INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Query-by-Example Search with Discriminative Neural Acoustic Word Embeddings

Shane Settle[1], *Keith Levin*[2], *Herman Kamper*[1], *Karen Livescu*[1]
[1]TTI-Chicago, USA
[2]Johns Hopkins University, USA
{settle.shane, kamperh, klivescu}@ttic.edu, klevin@jhu.edu

Abstract

Query-by-example search often uses dynamic time warping (DTW) for comparing queries and proposed matching segments. Recent work has shown that comparing speech segments by representing them as fixed-dimensional vectors — acoustic word embeddings — and measuring their vector distance (e.g., cosine distance) can discriminate between words more accurately than DTW-based approaches. We consider an approach to query-by-example search that embeds both the query and database segments according to a neural model, followed by nearest-neighbor search to find the matching segments. Earlier work on embedding-based query-by-example, using template-based acoustic word embeddings, achieved competitive performance. We find that our embeddings, based on recurrent neural networks trained to optimize word discrimination, achieve substantial improvements in performance and run-time efficiency over the previous approaches.
Index Terms: query-by-example, acoustic word embeddings, word discrimination, recurrent neural networks

1. Introduction

Query-by-example speech search (QbE) is the task of searching for a spoken query term (a word or phrase) in a collection of speech recordings. Unlike keyword search and spoken term detection, where the search terms are given as text, QbE involves matching audio segments directly. This task arises naturally when the search terms may be out-of-vocabulary [1, 2], in hands-free settings, or in low- or zero-resource settings [3].

For QbE in high-resource settings, one can train a model to map the audio query to a sequence of subword units, such as phonemes, and search for this sequence in a lattice built from the search collection [2, 4]. This approach requires very significant resources, since it involves much the same process as training a full speech recognition system.

In low-resource settings, typical approaches for this task use dynamic time warping (DTW) to determine the similarity between audio segments. Early approaches to low-resource QbE were based on performing DTW alignment of the query against a search collection either exactly [5, 6] or approximately [7–9].

An alternative to DTW for QbE, which we explore in this paper, is to represent variable-duration speech segments as fixed-dimensional vectors and directly measure similarity between them via a simple vector distance. In this approach, shown in Figure 1, the query is embedded using an *acoustic word embedding* function, producing a vector representation of the query. All potential segments in the search collection are then represented as vectors using the same embedding function. The putative hits (matches) correspond to those segments in the search collection that are closest to the query in the fixed-dimensional embedding space. This type of approach requires preprocessing steps for learning the embedding function and generating the embeddings

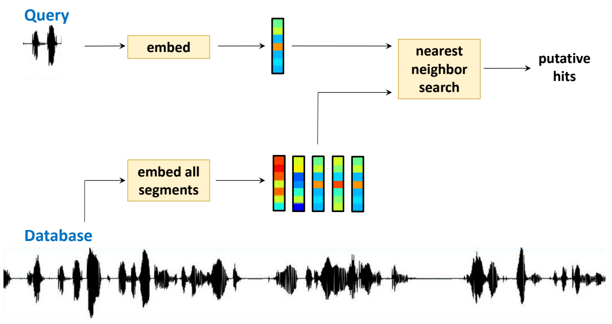

Figure 1: *Embedding-based query-by-example search.*

for the search collection. At test time, efficient approximate nearest-neighbor search can greatly speed up computation.

In prior work, Levin *et al.* [10] used a template-based acoustic word embedding function, and showed that this type of embedding-based QbE search can greatly speed up search compared to a purely DTW-based system, while matching or improving performance. Their template-based embedding approach does not require any labeled supervision. However, in many practical settings, a limited amount of training data might be available. In this work we consider this low-resource setting; in particular, we use acoustic word embeddings based on neural models learned to discriminate between words given a limited (roughly 2-hour) training set.

We build on a growing body of work on neural network-based acoustic word embeddings [11–15]. In several of these studies, neural approaches are shown to far outperform template-based embeddings (such as those used in [10]) on an isolated-word discrimination task, which can be viewed as a proxy for QbE. Here, we use the neural embedding approach of [13], based on Siamese recurrent neural networks, and incorporate these into a complete QbE system using the embedding-based approach of Levin *et al.* [10]. We show that these neural embeddings, trained only on a small amount of labeled data, achieve large improvements in true QbE performance.

