Audio Word2Vec: Unsupervised Learning of Audio Segment Representations using Sequence-to-sequence Autoencoder

Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, Lin-Shan Lee

College of Electrical Engineering and Computer Science, National Taiwan University

{b01902040, b01902038, r04921047, hungyilee}@ntu.edu.tw, lslee@gate.sinica.edu.tw

Abstract

The vector representations of fixed dimensionality for words (in text) offered by Word2Vec have been shown to be very useful in many application scenarios, in particular due to the semantic information they carry. This paper proposes a parallel version, the Audio Word2Vec. It offers the vector representations of fixed dimensionality for variable-length audio segments. These vector representations are shown to describe the sequential phonetic structures of the audio segments to a good degree, with very attractive real world applications such as query-by-example Spoken Term Detection (STD). In this STD application, the proposed approach significantly outperformed the conventional Dynamic Time Warping (DTW) based approaches at significantly lower computation requirements. We propose unsupervised learning of Audio Word2Vec from audio data without human annotation using Sequence-to-sequence Autoencoder (SA). SA consists of two RNNs equipped with Long Short-Term Memory (LSTM) units: the first RNN (encoder) maps the input audio sequence into a vector representation of fixed dimensionality, and the second RNN (decoder) maps the representation back to the input audio sequence. The two RNNs are jointly trained by minimizing the reconstruction error. Denoising Sequence-to-sequence Autoencoder (DSA) is further proposed offering more robust learning.

1. Introduction

Word2Vec [1, 2, 3] transforming each word (in text) into a vector of fixed dimensionality has been shown to be very useful in various applications in natural language processing. In particular, the vector representations obtained in this way usually carry plenty of semantic information about the word. It is therefore interesting to ask a question: can we transform the audio segment of each word into a vector of fixed dimensionality? If yes, what kind of information these vector representations can carry and in what kind of applications can these "audio version of Word2Vec" be useful? This paper is to try to answer these questions at least partially.

Representing variable-length audio segments by vectors with fixed dimensionality has been very useful for many speech applications. For example, in speaker identification [4], audio emotion classification [5], and spoken term detection (STD) [6, 7, 8], the audio segments are usually represented as feature vectors to be applied to standard classifiers to determine the speaker or emotion labels, or whether containing the input queries. In query-by-example spoken term detection (STD), by representing each word segment as a vector, easier indexing makes the retrieval much more efficient [9, 10, 11].

But audio segment representation is still an open problem. It is common to use i-vectors to represent utterances in speaker identification [4], and several approaches have been successfully used in STD [10, 6, 7, 8]. But these vector representations may not be able to precisely describe the sequential phonetic structures of the audio segments as we wish to have in this paper, and these approaches were developed primarily in more heuristic ways, rather than learned from data. Deep learning has also been used for this purpose [12, 13]. By learning 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 [13]. However, this approach is supervised and therefore needs a large amount of labeled training data.

On the other hand, autoencoder has been a very successful machine learning technique for extracting representations in an unsupervised way [14, 15], but its input should be vectors of fixed dimensionality. This is a dreadful limitation because audio segments are intrinsically expressed as sequences of arbitrary length. A general framework was proposed to encode a sequence using Sequence-to-sequence Autoencoder (SA), in which a RNN is used to encode the input sequence into a fixed-length representation, and then another RNN to decode this input sequence out of that representation. This general framework has been applied in natural language processing [16, 17] and video processing [18], but not yet on speech signals to our knowledge.

In this paper, we propose to use Sequence-to-sequence Autoencoder (SA) to represent variable-length audio segments by vectors with fixed dimensionality. We hope the vector representations obtained in this way can describe more precisely the sequential phonetic structures of the audio signals, so the audio segments that sound alike would have vector representations nearby in the space. This is referred to as Audio Word2Vec in this paper. Different from the previous work [12, 13, 11], here learning SA does not need any supervision. That is, only the audio segments without human annotation are needed, good for applications in the low resource scenario. Inspired from denoising autoencoder [19, 20], an improved version of Denoising Sequence-to-sequence Autoencoder (DSA) is also proposed. Among the many possible applications, we chose query-by-example STD in this preliminary study, and show that the proposed Audio Word2Vec can be very useful. Query-by-example STD using Audio Word2Vec here is much more efficient than the conventional Dynamic Time Warping (DTW) based approaches, because only the similarities between two single vectors are needed, in additional to the significantly better retrieval performance obtained.

