Audio Word2vec: Sequence-to-Sequence Autoencoding for Unsupervised Learning of Audio Segmentation and Representation

Yi-Chen Chen, Sung-Feng Huang, Hung-yi Lee, Yu-Hsuan Wang, and Chia-Hao Shen

***Abstract*—In text, word2vec transforms each word into a fixed-size vector used as the basic component in applications of natural language processing. Given a large collection of unannotated audio, audio word2vec can also be trained in an unsupervised way using a sequence-to-sequence autoencoder (SA). These vector representations are shown to effectively describe the sequential phonetic structures of the audio segments. In this paper, we further extend this research in the following two directions. First, we disentangle phonetic information and speaker information from the SA vector representations. Second, we extend audio word2vec from the word level to the utterance level by proposing a new segmental audio word2vec in which unsupervised spoken word boundary segmentation and audio word2vec are jointly learned and mutually enhanced, and utterances are directly represented as sequences of vectors carrying phonetic information. This is achieved by means of a segmental sequence-to-sequence autoencoder, in which a segmentation gate trained with reinforcement learning is inserted in the encoder.**

***Index Terms*—Audio word2Vec, sequence-to-sequence autoencoder.**

I. Introduction

HUMAN infants acquire languages with little formal teaching; machines, however, must learn from a large amount of annotated data, which makes the development of speech technology for a new language challenging. For typical spoken language understanding, one can simply convert spoken content into word sequences using an off-the-shelf speech recognizer. However, to train a high-quality speech recognition system, huge quantities of annotated audio data are needed. Therefore, for low-resource languages with scarce annotated data, or languages without written forms, sufficiently accurate speech recognition is difficult to achieve. Some previous work [1], [2] focus on speech recognition with mismatched crowdsourcing and probabilistic transcriptions.

Annotating audio data for speech recognition is expensive, but unannotated audio data is relatively easy to collect. If the machine can acquire the word patterns behind speech signals from a large collection of unannotated speech data without speech recognition, it would be able to learn a new language in a novel linguistic environment with little supervision. Imagine a Hokkien-speaking family buying an intelligent device: although at first the machine does not understand Hokkien, by hearing people speak it, it automatically learns the language. This paper is one step toward this dream [3]–[5].

In text, word2vec [6]–[8] transforms each word into a fixed-dimension vector used as the basic component of applications of natural language processing. Word2vec is useful because it is learned from a large collection of documents without supervision. It is therefore interesting to ask: Given a large collection of unannotated audio, can the machine automatically represent an utterance as a sequence of vector representations, each of which corresponds to a word? The vector representations should describe the sequential phonetic structures of the audio signals. In some papers, such vector representations are called "acoustic embedding". Thus vector representations for words that sound alike should be located in close proximity to each other in the vector space, regardless of the unique characteristics of the speakers uttering the segments. If this can be achieved, this kind of representation would serve as the basic component of downstream spoken language understanding applications, similar to Word2Vec in natural language processing. For example, consider translating the speech of a low-resource language into the text of another language [9]. With training data in the form of audio paired with text translations, representing the audio by audio Word2Vec would make training more efficient than if using acoustic features like MFCCs.

In this paper, a sequence-to-sequence autoencoder (SA) is used to represent variable-length audio segments using fixed-length vectors [10], [11]. With SA, only audio segments without human annotation are needed, which suits it for low-resource applications. Although autoencoding is a successful machine learning technique for extracting representations in an unsupervised way [12], [13], it requires fixed-length input vectors. This is a considerable limitation because audio segments are intrinsically expressed as sequences of arbitrary length. SA, proposed to encode sequences based on sequence-to-sequence learning,

Manuscript received December 4, 2018; revised May 5, 2019; accepted May 30, 2019. Date of publication June 13, 2019; date of current version June 28, 2019. This work was supported by the Ministry of Science and Technology of Taiwan. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Hsin-min Wang. *(Corresponding author: Hung-yi Lee.)*

Y.-C. Chen and S.-F. Huang are with the Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: b02901074@ntu.edu.tw; r06942045@ntu.edu.tw).

H.-yi Lee is with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: hungyilee@ntu.edu.tw).

Y.-H. Wang is with the Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: r04922167@ntu.edu.tw).

C.-H. Shen is with the CompStak, New York, NY 10003 USA (e-mail: r04921047@ntu.edu.tw).

Digital Object Identifier 10.1109/TASLP.2019.2922832

has been applied in natural language processing [14], [15] and video processing [16]. SA consists of an RNN encoder and decoder. The RNN encoder reads an audio segment represented as an acoustic feature sequence and maps it to a vector representation with a fixed length of z; the RNN decoder maps the vector z to another sequence. The RNN encoder and decoder are trained to minimize the reconstruction error of the input acoustic sequence. It has been shown that the representation z contains phonetic information [10], [11]. For example, the vector for "night" subtracted by the vector for "fight" is approximately equal to the vector for "name" subtracted by the vector for "fame" [10].

In this paper, we further improve on the SA vector representation. Because the representation z extracted by the RNN encoder must reconstruct the input signals, it includes not only phonetic information but also speaker, environment, and channel information in various dimensions. That is, audio segments corresponding to the same terms with the same phonetic structure may have different z if produced by different speakers. Therefore, it is necessary to disentangle the phonetic information in z from other information, so the phonetic information can be further used in downstream applications. Drawing from adversarial training [17], we use a classifier. The classifier learns to distinguish whether two segments were uttered by the same speaker or not based on the representations. To confuse the classifier, the encoder learns to extract speaker-invariant representations.

Word segmentation of speech is critical but challenging for zero-resource speech technology because word boundaries are usually not available for given speech utterances or corpora [3], [18]–[20]. Although there are approaches to estimating word boundaries [21]–[25], we hypothesize that the audio segmentation and SA can be integrated and jointly learned, so that they can enhance each other. This means that the machine learns to segment the utterances into a sequence of spoken words while at the same time transforming these spoken words into a sequence of vectors. We propose the segmental sequence-to-sequence autoencoder (SSAE) [26], a new model to jointly train the segmenter while extracting the representation. The SSAE contains a segmentation gate jointly learned with SA from an unlabeled corpus in a completely unsupervised way. During training, the SSAE learns to convert the utterances into sequences of embeddings, and then reconstructs the utterances with these embedding sequences. The only thing needed during training is a guideline for the proper number of vectors (or words) within an utterance of a given length, to ensure that the machine segments the utterances into word-level segments. Since the model is not completely differentiable, standard backpropagation is not applicable [27], [28]; thus we use reinforcement learning to train SSAE.

In this paper, we employ query-by-example spoken term detection (QbE STD), a real-world application, to evaluate the phonetic structure information of the original utterances contained in these generated word vector sequences. When based on audio word2vec, QbE STD is much more efficient than conventional dynamic time warping (DTW) based approaches, because only the similarities between two single vectors are needed; this is in addition to the significantly better retrieval performance that it yields.

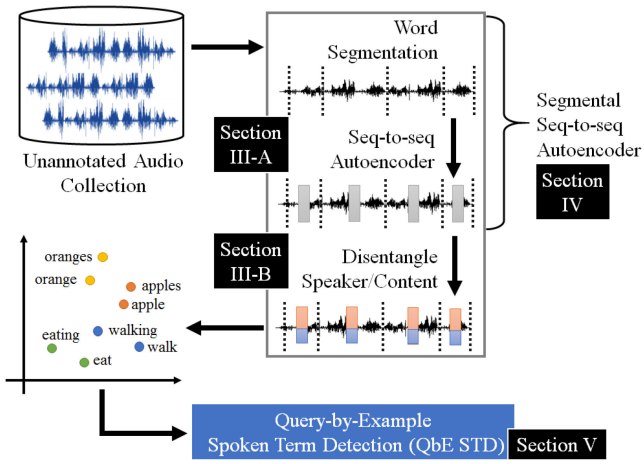

Fig. 1. Audio word2vec framework.

The audio word2vec framework is summarized in Fig. 1. Given a large collection of annotated audio, it is first segmented into word-level segments. Then the SA model generates an embedding for each audio segment, as described in Section III-A. In Section III-B, we describe how to disentangle speaker and speech content from the embedding. In Section IV, we describe the proposed SSAE model, which jointly learns segmentation and embedding. In Section V the embedding is evaluated on QbE STD.

II. Related Work

Audio segment representation is still an open problem. It is common to use i-vectors to represent utterances in speaker identification [29]. However, i-vectors are not designed to precisely describe the sequential phonetic structure of audio segments as desired for our task. Embedding audio word segments into fixed-length vectors also has useful applications in spoken term detection (STD) [30]–[35], in which audio segments are usually represented as feature vectors to be applied to a standard classifier which determines whether the input queries are included [32]–[34]. In previous work, embedding approaches were developed primarily in heuristic ways, rather than learned from data. Graph-based embedding approaches are also used to represent audio segments as fixed-length vectors [30], [31]. Retrieval task efficiency is improved by searching audio content using fixed-length vectors instead of using the original acoustic features [30], [31].