2. Neural embedding-based QbE

As illustrated in Figure 1, embedding-based query-by-example (QbE) consists of an embedding method and a nearest neighbor search component. We first describe our neural acoustic word embedding approach, and then give details of the embedding-based QbE search system in which the embeddings are used.

2.1. Neural acoustic word embeddings (NAWEs)

An acoustic word embedding function g maps a variable-length speech segment $Y = y_1, y_2, \ldots, y_T$, where each y_i is an acous-

http://dx.doi.org/10.21437/Interspeech.2017-1592

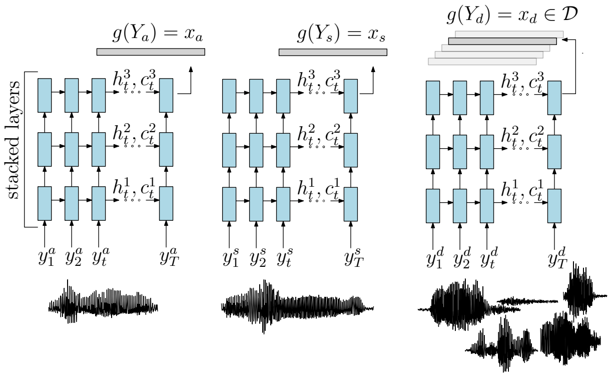
$g(Y_a) = x_a$
$g(Y_s) = x_s$
$g(Y_d) = x_d \in \mathcal{D}$
stacked layers
h_t^3, c_t^3
h_t^2, c_t^2
h_t^1, c_t^1
$y_1^a\ y_2^a\ y_t^a\ y_T^a$
$y_1^s\ y_2^s\ y_t^s\ y_T^s$
$y_1^d\ y_2^d\ y_t^d\ y_T^d$

Figure 2: *Triplet Siamese training setup (unidirectional LSTM depicted for simplicity).*

tic feature frame, to a single embedding vector $x \in \mathbb{R}^d$. Ideally, g should map instances of the same word to nearby vectors, while different words are mapped far apart. Once segments are embedded, they can be compared by computing a vector distance between their embeddings, rather than using DTW.

In [16], Levin *et al.* proposed a template-based approach for the embedding function g. For a target segment, a *reference vector* is defined as the vector of DTW alignment costs to a set of template segments. Dimensionality reduction (based on Laplacian eigenmaps [17]) is then applied to the reference vector to obtain an embedding in $\mathbb{R}^d$. This template-based embedding approach was subsequently used for unsupervised speech recognition in [18] and, more importantly, the full QbE system of [10], which we consider to be a baseline in our experiments.

Recently, neural acoustic word embeddings (NAWEs) have been proposed as an alternative [11–15]. In this recent work, NAWEs have achieved much better performance than the template-based approach, but only in an isolated-word discrimination task that can be seen as a proxy for QbE [19]. Here, we specifically focus on the NAWE approach developed in [13], where it was shown that embeddings based on Long Short-Term Memory (LSTM) [20] networks outperform competing feedforward and convolutional methods. Rather than the proxy task, here we apply these NAWEs in a complete QbE system.

Concretely, we use the concatenation of the hidden representations from a deep bidirectional LSTM network as our embedding function, i.e. $x = g(Y) = [\overrightarrow{h_T}; \overleftarrow{h_1}]$, where $\overrightarrow{h_T}, \overleftarrow{h_1}$ refer to the final hidden state vector from the forward and backward LSTMs, respectively. This LSTM is trained using a Siamese weight-sharing scheme [21] depicted in Figure 2 with a contrastive triplet loss [22,23], $l_{\text{cos hinge}}(Y_a, Y_s)$, defined as

$$\max\left\{0, m + d_{\cos}(x_a, x_s) - \max_{x_d \in \mathcal{D}} d_{\cos}(x_a, x_d)\right\}$$

In this definition, Y_a and Y_s are two segments that have the same word label, and x_a, x_s are their embeddings as output by the neural embedding network. The goal is to push the embeddings x_a and x_s together, until they are closer to each other by a margin m than the embedding x_d of a negative example. Here $d_{\cos}(x_1, x_2) = (1 - \cos(x_1, x_2))$ is the cosine distance between vectors x_1 and x_2. Rather than sampling a single negative example as in [12], or keeping track of confusion statistics as in [13], we sample a set of k embedded segments $\mathcal{D}$ from the whole training set with labels different from Y_a and consider only the example embedding, $x_d \in \mathcal{D}$, that most violates the margin constraint. This improves both performance and rate of convergence on the proxy task.