2. Proposed Approach

The goal here is to transform an audio segment represented by a variable-length sequence of acoustic features such as MFCC,

$\mathbf{x} = (x_1, x_2, ..., x_T)$, where x_t is the acoustic feature at time t and T is the length, into a vector representation of fixed dimensionality $\mathbf{z} \in \mathbb{R}^d$, where d is the dimensionality of the encoded space. It is desired that this vector representation can describe to some degree the sequential phonetic structure of the original audio segment. Below we first give a recap on the RNN Encoder-Decoder framework [21, 22] in Section 2.1, followed by the formal presentation of the proposed Sequence-to-sequence Autoencoder (SA) in Section 2.2, and its extension in Section 2.3.

2.1. RNN Encoder-Decoder framework

RNNs are neural networks whose hidden neurons form a directed cycle. Given a sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$, RNN updates its hidden state $\mathbf{h}_t$ according to the current input x_t and the previous $\mathbf{h}_{t-1}$. The hidden state $\mathbf{h}_t$ 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. In practice, RNN does not seem to learn long-term dependencies [23], so LSTM [24] has been widely used to conquer such difficulties. Because many amazing results were achieved by LSTM, RNN has widely been equipped with LSTM units [25, 26, 27, 28, 22, 29, 30].

RNN Encoder-Decoder [22, 21] consists of an Encoder RNN and a Decoder RNN. The Encoder RNN reads the input sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$ sequentially and the hidden state $\mathbf{h}_t$ of the RNN is updated accordingly. After the last symbol x_T is processed, the hidden state $\mathbf{h}_T$ is interpreted as the learned representation of the whole input sequence. Then, by taking $\mathbf{h}_T$ as input, the Decoder RNN generates the output sequence $\mathbf{y} = (y_1, y_2, ..., x_{T'})$ sequentially, where T and T' can be different, or the length of $\mathbf{x}$ and $\mathbf{y}$ can be different. Such RNN Encoder-Decoder framework is able to handle input of variable length. Although there may exist a considerable time lag between the input symbols and their corresponding output symbols, LSTM units in RNN are able to handle such situations well due to the powerfulness of LSTM in modeling long-term dependencies.
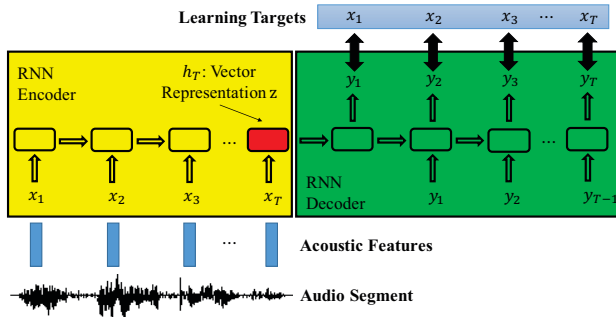
2.2. Sequence-to-sequence Autoencoder (SA)

Figure 1: Sequence-to-sequence Autoencoder (SA) consists of two RNNs: RNN Encoder (the left large block in yellow) and RNN Decoder (the right large block in green). The RNN Encoder reads an audio segment represented as an acoustic feature sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$ and maps it into a vector representation of fixed dimensionality $\mathbf{z}$; and the RNN Decoder maps the vector $\mathbf{z}$ to another sequence $\mathbf{y} = (y_1, y_2, ..., y_T)$. The RNN Encoder and Decoder are jointly trained to make the output sequence $\mathbf{y}$ as close to the input sequence $\mathbf{x}$ as possible, or to minimize the reconstruction error.