Recently, deep learning has been used to encode acoustic information as vectors [36]–[44]. This transformation successfully produces vector spaces in which word audio segments with similar phonetic structures are located in close proximity. By training a recurrent neural network (RNN) with an audio segment as the input and the corresponding word as the target, the outputs of the hidden layer at the last few time steps can be taken as the representation of the input segment [37], [45]. However, this approach is supervised and therefore necessitates a large amount of

labeled training data. In [38], the authors train a neural network with side information to obtain embeddings that separate same-word and different-word pairs. Since human annotated data is still required, the scenario is weakly supervised. For non-speech audio, some approaches obtain labeled paired data based on the nature of signals [46], but the approaches have not yet been applied to speech.

Feature disentanglement of audio features has been studied with variational autoencoders [47]–[49]. In contrast to previous work, the feature disentanglement approach here uses a classifier as discriminator. This approach is borrowed from work on domain adversarial training [17], [50]. Similar ideas have been applied to domain adaptation for speech recognition [51]; while senone labels are needed in the previous work, here the feature disentanglement is completely unsupervised.

In unsupervised pattern discovery, segmentation followed by clustering is typical [52]–[63].[1] Probabilistic Bayesian models are developed to construct a model which learns segmentation and representation (or clustering) jointly [21], [25], [64]. Although Bayesian models yield successful results, they do not scale well to large speech corpora; as such the embedded segmental K-means model has been proposed as an approximation of the Bayesian model [65] – this however does not take advantage of deep learning. Another approach to learn segmentation is using an autoencoder with a sample-based algorithm [66]. First an LSTM is used to model a proposal distribution over sequences of segment boundaries for each utterance. Then m sequences of boundaries from the distribution are sampled to split the utterance into words and the reconstruction loss for each sequence is obtained with an autoencoder. The losses and the distribution are used to compute an importance weight for each sample and breakpoint. A breakpoint is more likely if it appeared in samples with low reconstruction loss. In comparison, SSAE as proposed in this paper is a "whole network model" using reinforcement learning, which scales well and can be trained in an end-to-end fashion.

This journal paper is an extension of previous conference papers. Both SA [10] and SSAE [26] have been proposed in previous conference papers; SA's language transfer ability has also been verified in a conference paper [11]. However, SA and SSAE have not yet been used with feature disentanglement based on adversarial training.

III. Audio Representation

The goal for the audio word2vec model is to identify the phonetic patterns in sequences of acoustic features such as MFCCs. Given a word-level audio segment $\mathbf{x} = (x_1, x_2, ..., x_T)$ where x_t is the acoustic feature at time t, and T is the length, audio word2vec transforms the features into a fixed-length vector $\mathbf{z} \in \mathbb{R}^d$ with dimension d. In this section, we assume the word boundaries are ready available. In the next section we describe how to jointly learn segmentation and representation.

[1]In most approaches, models learn to cluster audio segments instead of producing distributed representations as in this paper; however we can consider clustering as representing audio segments with a one-hot encoding.

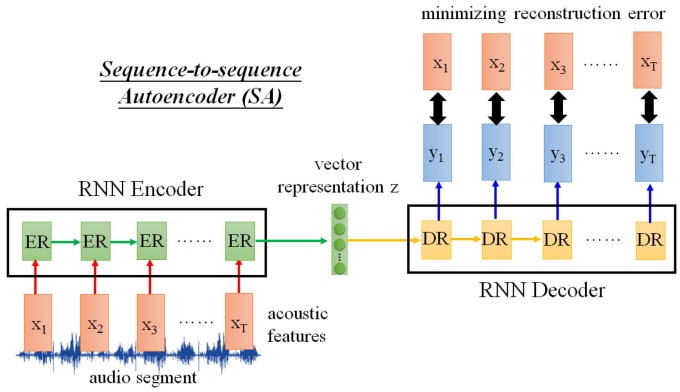

Fig. 2. Sequence-to-sequence autoencoder (SA), consisting an RNN encoder (ER) and an RNN decoder (DR). The encoder reads an audio segment represented as an acoustic feature sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$ and maps it to a fixed-length vector representation with dimension $\mathbf{z}$; the decoder maps the vector z to another sequence $\mathbf{y} = (y_1, y_2, ..., y_T)$. The RNN encoder and decoder are jointly trained such that the output sequence $\mathbf{y}$ is as close to the input sequence $\mathbf{x}$ as possible.

A. Sequence-to-Sequence Autoencoder

Recurrent neural networks (RNNs) have shown great success in many NLP tasks with their ability to capture sequential information. The hidden neurons form a directed cycle and perform the same task for every element in a sequence. Given a sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$, the RNN updates its hidden state h_t according to the current input x_t and the previous h_{t-1}. The hidden state h_t acts as an internal memory at time t that enables the network to capture dynamic temporal information, and also allows the network to process sequences of variable length.

The RNN encoder-decoder architecture [67], [68] consists of an RNN encoder and an RNN decoder. The encoder reads the input sequence $\mathbf{x} = (x_1, x_2, ..., x_{T_1})$ sequentially, and the hidden state h_t of the RNN is updated accordingly. After the last symbol x_{T_1} is processed, the hidden state h_{T_1} is interpreted as the learned representation of the whole input sequence. Then, taking h_{T_1} as input, the RNN decoder generates the output sequence $\mathbf{y} = (y_1, y_2, ..., y_{T_2})$ sequentially, where T_1 and T_2 can be different, or the length of $\mathbf{x}$ and $\mathbf{y}$ can be different. This RNN encoder-decoder framework is able to handle variable-length input and output.

Fig. 2 depicts the structure of the sequence-to-sequence autoencoder (SA), which integrates the RNN encoder-decoder framework with an autoencoder for the unsupervised learning of audio segment representations. The SA consists of an RNN encoder (the left part of Fig. 2) and decoder (the right part). Given an audio segment represented as an acoustic feature sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$ of any length T, the RNN encoder reads each acoustic feature x_t sequentially and the hidden state h_t is updated accordingly. After the last acoustic feature x_T has been read and processed, the hidden state h_T of the encoder is taken as the learned representation z of the input sequence (the vector in the middle of Fig. 2).

The RNN decoder takes h_T as the initial state, and generates a sequence $\mathbf{y}$. Based on autoencoder principles [12], [13], the target of the output sequence $\mathbf{y} = (y_1, y_2, ..., y_T)$ is the input

sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$. In other words, the RNN encoder and decoder are jointly trained by minimizing the reconstruction error $\mathcal{L}_{mse}$,

$$\mathcal{L}_{mse} = \sum_{\mathbf{x}} \sum_{t=1}^{T_{\mathbf{x}}} \|x_t - y_t\|^2, \tag{1}$$

where $\mathcal{L}_{mse}$ is the sum over all the audio segments $\mathbf{x}$ in the data collection, and $T_{\mathbf{x}}$ represents the length of the segment $\mathbf{x}$. Because the input sequence is taken as the learning target, the training process requires no labeled data. The fixed-length vector representation z is thus a meaningful representation for the input audio segment $\mathbf{x}$ because the whole input sequence $\mathbf{x}$ can be reconstructed from z with the RNN decoder. Although in Fig. 2 both the RNN encoder and decoder have only one hidden layer, this does not preclude the use of multiple layers.

In Fig. 2, after generating output y_1, instead of taking y_1 as the input of the next time step, a zero vector is used as input to generate y_2, and so on, in contrast to the typical encoder-decoder architecture. This use of historyless decoding is critical here. We found that if a typical decoder is used (that is, RNN takes y_1 as input to generate y_2, and so on), despite the resultant low reconstruction error, the SA-learned vector representations do not include useful information. This is because a strong decoder focuses less on including more information in the vector representation. Historyless decoding yields a weakened decoder because the input of the decoder is removed, which forces the model to rely more on the vector representation. Historyless decoding is also used in some NLP applications [69]–[71].

B. Feature Disentanglement

Because the representation z extracted by the RNN encoder in Fig. 2 must reconstruct the input signals, it includes not only phonetic information but also speaker, environment, and channel information in various dimensions. Therefore, it is necessary to disentangle the phonetic information from other information.

As shown in Fig. 3(A), the basic idea of disentanglement is to add an additional speaker encoder. The original encoder in Fig. 2 is the phonetic encoder, which takes the audio segment $\mathbf{x}$ as input and outputs embedding vector z. The speaker encoder has the same RNN architecture as the phonetic encoder. Its output embedding is denoted as e. The input of the RNN decoder is the concatenation of vectors z and e. The phonetic encoder, speaker encoder, and RNN decoder are jointly learned to minimize the reconstruction error $\mathcal{L}_{mse}$ in (1).