2.2. Embedding-based QbE

Our system needs to quickly retrieve from a large collection those segments nearest to a given spoken query. For this, we use the Segmental Randomized Acoustic Indexing and Logarithmic-Time Search (S-RAILS) system [10], an embedding-based QbE approach. Although S-RAILS was first applied using the template-based embedding method, it is agnostic to the embedding type, and here we apply it to our neural embeddings.

S-RAILS provides a simple platform for performing approximate nearest neighbor search over vectors, relying on a version of locality-sensitive hashing (LSH) [24, 25]. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_N\} \in \mathbb{R}^d$ be the search collection. LSH is a method for representing vectors in $\mathbb{R}^d$ as bit vectors, referred to as signatures, such that if two vectors $x_i, x_j \in \mathcal{X}$ are close under the cosine distance, then their signatures $s_i, s_j \in \{0,1\}^b$ will agree in most of their entries.

S-RAILS uses LSH to replace the comparatively expensive d-dimensional cosine distance between embeddings with a fast approximation. S-RAILS arranges the signatures $s(\mathcal{X}) = \{s_i : 1 \leq i \leq N\}$ into a lexicographically sorted list $\mathcal{S}$. Given a query vector $q \in \mathbb{R}^d$, we map q to its LSH signature $s = s(q) \in \{0,1\}^b$, and find its location in the sorted signature list $\mathcal{S}$ in $O(\log b)$ time. A set of (approximate) near neighbors to q can be read off this list by looking at the B entries appearing before s and the B entries after s. Bits appearing earlier in the signature have far more influence on whether or not two vectors $x_i, x_j \in \mathcal{X}$ will be judged similar. To ameliorate this effect, S-RAILS performs this lexicographic lookup under P different permutations of the bits. $\pi_1, \pi_2, \ldots, \pi_P \in S_b$,

S-RAILS has three parameters: the signature length b, the beamwidth B, and the number of permutations P. Increasing any of these parameters will tend to improve performance either because it increases the fidelity of our approximation to the cosine distance (in the case of b and P) or because it improves recall (in the case of B). However, any such improvements come at the cost of increased memory required to store the index and the permuted lists (in the case of b and P) and increased runtime (in the case of B and, to a lesser extent, b and P). All told, building the index requires $O(PbN \log N)$ time in the worst case, and querying the index requires $O(B + Pb \log N)$ time.

3. Experimental setup

We use data from the Switchboard corpus of (primarily American) English conversational telephone speech [26]. For training the NAWE model, we use a training set consisting of approximately 10k word segments covering less than 2 hours of speech taken from conversation sides distinct from those used to extract the query set and the evaluation collection. The size of this set is comparable to those used for training in prior work on acoustic word embeddings [13,27,28]. As acoustic features, we use 39-dimensional MFCC+Δ+$\Delta\Delta$s. For QbE, we partition Switchboard into a 37-hour set from which to draw our query terms, a 48-hour development search collection on which to tune parameters of S-RAILS, and a 433-hour evaluation set. These partitions are identical to those used in prior work [8, 10] for the QbE task. We use a set of 43 query words previously used in [8, 10], which were chosen subject to the constraints that the median word duration of each type across the entire corpus is at least 0.5 seconds and the orthographic representation of each word type has at least six characters [8, 10, 16]. Each word type appears 20 to 162 times in the query set, 2 to 188 times in the development search collection, and 39 to 1386 times in the