Figure 1 depicts the structure of the proposed Sequence-to-sequence Autoencoder (SA), which integrates the RNN Encoder-Decoder framework with Autoencoder for unsupervised learning of audio segment representations. SA consists

of an Encoder RNN (the left part of Figure 1) and a Decoder RNN (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 Encoder RNN reads each acoustic feature x_t sequentially and the hidden state $\mathbf{h}_t$ is updated accordingly. After the last acoustic feature x_T has been read and processed, the hidden state $\mathbf{h}_T$ of the Encoder RNN is viewed as the *learned representation* $\mathbf{z}$ of the input sequence (the small red block in Figure 1). The Decoder RNN takes $\mathbf{h}_T$ as the first input and generates its first output y_1. Then it takes y_1 as input to generate y_2, and so on. Based on the principles of Autoencoder [14, 15], 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, measured by the general mean squared error $\sum_{t=1}^{T} \|x_t - y_t\|^2$. Because the input sequence is taken as the learning target, the training process does not need any labeled data. The fixed-length vector representation $\mathbf{z}$ will be a meaningful representation for the input audio segment $\mathbf{x}$, because the whole input sequence $\mathbf{x}$ can be reconstructed from $\mathbf{z}$ by the RNN Decoder.

2.3. Denoising Sequence-to-sequence Autoencoder (DSA)

To learn more robust representation, we further apply the denoising criterion [20] to the SA learning proposed above. The input acoustic feature sequence $\mathbf{x}$ is randomly added with some noise, yielding a *corrupted* version $\tilde{\mathbf{x}}$. Here the input to SA is $\tilde{\mathbf{x}}$, and SA is expected to generate the output $\mathbf{y}$ closest to the original $\mathbf{x}$ based on $\tilde{\mathbf{x}}$. The SA learned with this denoising criterion is referred to as Denoising SA (DSA) below.

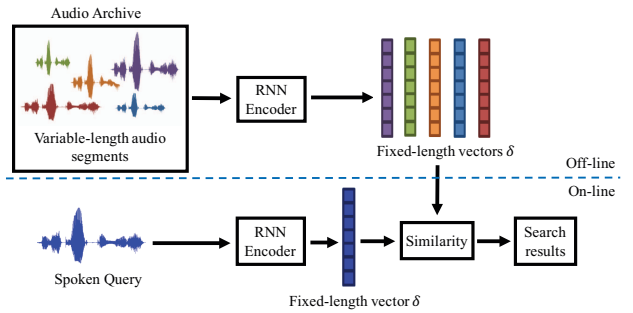# 3. An Example Application: Query-by-example STD

Figure 2: The example application of query-by-example STD. All audio segments in the audio archive are segmented based on word boundaries and represented by fixed-length vectors off-line. When a spoken query is entered, it is also represented as a vector, and the similarity between this vector and all vectors for segments in the archive are calculated, and the audio segments are ranked accordingly.

The audio segment representation $\mathbf{z}$ learned in the last section can be applied in many possible scenarios. Here in the preliminary tests we consider the unsupervised query-by-example STD, whose target is to locate the occurrence regions of the input spoken query term in a large spoken archive without speech recognition. Figure 2 shows how the representation $\mathbf{z}$ proposed here can be easily used in this task. This approach is inspired from the previous work [10], but completely different in the ways to represent the audio segments. In the upper half of Figure 2, the audio archive are segmented based on word boundaries into variable-length sequences, and then the system exploits the trained RNN encoder in Figure 1 to encode these au-

dio segments into fixed-length vectors. All these are done off-line. In the lower left corner of Figure 2, when a spoken query is entered, the input spoken query is similarly encoded by the same RNN encoder into a vector. The system then returns a list of audio segments in the archive ranked according to the cosine similarities evaluated between the vector representation of the query and those of all segments in the archive. Note that the computation requirements for the on-line process here are extremely low.

4. Experiments

Here, we report the experiments and results including the experimental setup (Section 4.1), analysis for the vector representations learned (Section 4.2), and query-by-example STD results (Section 4.3).

4.1. Experimental Setup

We used LibriSpeech corpus [31] as the data for the experiments. Due to the computation resource limitation, only 5.4 hours *dev-clean* (produced by 40 speakers) was used for training SA and DSA. Another 5.4 hours *test-clean* (produced by a completely different group of 40 speakers) was the testing set. MFCCs of 13-dim were used as the acoustic features. Both the training and testing sets were segmented according to the word boundaries obtained by forced alignment with respect to the reference transcriptions. Although the oracle word boundaries were used here for the query-by-example STD in the preliminary tests, the comparison in Section 4.3 was fair since the baselines used the same segmentation.

SA and DSA were implemented with Theano [32, 33]. The network structure and hyper parameters were set as below without further tuning if not specified:

- Both RNN Encoder and Decoder consisted of one hidden layer with 100 LSTM units. That is, each input segment was mapped into a 100-dim vector representation by SA or DSA. We adopted the LSTM described in [34], and the *peephole connections* [35] were added.