Ensuring that z contains phonetic information and e contains speaker information is not as simple as merely minimizing $\mathcal{L}_{mse}$; rather, to achieve this goal, we require additional training criteria for speaker encoder, as shown in Fig. 3(B). The speaker encoder learns to minimize the distance between the e of audio segments uttered by the same speaker [72], and enlarge the distance between the e of different speakers past a threshold. Given audio segments $\mathbf{x}^i$ and $\mathbf{x}^j$, we obtain two embeddings e^i and e^j from the speaker encoder.[2] If $\mathbf{x}^i$ and $\mathbf{x}^j$ are from the same speaker,

[2]We use superscripts to represent the indices of whole audio segments, and subscripts for acoustic features within an audio segment.

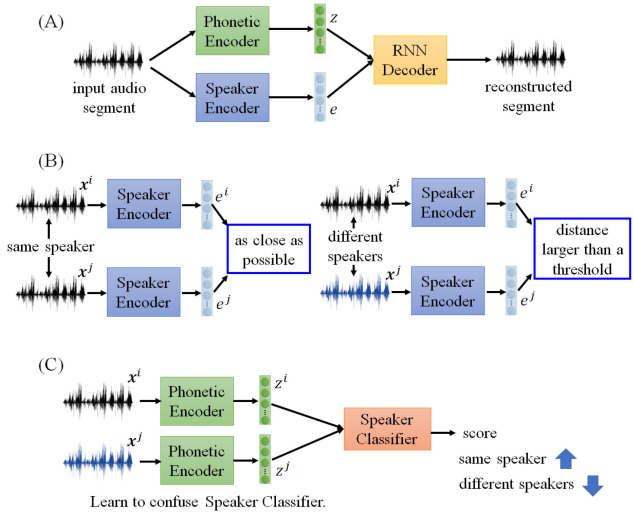

Fig. 3. Feature disentanglement. (A) Adding speaker encoder for reconstruction. (B) Additional training criteria for the speaker encoder. (C) Speaker classifier learns to distinguish whether two z's are from the same speaker or not; phonetic encoder attempts to confuse the classifier with z.

the speaker encoder learns to minimize $\|e^i - e^j\|^2$. If $\mathbf{x}^i$ and $\mathbf{x}^j$ are from the different speakers, however, the speaker encoder learns to minimize $max(\lambda_d - \|e^i - e^j\|^2, 0)$, such that the distance between e^i and e^j is larger than a predefined threshold λ_d. This assumes that speaker labels are available. If speaker information is not available, the speaker encoder can still be learned by assuming that segments from the same utterance are produced by the same speaker. Although we only consider the speaker information here, it is possible to use the same approach to consider other information such as the channel.

Note that the above constraint on the speaker encoder is not sufficient because it does not prevent the phonetic encoder from putting the speaker information in z. To address this problem, we borrow the speaker classifier technique from adversarial training [17], as shown in Fig. 3(C). The inputs of the speaker classifier are the z^i and z^j from two audio segments $\mathbf{x}^i$ and $\mathbf{x}^j$; the classifier produces a score representing whether z^i and z^j are from the same speaker. The speaker classifier loss is the same as that for the Wasserstein generative adversarial network (WGAN) [73]. This loss is defined as the difference between the summation of the scores of the pairs from the same speaker minus that from different speakers. To minimize this loss, the classifier increases scores for pairs from the same speaker and decreases scores for those from different speakers. Gradient penalty is used to learn the classifier [74]. Note that the learning targets of the speaker classifier and the phonetic encoder are opposites: the phonetic encoder updates its parameters to maximize the loss of the speaker classifier, while the speaker classifier does its best to distill the speaker information from z in order to determine whether z^i and z^j are from the same speaker. Thus, the phonetic encoder tries its utmost to generate z vectors that confuse the speaker classifier. If it successfully achieves this after training, it produces a z that contains no speaker information, and an e that contains all the speaker information. The complete procedure for feature disentanglement is shown in Algorithm 1.

Algorithm 1: Feature Disentanglement.

Input: Audio segment collection $\mathcal{X}$, total update iterations T_{train}, Speaker classifier update iterations T_{dis}
Output: Phonetic encoder parameters θ_p

Initialize phonetic encoder parameters θ_p
Initialize speaker encoder parameters θ_s
Initialize RNN decoder parameters θ_d
Initialize speaker classifier parameters θ_c
for $t = 1$ to T_{train} **do**
  Sample M audio segments $\mathcal{X}_r$ from $\mathcal{X}$
  Sample M audio segment pairs $\mathcal{X}_p$ from $\mathcal{X}$ in which the paired segments are from the same speakers
  Sample M audio segment pairs $\mathcal{X}_n$ from $\mathcal{X}$ in which the paired segments are from different speakers
  % Train phonetic classifier
  Compute phonetic classifier loss $\mathcal{L}_c$ based on the $2M$ segment pairs $\{\mathcal{X}_p, \mathcal{X}_n\}$

$$\mathcal{L}_c = \sum_{(\mathbf{x}^i, \mathbf{x}^j) \in \mathcal{X}_p} C(z^i, z^j) - \sum_{(\mathbf{x}^i, \mathbf{x}^j) \in \mathcal{X}_n} C(z^i, z^j), \tag{2}$$

  where $C(z^i, z^j)$ is the speaker classifier output score given the phonetic embedding of the segment pair $(\mathbf{x}^i, \mathbf{x}^j)$ with parameters θ_c
  % Phonetic classifier parameters updated with additional iterations as in typical GAN framework
  for $t' = 1$ to T_{dis} **do**
    $\theta_c \leftarrow \theta_c - \eta \bigtriangledown \mathcal{L}_c + gradient\ penalty$
  end for
  % Minimize reconstruction error
  Compute reconstruction loss $\mathcal{L}_{mse}$ in (1) over segments in $\mathcal{X}_r$
  % Extra training criteria for speaker encoder
  Compute loss $\mathcal{L}_s$

$$\mathcal{L}_s = \sum_{\mathbf{x}^i, \mathbf{x}^j \in \mathcal{X}_p} \|e^i - e^j\|^2 + \sum_{\mathbf{x}^i, \mathbf{x}^j \in \mathcal{X}_n} max(\lambda_d - \|e^i - e^j\|^2, 0), \tag{3}$$

  where e^i and e^j are output of speaker encoder given $\mathbf{x}^i$ and $\mathbf{x}^j$.
  % Compute the total loss $\mathcal{L}_{tot}$ for updating the whole model

$$\mathcal{L}_{tot} = \mathcal{L}_{mse} + \mathcal{L}_s - \mathcal{L}_c \tag{4}$$

  $\theta_p \leftarrow \theta_p - \eta \bigtriangledown \mathcal{L}_{tot}$
  $\theta_s \leftarrow \theta_s - \eta \bigtriangledown \mathcal{L}_{tot}$
  $\theta_d \leftarrow \theta_d - \eta \bigtriangledown \mathcal{L}_{tot}$
end for

IV. Jointly Learning Segmentation and Representation

A. Segmental Sequence-to-Sequence Autoencoder (SSAE)

The proposed structure for SSAE is depicted in Fig. 4, in which the segmentation gate is inserted into the SA. The input of SSAE is an utterance $\mathbf{x}' = \{x_1, x_2, ..., x_{T'}\}$, where $\mathbf{x}_t$ represents the t-th acoustic feature, and T' is the length of the utterance. In general, an utterance $\mathbf{x}'$ is much longer than an audio segment x with length T in Section III, that is, $T' \gg T$. Given an input utterance, the model learns to determine the word boundaries in the utterance, and generates N audio segments. The model then produces the embeddings for the N generated audio segments, $\mathbf{z} = \{z_1, z_2, ..., z_N\}$, where z_n is the n-th embedding and $N \leq T'$. As with the conventional autoencoder, the proposed SSAE consists of an RNN encoder and an RNN decoder. The encoder includes an extra segmentation gate, controlled by another RNN (shown in Fig. 4 as the sequence of blocks labeled **S**). The segmentation gate can be considered an "agent" in the parlance of typical reinforcement learning. At each time t, the segmentation gate agent performs an action a_t, according to a given input feature s_t (or *state* per reinforcement learning). The action a_t is either "segment" or "pass". If "segment", then $\mathbf{x}_t$ is regarded as a word boundary.

For the segmentation gate, the input state at time t, s_t, is defined as the concatenation of the input acoustic feature x_t, the gate activation signal (GAS) g_t, and the previous taken action a_{t-1},

$$s_t = \left[x_t || g_t || a_{t-1}\right]. \tag{5}$$

The GAS feature g_t is an unsupervised feature for segmentation. This feature is extracted from the values of the update gates of the GRU of another pre-trained RNN autoencoder, which is trained simply to minimize the reconstruction loss. We use GAS as extra input formation here because it has been verified that the temporal structure of such signals is correlated with the phoneme boundaries [22]. Here the same set of audio data is used to train the SSAE and learn GAS features. The policy π_t at time t is modeled by the output h_t of the layers of the segmentation gate RNN (**S** blocks in Fig. 4) followed by a linear transform (W^π, b^π) and a softmax nonlinearity:

$$h_t = RNN(s_1, s_2, ..., s_t), \tag{6}$$

$$\pi_t = softmax(W^\pi h_t + b^\pi). \tag{7}$$

This π_t gives two probabilities respectively for "segment" and "pass". An action a_t is then sampled from this distribution during training to encourage exploration. During testing, the action with the highest probability is taken.