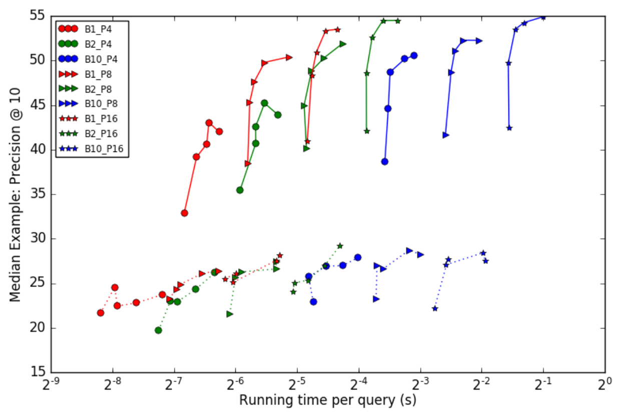

Figure 3: *Performance (median P@10) of S-RAILS (dotted) and S-RAILS+NAWE (solid) on the development search collection. Each sequence of connected points indicates results for a fixed permutation number (P) and beamwidth (B) while signature length (b) is varied from 128 to 2048. In the legend, "Bx_Py" indicates a beamwidth of* $1000x$ *and number of permutations* y.

evaluation set.

For our NAWE model (see Section 2.1), we use a stacked 3-layer bidirectional LSTM with 256 hidden units in each direction; the embeddings produced by the model are therefore 512-dimensional. Dropout is applied with probability 0.3 between LSTM layers. For the margin of the contrastive loss, $l_{\text{cos hinge}}$, we use $m = 0.5$, and we sample $k = 10$ negative instances per anchor segment. We use the Adam optimization algorithm [29] with a batch size of 32, learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \cdot 10^{-8}$. We tuned these parameters based on development set performance on the isolated word discrimination task of [19]. For our QbE evaluation experiments, we trained all models for 100 epochs.

We evaluated the quality of search results according to three commonly used metrics: *figure-of-merit* (FOM), *oracular term weighted value* (OTWV), and *precision at 10* (P@10). FOM is the recall averaged over the ten operating points at which the false alarm rate per hour of search audio is equal to $1, 2, \dots, 10$. OTWV is a query-specific weighted difference between the recall and the false alarm rate (further explanation can be found in [30]). P@10 is the fraction of the ten top-scoring results that are correct matches to the query.

Since the multiple query examples within each query type can have significant variation, we report *average median example* and *average maximum example* scores for each of these three metrics. That is, we compute the median and maximum score over all examples of each query type, and report an unweighted arithmetic mean across the 43 query types.

4. Results

We first present QbE performance on the development data in order to show how performance differs across parameter settings, and then give evaluation results. In this section, we refer to the QbE system that employs the original template-based embeddings simply as S-RAILS, and to the system with neural embeddings as S-RAILS+NAWE.

4.1. Development set performance

Figure 3 shows development set performance, in terms of median P@10, for the baseline QbE system using template-based embeddings (S-RAILS) and our system using NAWEs (S-RAILS+NAWE). Tables 1, 2, and 3 show development set performance for S-RAILS+NAWE as the signature length b, permutations P, and beamwidth B are varied, respectively.

Table 1: *Effect of signature length b on S-RAILS+NAWE performance on the development set, for $P = 16$ permutations and beamwidth $B = 2,000$.*

	Median Example			**Best Example**		
b	**FOM**	**OTWV**	**P@10**	**FOM**	**OTWV**	**P@10**
128	62.1	37.4	42.1	81.7	60.8	83.8
256	67.2	42.6	48.6	83.0	65.4	84.9
512	68.2	44.8	52.6	83.6	65.9	84.9
1024	69.1	46.5	54.5	84.1	66.7	84.8
2048	70.4	48.3	54.5	85.0	66.8	86.0

Table 2: *Effect of number of permutations P on S-RAILS+NAWE performance on the development set, for signature length b =* 1024 *and beamwidth $B = 2,000$.*

	Median Example			**Best Example**		
P	**FOM**	**OTWV**	**P@10**	**FOM**	**OTWV**	**P@10**
4	48.8	33.2	45.2	75.2	59.0	83.0
8	60.9	41.0	50.3	80.3	63.8	85.0
16	69.1	46.5	54.5	84.1	66.7	84.8

Table 3: *Effect of beamwidth B on S-RAILS+NAWE performance on the development set, for signature length b = 1024 and $P = 16$ permutations.*

	Median Example			**Best Example**		
B	**FOM**	**OTWV**	**P@10**	**FOM**	**OTWV**	**P@10**
1000	65.8	44.8	53.4	83.0	65.6	85.0
2000	69.1	46.5	54.5	84.1	66.7	84.8
10000	74.6	49.5	54.2	86.3	67.9	84.8

Figure 3 shows that neural embeddings improve the performance of S-RAILS by large margins at all running time operating points. This figure also shows that increased signature length yields much larger improvements in P@10 for S-RAILS+NAWE than it does for the baseline S-RAILS system. Significant improvements in P@10 can be seen when holding fixed any combination of settings for P and B. Our performance on P@10 saturates with signatures around 1024 bits, while S-RAILS' saturates, for the most part, at 256 bits.