- The networks were trained by SGD without momentum, with a fixed learning rate of 0.3 and 500 epochs.

- For DSA, zero-masking [20] was used to generate the noisy input $\tilde{x}$, which randomly wiped out some elements in each input acoustic feature and set them to zero. The probability to be wiped out was set to 0.3.

For query-by-example STD, the testing set served as the audio archive to be searched through. Each word segment in the testing set was taken as a spoken query once. When a word segment was taken as a query, it was excluded from the archive during retrieval. There were 5557 queries in total. Mean Average Precision (MAP) [36] was used as the evaluation measure for query-by-example STD as the previous work [37].

4.2. Analysis of the Learned Representations

In this section, we wish to verify that the vector representations obtained here describe the sequential phonetic structures of the audio segments, or words with similar pronunciations have close vector representations.

We first used the SA and DSA learned from the training set to encode the segments in the testing set, which were never seen during training, and then computed the cosine similarity between each segment pair. The average results for groups of pairs clustered by the phoneme edit distances between the two words in the pair are in Table 1. It is clear from Table 1 that

two segments for words with larger phoneme sequence edit distances have obviously smaller cosine similarity in average. We also found that SA and DSA can even very clearly distinguish those word segments with only one different phoneme, because the average similarity for the cluster of zero edit distance is 0 (exactly the same pronunciation) is remarkably larger than that for edit distance being 1. The gap is even larger for clusters with edit distances being 2 vs. 1. For the cluster of zero edit distance, the average similarity is only 0.48-0.50. This is reasonable because it is well-known that even with exactly identical phoneme segments, the acoustic realizations can be very different. It seems DSA is slightly better than SA, because the similarity by DSA for pairs with 0 or 1 edit distances is slightly larger, while much smaller for those with 4, 5 or more edit distances. These results show that SA and DSA can in fact encode the acoustic signals into vector representations describing the sequential phonetic structures of the signals. The performance between SA and DSA can be further compared with other baselines in Section 4.3.

Table 1: The average cosine similarity between the vector presentations for all the segment pairs in the testing set, clustered by the phoneme sequence edit distances.

Phoneme Sequence Edit Distance	# of pairs	Average cosine similarity	
		SA	DSA
0 (same)	95222	0.4847	0.5012
1	95355	0.4016	0.4237
2	987934	0.2674	0.2798
3	4651318	0.0835	0.0986
4	4791482	0.0255	0.0196
5 or more	3078684	0.0051	0.0013

Because RNN Encoder in Figure 1 reads the input acoustic sequence sequentially, it may be possible that the last few acoustic features dominate the vector representation, and as a result those words with the same suffixes are hard to be distinguished by the learned representations. To justify this hypothesis, we analyze the following two sets of words with the same suffixes:

- Set 1: father, mother, another

- Set 2: ever, never, however

Same as Table 1, we used cosine similarity between the two encoded vectors averaged over each group of pairs to measure the differences, with results in Table 2. For Set 1 in Table 2,

Table 2: The average cosine similarity between the vector representations for words with the same suffixes.

| Set 1	# of pairs	Average cosine similarity	
		SA	DSA
father vs. father	406	0.5047	0.5209
father vs. mother	832	0.2639	0.2748
father vs. another	1276	0.1964	0.1861
Set 2	# of pairs	Average cosine similarity	
		SA	DSA
ever vs. ever	46	0.4027	0.4265
ever vs. never	1581	0.2475	0.2238
ever vs. however	713	0.2638	0.2419

we see the vector representations of the audio segments of "father" are much closer to each other than to those for "mother", while those for "another" are even much farther away. Similar for Set 2. There exist much more similar examples in the

dataset, but we just show two here due to the space limitation. These results show that although SA or DSA read the acoustic signals sequentially, words with the same suffix are actually clearly distinguishable from the learned vectors.

For another test, we selected two sets of word pairs differing by only the first phoneme or the last phoneme as below.