When a_t is "segment", the time t is viewed as a word boundary, after which the segmentation gate passes the output of the RNN encoder. If this is the n-th time the action "segment" is taken, and last time action "segment" is taken at the t'-th time step, yielding the n-th audio segment, when t' and t refer to the beginning and ending time for the n-th audio segment. The output of the RNN encoder at time step t is the vector representation of the n-th audio segment z_n:

$$z_n = Encoder(x_{t'}, x_{t'+1}, ..., x_t). \tag{8}$$

After "segment" is taken, the internal state of the RNN encoder is reset to its initial value, so z_n is generated based only on the acoustic features of the n-th audio segment.

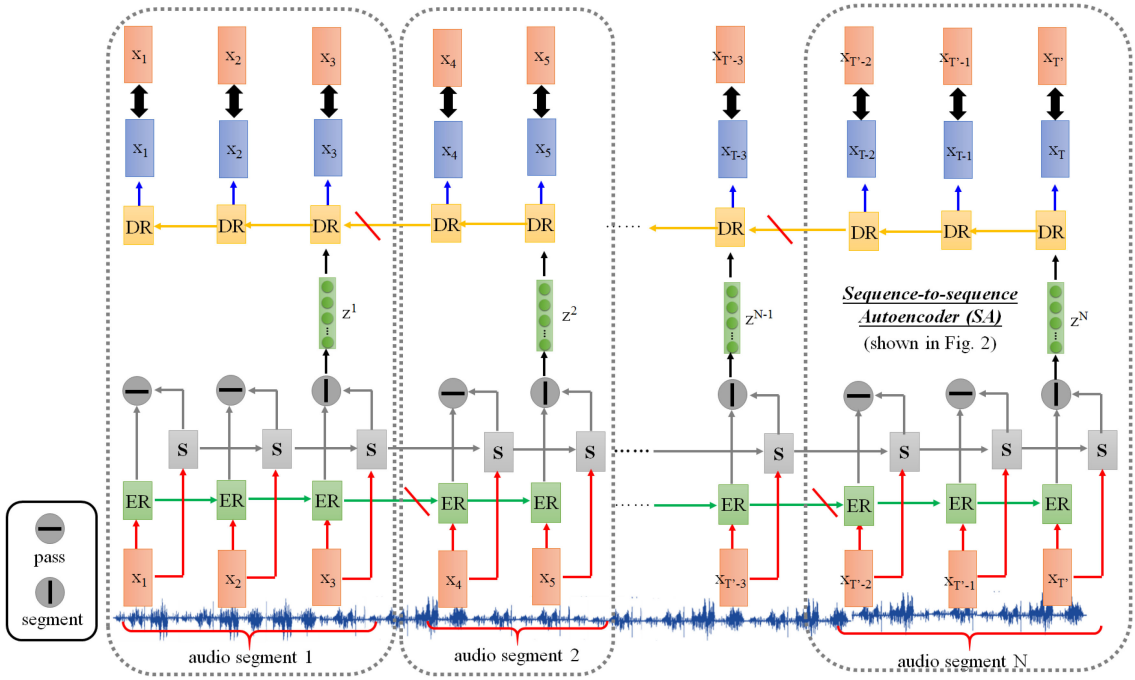

Fig. 4. Segmental sequence-to-sequence autoencoder (SSAE). In addition to the RNN encoder (ER blocks) and RNN decoder (DR blocks), a segmentation gate (**S** blocks) is included in the model to estimate word boundaries. During transitions across segment boundaries, the RNN encoder and decoder are reset (illustrated as a slashed arrow) to prevent information flow across segment boundaries. Each segment (the boxes with dotted lines) can be viewed as performing the sequence-to-sequence training from Fig. 2 individually.

Then the input utterance $\mathbf{x}$ is reconstructed with the embedding sequence $\mathbf{z} = \{z_1, z_2, ..., z_N\}$. Given an embedding z_n for the n-th input segment in (8) above, the RNN decoder generates $\{y_{t'}, y_{t'+1}, ..., y_t\}$ to reconstruct the input acoustic features $\{x_{t'}, x_{t'+1}, ..., x_t\}$. When the RNN decoder begins decoding each audio segment, its state is also reset.

In Fig. 4, each audio segment in the boxes with dotted lines can be viewed as performing the sequence-to-sequence training from Fig. 2 individually. Note that the sequence-to-sequence training in Fig. 4 reconstructs the target sequence in reverse order, in contrast to that shown in Fig. 2. In preliminary experiments, we found that the order of reconstruction does not significantly influence the performance of the representation. We use the reverse order here because it can be implemented more efficiently.

B. SSAE Training

Although all the parameters in the SSAE model can be trained simultaneously, we actually train our model using an iterative process consisting of two phases. In the first phase, the RNN encoder and decoder parameters are updated, while in the second phase, only the segmentation gate parameters are updated. The two phases are performed iteratively.

1) First Phase - RNN Encoder and Decoder: In the first phase, we train only the RNN encoder and decoder to minimize reconstruction error while fixing the parameters of the segmentation gate. Because the segments are already provided by the segmentation gate, the first phase training is parallel to training a typical SA as in Section III-A. That is, the encoder and decoder learn to minimize the reconstruction error $\mathcal{L}_{mse}$ in (1). Each time in phase one, the encoder and decoder are learned from random initialized parameters, instead of starting off with the parameters learned in the previous iteration – this was found to offer better training stability.

2) Second Phase – Segmentation Gate: In the second phase, we update the parameters of the segmentation gate while fixing the parameters of the encoder and decoder. Although the encoder and decoder are not updated in this phase, they are involved in computing the reward for training the segmentation gates by reinforcement learning.

The segmentation gate is trained using reinforcement learning. After the gate performs the segmentation for each utterance, it receives a reward r and a reward baseline r_b for updating the parameters. r and r_b are defined later. We can express the expected reward for the gate under policy π as $J(\theta) = \mathbf{E}_{\pi}[r]$, where θ is the parameter set. To maximize the expected reward $J(\theta)$, policy gradient [75] is used to update the parameters of the segmentation gate using the parameter update formulation below.

$$\nabla_{\theta} J(\theta) = \mathbf{E}_{a\sim\pi}\left[\nabla_{\theta}(r - r_b)\sum_{t=1}^{T'} log\pi_t^{(\theta)}(a_t)\right], \qquad (9)$$

where $\pi_t^{(\theta)}(a_t)$ is the probability for the action a_t taken per (7).

The reconstruction error is an effective indicator of whether the segmentation boundaries are good, since the embeddings are generated based on the segmentation. We hypothesize that good boundaries, for example those close to word boundaries, result in smaller reconstruction errors because the audio segments for words appear more frequently in the corpus and thus

the embeddings are trained better with lower reconstruction errors. Therefore, one proper choice for the first term in the reward function may be r_{mse}, which is the negative reconstruction error, $r_{mse} = -\mathcal{L}_{mse}$.

At the same time, it is important to have a guideline for the proper number of segments N in an utterance of a given length T'. Without this guideline, the segmentation gate generates as many segments as possible in order to minimize the reconstruction error. Therefore, we design the reward such that the smaller number of segments N normalized by the utterance length T', the higher the reward:

$$r_{num} = -\frac{N}{T'}, \tag{10}$$

where N and T' are respectively the numbers of segments and frames for the utterance as in Fig. 4.

The total reward r is obtained by choosing the minimum between r_{mse} and r_{num}:

$$r = min(r_{mse}, \lambda r_{num}) \tag{11}$$

where λ is a hyperparameter to be tuned for a reasonable guideline to estimate the proper number of segments for an utterance of length T. λ is determined to make the values of r_{mse} roughly equivalent to λr_{num}. r_{mse} and r_{num} are unknown before the model training, but it is possible to estimate their average values. Here we assume the average length of spoken words is known as the prior knowledge (this is the only language specific prior knowledge we used), so the average value of r_{num} can be roughly estimated. r_{mse} is estimated by randomly segmenting the utterances first and then training a sequence-to-sequence auto-encoder. Interpolating r_{mse} and r_{num} as total reward r is also possible, but in our preliminary experiments, we found that the minimum function yielded better results than interpolation.

