, “Exploiting discriminative point process models for spoken term detection,” in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[18] K. Levin, A. Jansen, and B. Van Durme, “Segmental acoustic indexing for zero resource keyword search,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5828–5832.

[19] H. Kamper, W. Wang, and K. Livescu, “Deep convolutional acoustic word embeddings using word-pair side information,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4950–4954.

[20] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: http://www.aclweb.org/anthology/D14-1179

[21] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[22] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, and L.-S. Lee, “Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder,” *arXiv preprint arXiv:1603.00982*, 2016.

[23] C.-H. Shen, J. Y. Sung, and H.-Y. Lee, “Language transfer of audio word2vec: Learning audio segment representations without target language data,” *arXiv preprint arXiv:1707.06519*, 2017.

[24] S. Bansal, H. Kamper, A. Lopez, and S. Goldwater, “Towards speech-to-text translation without speech recognition,” *arXiv preprint arXiv:1702.03856*, 2017.

[25] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou, “Word translation without parallel data,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[26] G. Lample, L. Denoyer, and M. Ranzato, “Unsupervised machine translation using monolingual corpora only,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[28] Y.-H. Wang, H.-y. Lee, and L.-s. Lee, “Segmental audio word2vec: Representing utterances as sequences of vectors with applications in spoken term detection,” in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference*. IEEE, 2018.

[29] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” *arXiv preprint arXiv:1701.07875*, 2017.

[30] L. Yu, W. Zhang, J. Wang, and Y. Yu, “Seqgan: Sequence generative adversarial nets with policy gradient.” 2017.

[31] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5767–5777. [Online]. Available: http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf

[32] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax,” *arXiv preprint arXiv:1611.01144*, 2016.

[33] C. J. Maddison, A. Mnih, and Y. W. Teh, “The concrete distribution: A continuous relaxation of discrete random variables,” *arXiv preprint arXiv:1611.00712*, 2016.

[34] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, “Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1,” *NASA STI/Recon technical report n*, vol. 93, 1993.

[35] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, V. Logacheva, C. Monz *et al.*, “Findings of the 2016 conference on machine translation.” in *ACL 2016 FIRST CONFERENCE ON MACHINE TRANSLATION (WMT16)*. The Association for Computational Linguistics, 2016, pp. 131–198.

[36] C. J. Maddison, D. Tarlow, and T. Minka, “A* sampling,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3086–3094.