- Set 3: (new, few), (near, fear), (name, fame), (night, fight)
- Set 4: (hand, hands), (word, words), (day, days), (say, says), (thing, things)

We evaluated the difference vectors between the averages of the vector representations by SA for the two words in a pair, e.g., $\delta(\text{word1}) - \delta(\text{word2})$, and reduced the dimensionality of the difference vectors to 2 to see more easily how they actually differentiated in the vector space [1]. The results for Sets 3 and 4 are respectively in Figure 3(a) and Figure 3(b). We see in each case the difference vectors are in fact close in their directions and magnitudes, for example, $\delta(\text{new}) - \delta(\text{few})$ is close to $\delta(\text{night}) - \delta(\text{fight})$ in Figure 3(a), and $\delta(\text{days}) - \delta(\text{day})$ is close to $\delta(\text{things}) - \delta(\text{thing})$ in Figure 3(b). This implies "phoneme replacement" is somehow realizable here, for example, $\delta(\text{new}) - \delta(\text{few}) + \delta(\text{fight})$ is not very far from $\delta(\text{fight})$ in Figure 3(a), and $\delta(\text{things}) - \delta(\text{thing}) + \delta(\text{day})$ is not very far from $\delta(\text{days})$ in Figure 3(b). They are not "very close" because it is well-known that the same words can have very different acoustic realizations.
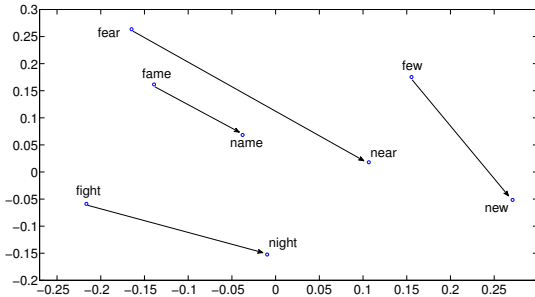
(a) Set 3: the first phoneme changes from f to n.
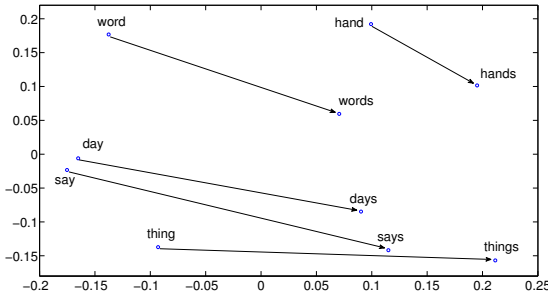
(b) Set 4: the last phoneme differs by existing s or not.

Figure 3: Difference vectors between the average vector representations for word pairs differing by (a) the first phoneme as in Set 3 and (b) the last phoneme as in Set 4.

All results above show the vector representations in test offer very good descriptions for the sequential phonemic structures of the acoustic segments.

4.3. Query-by-example STD

Here we compared the performance of using the vector representations from SA and DSA in query-by-example STD with two baselines. The first baseline used frame-based DTW [38] which is commonly used in query-by-example STD tasks. Here

we used the vanilla version of DTW (with Euclidean distance as the frame-level distance) to compute the similarities between the input query and the audio segments, and ranked the segments accordingly. The second baseline used the same framework in Figure 2, except that the RNN Encoder is replaced by a manually designed encoder. In this encoder, we first divide the input acoustic feature sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$, where x_t is the 13-dimensional acoustic feature vector at time t, into m segments with roughly equal length $\frac{T}{m}$, then average each segment of vectors into a single 13-dimensional vector, and finally concatenate these m average vectors sequentially into a vector representation of dimensionality $13 \times m$. We refer to this approach as Naïve Encoder (NE) below. Although NE is simple, similar approaches have been used in STD and achieved successful results [6, 7, 8]. We found empirically that NE with 52, 78 and 104 dimensions (or $m = 4, 6, 8$), represented by NE_{52}, NE_{78}, and NE_{104} below, yielded the best results. So NE_{52}, NE_{78}, and NE_{104} were used in the following experiments.
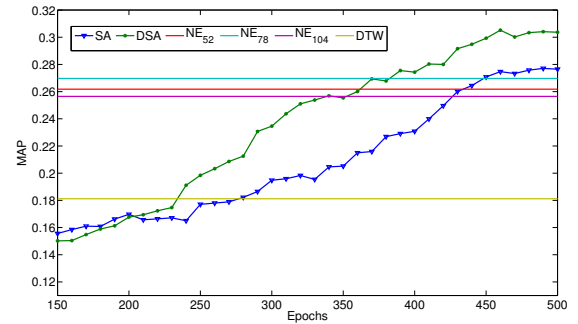
Figure 4: The retrieval performance in MAP of DTW, Naïve Encoder (NE_{52}, NE_{78}, and NE_{104}), SA and DSA. The horizontal axis is the number of epochs in training SA and DSA, not relevant to all baselines.