For the reward baseline r_b, we further use an utterance-wise reward baseline to remove the bias between utterances. For each utterance, M different sets of segment boundaries are sampled by the segmentation gate, each of which is used to evaluate a reward r_m with (11). The reward baseline r_b for the utterance is then their average:

$$r_b = \frac{1}{M}\sum_{m=1}^{M} r_m. \tag{12}$$

V. Example Application: Unsupervised Query-by-Example Spoken Term Detection

Here we consider unsupervised query-by-example spoken term detection (QbE STD) as an example application to evaluate the quality of the embeddings. The task of unsupervised QbE STD here is to verify the existence of the input spoken query in an utterance or audio file without performing speech recognition [31]. With SSAE in Section IV, this is achieved as illustrated in Fig. 5. Given the acoustic feature sequences of a spoken query and a spoken document, SSAE represents these sequences as embeddings, $\mathbf{q} = \{ q_1, q_2, ..., q_{n_q} \}$ for the query and $\mathbf{d} = \{ d_1, d_2, ..., d_{n_d} \}$ for the document. Here $\mathbf{q}$ and $\mathbf{d}$ are sequence z obtained by SSAE in Section IV-A with a spoken

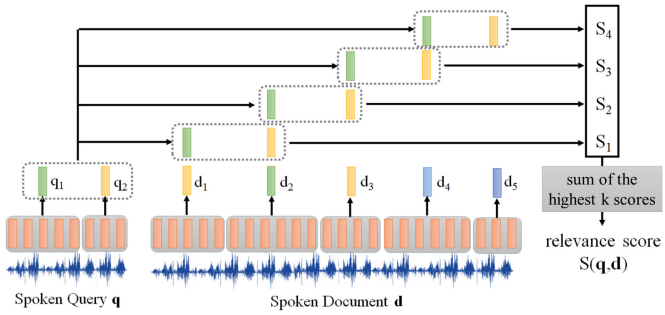

Fig. 5. Unsupervised query-by-example spoken term detection (QbE STD) with SSAE.

query or a spoken document as input, respectively. With the embeddings, simple subsequence matching is used to evaluate the relevance score $S(\mathbf{q}, \mathbf{d})$ between $\mathbf{q}$ and $\mathbf{d}$. First, we compute S_n for each position in $\mathbf{d}$.

$$S_n = \prod_{m=1}^{n_q} Sim(q_m, d_{m+n-1}). \tag{13}$$

Cosine similarity can be used in the similarity measure in (13). As shown in the right part of Fig. 5, $S_1 = sim(q_1, d_1) \cdot sim(q_2, d_2)$, $S_2 = sim(q_1, d_2) \cdot sim(q_2, d_3)$ and so on. The relevance score $S(\mathbf{q}, \mathbf{d})$ between the query $\mathbf{q}$ and document $\mathbf{d}$ is then the sum of the highest k scores among S_n obtained in (13).

VI. Experiment: Audio Representation

A. Experimental Setup

In this section, we used Librispeech [76] as the speech corpus. The dataset was segmented according to the word boundaries obtained by forced alignment to the reference transcriptions. We used the 100 hours clean speech audio data set for model training. For evaluation, another 3000 utterances were used. Thirty-nine-dimension MFCCs were used as the acoustic features. The phonetic encoder was a 2-layer GRU with a 256-node hidden layer; the speaker encoder uses the same architecture. In the case without disentanglement, the RNN decoder was also a 2-layer GRU with a 256-node hidden layer. However, in the disentanglement model, because the RNN decoder takes as input the concatenation of the phonetic and speaker encoders, we set its size to 512. The speaker classifier was a fully-connected feed-forward network with two 256-node hidden layers. The models were trained with the Adam optimizer with a batch size of 64. Algorithm 1's T_{dis} was set to 3.

B. Experimental Results

We trained the SA models on the training set to encode the segments in the evaluation set, which were never seen during training, and then computed the cosine similarity between each segment pair. We computed the average cosine similarity of the vector representations for each pair of audio segments in the testing set, and compared it with the phoneme sequence edit distance (PSED). Shown in Fig. 6 are the average and variance

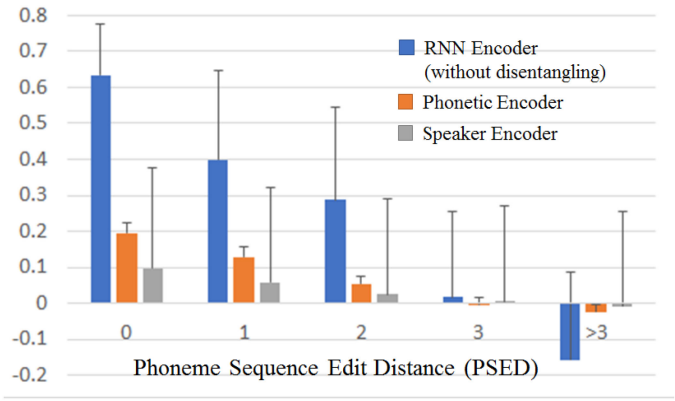

Fig. 6. Average cosine similarity and variance (length of black line on each bar) between vector representations for all segment pairs in the evaluation set, clustered by phoneme sequence edit distance (PSED).

(the length of the black line on each bar) of the cosine similarity for groups of pairs clustered by the PSED (PSED $= 0, 1, 2, 3$ and > 3) between the two words.

In Fig. 6, the cosine similarities of the RNN encoder and phonetic encoder decrease as the edit distances increase; that is, the vector representations for words with similar pronunciations are in close proximity to each other. This means that both the RNN encoder and the phonetic encoder indeed encode the sequential phonetic structures into fixed-length vectors. Clearly, the speaker encoder output includes little phonetic information because speaker encoder similarities are almost independent of the PSED. We also find from the means of similarities that the RNN encoder clearly distinguishes word segments with different phonemes even without disentangling features. For example, the similarity for word segments whose phonemes are exactly the same (PSED $= 0$) is 0.63, while the similarity for word segments with one different phoneme (PSED $= 1$) is only 0.40. However, their similarities have very large variances. For example, the variances of the group with one different phoneme is 0.24, which leads to ambiguity between different groups. This is reasonable because it is well-known that even with exactly identical phoneme sequences, acoustic realizations can differ greatly for different speakers. For the phonetic encoder, the mean similarities between different groups are not as remarkable as for the RNN encoder; however, the variances in each group are much smaller (0.018–0.029). This shows that disentangling features separate the values of the similarities of different groups.

In addition, we performed an ablation study to verify the effectiveness of different components in Fig. 3 to help disentangle the features. Similar to the setting used before [4], we used ABX metric to evaluate the performance. In all triplet minimal pairs, only words with three phonemes were considered. A and X corresponded to the same text word, and A and B only differed in the central phoneme. For the within-speaker task, A, B and X belonged to the same speaker (e.g. $A = beg_{T1}, B = bag_{T1}, X = beg'_{T1}$); for the across-speaker task, A and B belonged to the same speaker, and X to a different one (e.g. $A = beg_{T1}, B = bag_{T1}, X = beg_{T2}$). Then we calculated

TABLE I
WITHIN- AND ACROSS-SPEAKER ABX SCORES FOR THE LEARNED VECTOR REPRESENTATIONS. HERE AN ABLATION STUDY WAS PERFORMED BY REMOVING SOME LOSS TERMS IN ALGORITHM 1, OR SOME PARTS OF FIG. 3

Loss	Within Speaker	Across Speaker
Same as Alg. 1	**12.5**	**19.2**
No $\mathcal{L}_c$	14.0	19.8
No $\mathcal{L}_s$	13.1	20.2
No $\mathcal{L}_c$ & $\mathcal{L}_s$	15.2	20.0

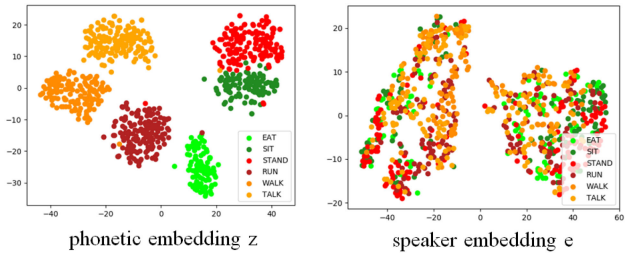

Fig. 7. Output of phonetic encoder and speaker encoder for six different words.

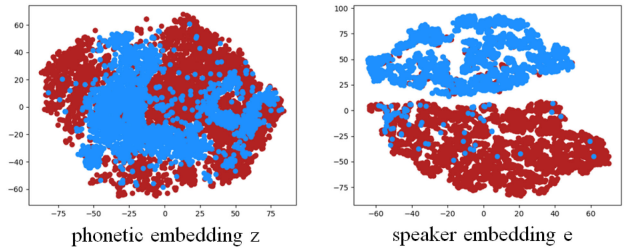

Fig. 8. Output of phonetic encoder and speaker encoder for two different speakers.

the cosine similarities of the (A, X) pair and (B, X) pair to obtain the discriminability. We converted the results into error rates listed in Table I. The first row in the table shows the result of the original Algorithm 1, and the results in other rows reflect the degradation of performance because some loss terms in Algorithm 1 or components of Fig. 3 were removed. The results clearly show that each of the components in Fig. 3 is helpful for disentanglement of features.

In the following experiments, we further compare the performance of the RNN encoder without feature disentanglement and the phonetic encoder on QbE STD.

C. Visualization

In this subsection, we visualize the representations of the audio segments in the evaluation set for further analysis.

Figures 7 and 8 show the difference between the outputs of the phonetic and speaker encoders. In Fig. 7, we show the representations of the audio segments of six different words ("eat", "sit", "stand", "run", "walk", and "talk") from a number of male and female speakers. Each point in Fig. 7 corresponds to a representation of an audio segment; different words are represented using different colors. The points in the left and right parts of Fig. 7 are the representations of the same sets of segments but with different encoders. The left part is the output of the phonetic encoder

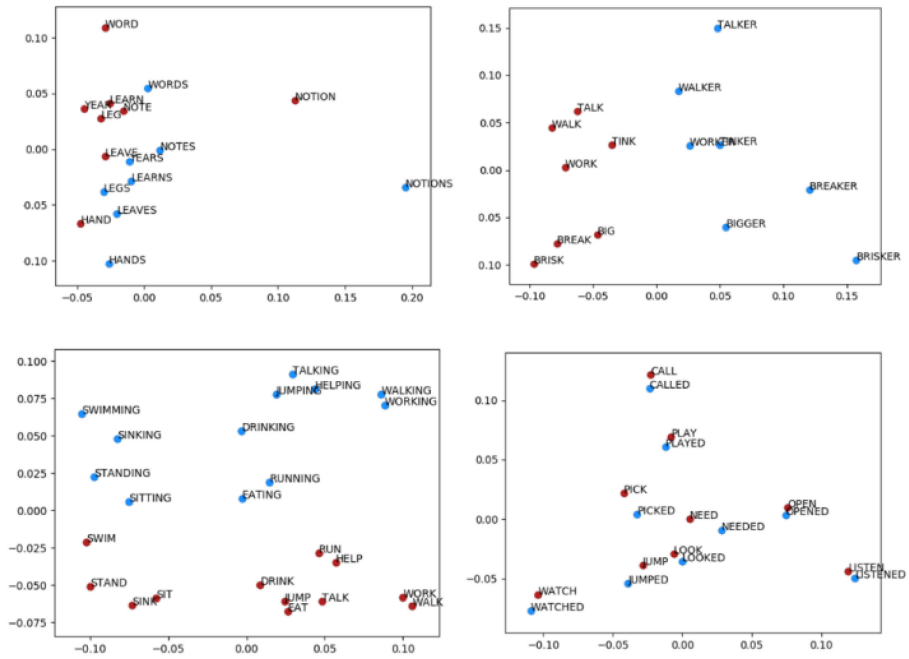

Fig. 9. Averaged representations for four sets of words.

z, while the right part is the output of the speaker encoder e. The representations were reduced to two dimensions using PCA. It is clear that the phonetic representations distinguish different words, while the speaker representations from the six words are mixed together. We also find that the speaker representations are clustered into two groups corresponding to males and females. The setup of Fig. 8 is parallel to that of Fig. 7; here we show the representations of the audio segments from two speakers. The segment representations of the two speakers correspond to the red and blue points. The phonetic representations do not distinguish the audio segments of the two speakers because their utterances show no remarkable differences. The audio segments of the two speakers, however, show very different speaker representations.

For another test, we selected four sets of words that differ only in the last few phonemes. We averaged the phonetic representations z of the audio segments corresponding to the same word, and reduced the dimensionality of the averaged presentations to 2 using PCA. The averaged representation of word w is denoted as $V(w)$. From the results, shown in Fig. 9, we see that the representations z constitute very good descriptions for the sequential phonemic structures of the acoustic segments. For example, in the leftmost figure of Fig. 9, we observe that $V(SIT) - V(SITTING) \approx V(STAND) - V(STANDING)$. Several similar examples are found in Fig. 9.

For quantitative analysis, we conducted an experiment to evaluate whether the difference vector between the phonetic representation of a certain word w and that of the word plus a suffix, such as $w - ing$, would be consistent regardless of what w was. More specifically, for a certain pair $(V(w_1), V(w_2))$, we calculated $V(w_2) + V(w_1 - ing) - V(w_1)$, and used mean reciprocal rank (MRR) (harmonic mean of the retrieval ranks) to serve as the retrieval evaluation measure of $V(w_2 - ing)$. The retrieval results of four sets of words plus suffixes ($w - ing$, $w - ed$, $w - s$ and $w - er$) are listed in Table II. Here we also compared the performance of representations with or without disentanglement. It can be observed that the difference vectors mentioned above were consistent to an extent, and again disentanglement of features improved the retrieval results.

TABLE II
MEAN RECIPROCAL RANK (MRR) SCORES FOR RETRIEVAL RESULTS OF VECTOR REPRESENTATIONS OF WORDS PLUS FOUR KINDS OF SUFFIXES

Loss	+ing	+ed	+s	+er
Same as Alg. 1	**0.09**	**0.26**	**0.13**	**0.09**
No $\mathcal{L}_c$ & $\mathcal{L}_s$	0.08	0.19	0.09	0.08

VII. EXPERIMENT: SEGMENTATION

A. Experimental Setup

We conducted segmentation experiments on TIMIT and GlobalPhone [77], specifically Czech, French, and German. For TIMIT the ground truth word boundaries were provided, while for GlobalPhone we used the forced-aligned word boundaries. In this section, both the RNN encoder and decoder of the SSAE consist of one hidden layer with 100 LSTM units. Feature disentanglement does not apply in the experiments of this section. That is, here we do not have a speaker encoder and speaker classifier. The segmentation gate consists of two 256-node LSTM layers. All parameters were trained with Adam [78]. We set M in (12) for estimating the reward baseline r_b to be 5, and $\lambda = 5$ in (11). The word boundaries were initialized randomly. The proximal policy optimization algorithm [79] was used in the policy

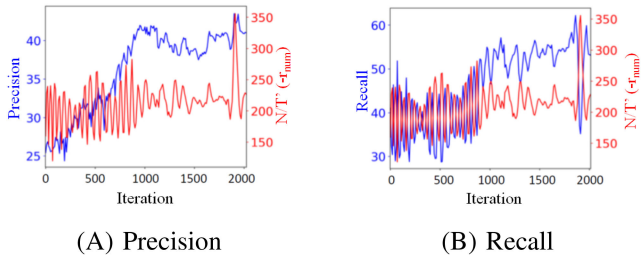

Fig. 10. SSAE learning curves on TIMIT validation set. Red curves are for N/T' ($-r_{num}$ in (10)), where N is the number of segments, and T' is the number of acoustic features. Blue curves are (A) precision and (B) recall.

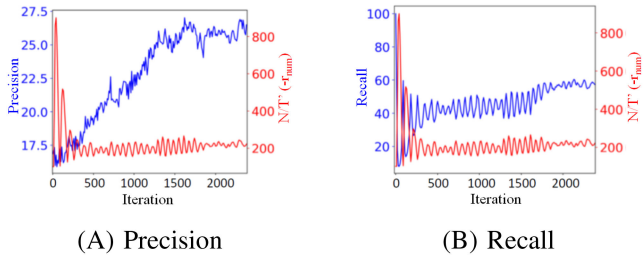

Fig. 11. SSAE learning curves on Czech validation set.

TABLE III
SPOKEN WORD SEGMENTATION PERFORMANCE, COMPARED TO DIFFERENT METHODS FOR VARIOUS CORPORA

Method	TIMIT			Czech	French	German
	Precision	Recall	F1	F1		
Random	24.60	41.08	30.77	22.56	32.66	25.41
HAC	26.84	46.21	33.96	30.84	33.75	27.09
GAS	33.22	52.39	40.66	29.53	31.11	32.89
SSAE	37.06	51.55	43.12	37.78	48.14	31.69

gradient. The tolerance window for word segmentation evaluation was taken as 40 ms. The acoustic features used were 39-dim MFCCs with utterance-wise CMVN.

B. Experimental Results

Fig. 10 shows the SSAE learning curves on the TIMIT validation set. Fig. 11 is the results for the Czech validation set; we do not show the French and German results because their trends mirror that of Fig. 11. We see that SSAE gradually learns to segment utterances into spoken words because both the precision and recall (blue curves in Fig. 10 and Fig. 11 respectively) increase during training. The reward r_{num} in (10) (red curves) fluctuates initially during training and tends to converge at the end.