Figure 4 shows the retrieval performance in MAP of DTW, NE_{52}, NE_{78}, NE_{104}, SA and DSA. The horizontal axis is the number of epochs in training SA and DSA, not relevant to all baselines. Besides DTW which computes the distance between audio segments directly, all other approaches map the variable-length audio segments to fixed-length vectors for similarity evaluation, but in different approaches. DTW is not comparable with NE_{52}, NE_{78}, and NE_{104} probably because only the vanilla version was used, and the averaging process in NE may smooth some of the local signal variations which may cause disturbances in DTW. We see as the training epochs increased, both SA and DSA resulted in higher MAP than NE. SA outperformed NE_{52}, NE_{78}, and NE_{104} after about 450 epochs, while DSA did that after about 390 epochs, and was able to achieve much higher MAP. Here DSA clearly outperformed SA in most cases. These results verified that the learned vector representations can be useful in real world applications.

5. Conclusions and Future Work

We propose to use Sequence-to-sequence Autoencoder (SA) and its extension for unsupervised learning of Audio Word2Vec to obtain vector representations for audio segments. We show SA can learn vector representations describing the sequential phonetic structures of the audio segments, and these representations can be useful in real world applications, such as query-by-example STD tested in this preliminary study. For the future work, we are training the SA on larger corpora with different dimensionality in different extensions, and considering other application scenarios.

6. References

[1] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

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

[3] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *arXiv preprint arXiv:1405.4053*, 2014.

[4] 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 *INTERSPEECH*, 2009.

[5] B. Schuller, S. Steidl, and A. Batliner, "The INTERSPEECH 2009 emotion challenge," in *INTERSPEECH*, 2009.

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

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

[8] A. Norouzian, A. Jansen, R. Rose, and S. Thomas, "Exploiting discriminative point process models for spoken term detection," in *INTERSPEECH*, 2012.

[9] K. Levin, K. Henry, A. Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *ASRU*, 2013.

[10] K. Levin, A. Jansen, and B. Van Durme, "Segmental acoustic indexing for zero resource keyword search," in *ICASSP*, 2015.

[11] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *ICASSP*, 2016.

[12] S. Bengio and G. Heigold, "Word embeddings for speech recognition," in *INTERSPEECH*, 2014.

[13] G. Chen, C. Parada, and T. N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," in *ICASSP*, 2015.

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

[15] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," *Unsupervised and Transfer Learning Challenges in Machine Learning, Volume 7*, p. 43, 2012.

[16] J. Li, M.-T. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," in *arXiv preprint arXiv: 1506.01057*, 2015.

[17] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler, "Skip-thought vectors," in *arXiv preprint arXiv: 1506.06726*, 2015.

[18] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using LSTMs," *arXiv preprint arXiv:1502.04681*, 2015.

[19] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *ICML*, 2008.

[20] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *JMLR*, vol. 11, pp. 3371–3408, 2010.

[21] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.

[22] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[23] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez, "Training recurrent networks by evolino," *Neural Computation*, vol. 19, no. 3, pp. 757–779, 2007.

[26] J. Bayer, D. Wierstra, J. Togelius, and J. Schmidhuber, "Evolving memory cell structures for sequence learning," in *ICANN*, 2009.

[27] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH*, 2014.

[28] P. Doetsch, M. Kozielski, and H. Ney, "Fast and robust training of recurrent neural networks for offline handwriting recognition," in *ICFHR*, 2014.

[29] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[30] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *arXiv preprint arXiv:1503.04069*, 2015.

[31] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *ICASSP*, 2015.

[32] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *SciPy*, 2010.

[33] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: new features and speed improvements," *CoRR*, vol. abs/1211.5590, 2012.

[34] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013.

[35] F. Gers, J. Schmidhuber *et al.*, "Recurrent nets that time and count," in *IJCNN*, 2000.

[36] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.

[37] 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 *ICASSP*, 2014.

[38] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, 1978.