Table III shows the spoken word segmentation performance of the proposed SSAE in terms of precision, recall, and F1 score. We compared the SSAE results with three baselines: random segments, gate activation signals (GAS) [22], and hierarchical agglomerative clustering (HAC) [24], [80]. We observe that SSAE significantly outperforms the other baselines on all languages other than German GAS, to which it is comparable. An example of segmentation by SSAE is shown in Fig. 12.

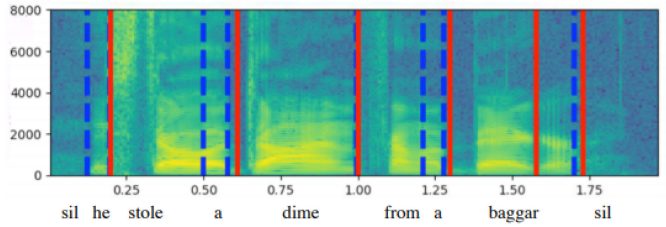

Fig. 12. An example of segmentation by SSAE. The blue dashed lines were the oracle boundaries and the red lines were produced by SSAE.

TABLE IV
SPOKEN TERM DETECTION PERFORMANCE IN MEAN AVERAGE PRECISION (MAP) FOR PROPOSED SSAE AS COMPARED TO AUDIO WORD2VEC EMBEDDINGS TRAINED WITH SPOKEN WORDS SEGMENTED WITH OTHER METHODS FOR DIFFERENT LANGUAGES. RANDOM BASELINE (RAN.) ASSIGNS A RANDOM SCORE TO EACH QUERY-DOCUMENT PAIR. STANDARD FRAME-BASED DTW IS THE PRIMARY BASELINE; ORACLE SEGMENTATION IS THE UPPER BOUND

			Embeddings (different seg.)			
Lang.	Ran.	DTW	GAS	HAC	SSAE	Oracle
TIMIT	0.74	12.02	8.29	0.91	23.27	30.28
Czech	0.38	16.59	0.68	1.13	19.41	22.56
French	0.27	11.72	0.40	0.92	21.70	29.66
German	0.18	6.07	0.27	0.26	13.82	21.52

VIII. EXPERIMENT: QBYE STD

In this section, we evaluate the performance of audio word2vec on QbyE STD. We use mean average precision (MAP) as the evaluation measure.

The first set of experiment is conducted on English (TIMIT), Czech, French and German. The testing set utterances were used as spoken documents [20]. We randomly selected as the query words five words for each language containing a variety of phonemes; from the training set we used several occurrences of each of these phoneme-rich words as spoken queries. For English, Czech, French, and German, we used 29, 21, 25, and 23 spoken queries for evaluation, respectively. The number of the highest scores among S_n obtained in (13), k, in Section V and Fig. 5 was set to be 1.

We compared the quality of the SSAE embeddings with other kinds of audio word2vec embeddings trained with the segments generated using the different segmentation methods. Although the spoken queries are single words, the models do not know this. The spoken queries are also segmented into segments using different segmentation approaches. The results are listed in Table IV. The performance for embeddings trained with the ground truth word boundaries (oracle) in the last column serves as the upper bound. For the random baseline in the first column, a random score was assigned to each query-document pair. In the second column we also report the performance of standard frame-based dynamic time warping (DTW) [20] as a primary baseline. From the table we observe that the oracle method outperforms all other methods on all corpora. SSAE outperforms the DTW baseline because DTW cannot identify spoken words if the speaker or gender characteristics are very different; such varying signal characteristics are better absorbed in the audio word2vec training. These experimental results confirm that the

TABLE V
MAP PERFORMANCE OF QUERY-BY-EXAMPLE SPOKEN TERM DETECTION (QBE STD). $NoDis$ AND PE ARE THE RESULTS WITHOUT FEATURE DISENTANGLEMENT AND USING THE PHONETIC ENCODER RESPECTIVELY. THE SEGMENTATION OF TRAINING AND TESTING SETS CAN BE EITHER ORACLE OR BY SSAE. DIFFERENT k FOR THE SEARCH ALGORITHM IN SECTION V AND FIG. 5 ARE TESTED

Training Set	Testing Set	$k=1$		$k=40$	
		$NoDis$	PE	$NoDis$	PE
Oracle	SSAE	16.41	17.24	17.27	21.79
SSAE	SSAE	15.54	17.09	17.60	19.60

SSAE embeddings do carry the sequential phonetic structure information from the utterances, leading to the better STD performance. In most cases, the performance of the GAS and HAC embeddings are not too far from random. It appears that audio word2vec does not train well if the performance of spoken word segmentation does not exceed some minimum level. That is, spoken word segmentation boundaries made the biggest impact on STD performance. We also note that although the GAS segmentation performance was slightly better than that of SSAE for German, SSAE clearly outperformed GAS on German QbyE STD.

Then we conducted experiments with more spoken queries based on Librispeech. The audio word2vec models were trained on the 100 hour clean data set. The spoken archive to be retrieved is the clean testing data set. The chapters are considered as the unit to be retrieved. We have 361 spoken queries also from Librispeech, but not included in the training set or retrieved utterances. All the spoken queries correspond to a single word in the experiments.

The experimental results are shown in Table V. $NoDis$ and PE are the results without feature disentanglement and using the phonetic encoder respectively. Oracle means we segmented the audio using the word boundaries obtained by forced alignment with the reference transcriptions. SSAE in Table V means the segments were obtained by SSAE. Different k ($k=1$ or 40) for the search algorithm in Section V and Fig. 5 are tested. Clearly, the output of the phonetic encoder outperformed the features without disentanglement because it reduces the speaker dependence (PE v.s. $NoDis$).

IX. CONCLUDING REMARKS

In this paper, we extend the research of audio word2vec. We use domain adversarial training to automatically learn encoders that encode different information. The experimental results show that this thus disentangles the phonetic and speaker information. We further propose an SSAE trained with reinforcement learning, in which word-level segmentation and segment representation are jointly learned.

Audio embedding has many possible applications beyond STD. For example, audio embedding can be considered as better audio representation for speech recognition. It is possible to use a large amount of unlabeled audio to learn the embeddings to improve low-resource speech recognition. The learned embeddings can also be used in the applications related to spoken language understanding of low-resource language like spoken question answering, spoken content summarization, and speech translation. These spoken language understanding systems can take the learned audio embedding as input, instead of the transcriptions of spoken content.

REFERENCES

[1] M. A. Hasegawa-Johnson *et al.*, "ASR for under-resourced languages from probabilistic transcription," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 1, pp. 50–63, Jan. 2017.

[2] N. F. Chen *et al.*, "Multitask learning for phone recognition of under-resourced languages using mismatched transcription," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 3, pp. 501–514, Mar. 2018.

[3] 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. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 8111–8115.

[4] E. Dunbar *et al.*, "The zero resource speech challenge 2017," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2017, pp. 323–330.

[5] M. Versteegh *et al.*, "The zero resource speech challenge 2015," in *Proc. INTERSPEECH*, 2015, pp. 3169–3173.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Information Process. Syst.*, 2013, pp. 3111–3119.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, arXiv:1301.3781.

[8] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.

[9] S. Bansal, H. Kamper, A. Lopez, and S. Goldwater, "Towards speech-to-text translation without speech recognition," in *Proc. 15th Conf. Eur. Ch. Assoc. Comput. Linguistics*, 2017, pp. 474–479.

[10] 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," in *Proc. INTERSPEECH*, 2016.

[11] C.-H. Shen, J. Y. Sung, and H.-Y. Lee, "Language transfer of audio word2vec: Learning audio segment representations without target language data," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 2231–2235.

[12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[13] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proc. Unsupervised Transfer Learn. Challenges Mach. Learn.* vol. 7, pp. 37–49, 2012.

[14] J. Li, M.-T. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics/7th Int. Joint Conf. Natural Lang. Process.*, 2015, pp. 1106–1115.

[15] R. Kiros *et al.*, "Skip-thought vectors," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3294–3302.

[16] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using LSTMs," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 843–852.

[17] Y. Ganin *et al.*, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, pp. 2096–2030, 2016.

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

[19] C.-T. Chung, C.-a. Chan, and L.-s. Lee, "Unsupervised spoken term detection with spoken queries by multi-level acoustic patterns with varying model granularity," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 7814–7818.

[20] Y. Zhang and J. R. Glass, "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2009, pp. 398–403.

[21] 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.

[22] 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, pp. 3822–3826.

[23] O. Räsänen, "Basic cuts revisited: Temporal segmentation of speech into phone-like units with statistical learning at a pre-linguistic level," in *Proc. Annu. Meeting Cogn. Sci. Soc.*, 2014, pp. 2817–2822.

[24] Y. Qiao, N. Shimomura, and N. Minematsu, "Unsupervised optimal phoneme segmentation: Objectives, algorithm and comparisons," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2008, pp. 3989–3992.

[25] C.-y. Lee and J. Glass, "A nonparametric Bayesian approach to acoustic model discovery," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics, Long Papers-Volume 1s*, 2012, pp. 40–49.

[26] 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 *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 6269–6273.

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

[28] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[29] N. Dehak, R. Dehak, P. Kenny, N. Brummer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Proc. INTERSPEECH*, 2009, pp. 1559–1562.

[30] K. Levin, K. Henry, A. Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2013, pp. 410–415.

[31] K. Levin, A. Jansen, and B. Van Durme, "Segmental acoustic indexing for zero resource keyword search," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 5828–5832.

[32] H.-Y. Lee and L.-S. Lee, "Enhanced spoken term detection using support vector machines and weighted pseudo examples," *IEEE Trans. Audio, Speech, Lang. Process.,* vol. 21, no. 6, pp. 1272–1284, Jun. 2013.

[33] I.-F. Chen and C.-H. Lee, "A hybrid HMM/DNN approach to keyword spotting of short words," in *Proc. INTERSPEECH*, 2013, pp. 1574–1578.

[34] A. Norouzian, A. Jansen, R. Rose, and S. Thomas, "Exploiting discriminative point process models for spoken term detection," in *Proc. INTERSPEECH*, 2012, pp. 2442–2445.

[35] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end ASR-free keyword search from speech," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 8, pp. 1351–1359, Dec. 2017.

[36] S. Bengio and G. Heigold, "Word embeddings for speech recognition," in *Proc. INTERSPEECH*, 2014, pp. 1053–1057.

[37] G. Chen, C. Parada, and T. N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 5236–5240.

[38] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 4950–4954.

[39] W. He, W. Wang, and K. Livescu, "Multi-view recurrent neural acoustic word embeddings," 2016, arXiv:1611.04496.

[40] S. Settle, K. Levin, H. Kamper, and K. Livescu, "Query-by-example search with discriminative neural acoustic word embeddings," pp. 2874–2878, 2017.

[41] A. L. Maas, S. D. Miller, T. M. Oneil, A. Y. Ng, and P. Nguyen, "Word-level acoustic modeling with convolutional vector regression," in *Proc. ICML Workshop Representation Learn.*, Edinburgh, U.K., 2012.

[42] Y.-A. Chung and J. Glass, "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech," in *Proc. INTERSPEECH*, 2018, pp. 811–815.

[43] N. Holzenberger, M. Du, J. Karadayi, R. Riad, and E. Dupoux, "Learning word embeddings: Unsupervised methods for fixed-size representations of variable-length speech segments," in *Proc. INTERSPEECH*, 2018, pp. 2683–2687.

[44] H. Kamper, "Truly unsupervised acoustic word embeddings using weak top-down constraints in encoder-decoder models," in *Proc. ICASSP 2019–2019 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 6535–6569.

[45] S. Settle and K. Livescu, "Discriminative acoustic word embeddings: Recurrent neural network-based approaches," in *Proc. IEEE Spoken Lang. Technol. Workshop*, 2016, pp. 503–510.

[46] A. Jansen *et al.*, "Towards learning semantic audio representations from unlabeled data," in *Proc. NIPS Workshop Mach. Learn. Audio Signal Process.*, 2017.

[47] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2017, pp. 16–23.

[48] W.-N. Hsu, Y. Zhang, and J. Glass, "Learning latent representations for speech generation and transformation," in *Proc. INTERSPEECH*, 2017, pp. 1273–1277.

[49] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1876–1887.

[50] Y.-C. Chen, S.-F. Huang, C.-H. Shen, H.-y. Lee, and L.-s. Lee, "Phonetic-and-semantic embedding of spoken words with applications in spoken content retrieval," in *Proc. IEEE Spoken Lang. Technol. Workshop*, 2018, pp. 941–948.

[51] Z. Meng, Z. Chen, V. Mazalov, J. Li, and Y. Gong, "Unsupervised adaptation with domain separation networks for robust speech recognition," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2017, pp. 214–221.

[52] C. T. Chung and L. S. Lee, "Unsupervised discovery of structured acoustic tokens with applications to spoken term detection," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 2, pp. 394–405, Feb. 2018.

[53] A. Garcia and H. Gish, "Keyword spotting of arbitrary words using minimal speech resources," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2006, p. I.

[54] A. Jansen and K. Church, "Towards unsupervised training of speaker independent acoustic models," in *Proc. INTERSPEECH*, 2011, pp. 1693–1696.

[55] A. Jansen, K. Church, and H. Hermansky, "Towards spoken term discovery at scale with zero resources," in *Proc. INTERSPEECH*, 2010, pp. 1676–1679.

[56] A. Park and J. Glass, "Unsupervised pattern discovery in speech," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 1, pp. 186–197, Jan. 2008.

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

[58] L. Wang, E. S. Chng, and H. Li, "An iterative approach to model merging for speech pattern discovery," in *Proc. Asia-Pac. Signal Inf. Process. Assoc.*, 2011.

[59] N. Vanhainen and G. Salvi, "Word discovery with beta process factor analysis," in *Proc. INTERSPEECH*, 2012, pp. 798–801.

[60] J. Driesen and H. Van hamme, "Fast word acquisition in an NMF-based learning framework," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2012, pp. 5137–5140.

[61] Y. Zhang and J. Glass, "Towards multi-speaker unsupervised speech pattern discovery," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2010, pp. 4366–4369.

[62] C.-H. Lee, F. K. Soong, and B.-H. Juang, "A segment model based approach to speech recoginition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1988, pp. 501–541.

[63] 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 *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2012, pp. 5157–5160.

[64] H. Kamper, A. Jansen, and S. Goldwater, "Unsupervised word segmentation and lexicon discovery using acoustic word embeddings," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 4, pp. 669–679, Apr. 2016.

[65] H. Kamper, K. Livescu, and S. Goldwater, "An embedded segmental K-means model for unsupervised segmentation and clustering of speech," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2017, pp. 719–726.

[66] M. Elsner and C. Shain, "Speech segmentation with a neural encoder model of working memory," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1070–1080.

[67] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, arXiv:1406.1078.

[68] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst*, 2014, pp. 3104–3112.

[69] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 10–21.

[70] S. Semeniuta, A. Severyn, and E. Barth, "A hybrid convolutional variational autoencoder for text generation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 627–637.

[71] P. Nema, M. M. Khapra, A. Laha, and B. Ravindran, "Diversity driven attention model for query-based abstractive summarization," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics (Vol. 1: Long Papers)*, 2017, pp. 1063–1072.

[72] N. Zeghidour, G. Synnaeve, N. Usunier, and E. Dupoux, "Joint learning of speaker and phonetic similarities with siamese networks," in *Proc. INTERSPEECH*, 2016, pp. 1295–1299.
[73] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," in *Proc. Int. Conf. Mach. Learn.*, 2017.
[74] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein GANs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5769–5779.
[75] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. 12th Int. Conf. Neural Inf. Process. Syst.*, 1999, pp. 1057–1063.
[76] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 5206–5210.
[77] T. Schultz, "Globalphone: A multilingual speech and text database developed at Karlsruhe University," in *Proc. INTERSPEECH*, 2002, pp. 345–348.
[78] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[79] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, arXiv:1707.06347.
[80] C.-a. Chan, "Unsupervised spoken term detection with spoken queries," Ph.D. dissertation, Nat. Taiwan Univ., Taipei, Taiwan, 2012.

Yi-Chen Chen received the bachelor's degree in electrical engineering, National Taiwan University (NTU), Taipei, Taiwan, in 2017. He is currently working toward the M.S. degree with the Graduate Institute of Communication Engineering, NTU, working on self-supervised/semi-supervised/transfer learning and speech processing.

Sung-Feng Huang received the bachelor's degree from National Taiwan University (NTU), Taipei, Taiwan, in 2017. He is currently working toward the master's degree with the Graduate Institute of Communication Engineering, NTU. He mainly works on learning representations unsupervisedly, speech recognition, spoken term detection, and machine learning techniques.

Hung-yi Lee received the M.S. and Ph.D. degrees from National Taiwan University, Taipei, Taiwan, in 2010 and 2012, respectively. From September 2012 to August 2013, he was a Postdoctoral Fellow with the Research Center for Information Technology Innovation, Academia Sinica. From September 2013 to July 2014, he was a visiting scientist with the Spoken Language Systems Group of MIT Computer Science and Artificial Intelligence Laboratory. He is currently an Assistant Professor with the Department of Electrical Engineering, NTU, with a joint appointment with the Department of Computer Science and Information Engineering, NTU. His research interests focus on spoken language understanding, speech recognition, and machine learning.

Yu-Hsuan Wang received the M.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 2018. He is currently working toward the M.S. degree with Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA, working on information extraction technology.

Chia-Hao Shen received the M.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2017. He is currently an NLP data scientist with CompStak, New York, NY, USA, working on natural language understanding and reinforcement learning. His research is focused on audio/speech representation.