Again in contrast to the S-RAILS system, our method responds strongly to increases in the number of permutations used. In both Figure 3 and Table 2, adjustment to this parameter improves performance consistently across signature lengths. This is to be expected if the neural embeddings provide a better measure of speech segment distances, since the increased number of permutations helps provide a more exact estimate of the embedding distance. We note that performance as measured in Table 2 has not plateaued in any of the Median Example metrics. Further increasing the number of permutations may further improve these metrics, but this incurs a large cost in memory.

Figure 3 and Table 3 show that, except for the cases with short signatures and few permutations, increasing beamwidth does not improve P@10 performance, while incurring significant cost. To obtain higher precision systems, it is more important to use computational resources for increasing the number of permutations or using longer signatures. However, as would be expected, the higher beamwidths help to significantly improve the FOM score, a metric concerned primarily with recall.

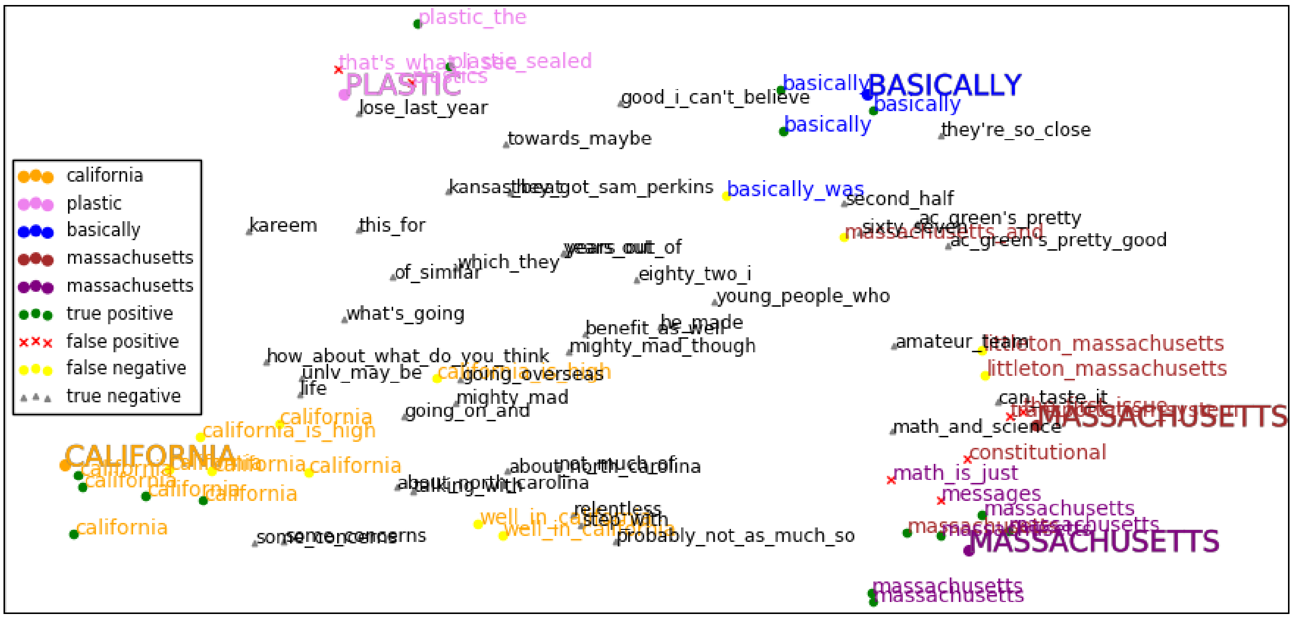

Figure 4: *Embeddings of queries and their top hits, visualized in two dimensions using t-SNE [31]. Queries are shown in capital letters. The top several hits for each query are shown in the same color as the query. Random segments from the search collection and their associated transcriptions are shown in gray.*

Table 4: *Comparison of QbE system performance on the evaluation set.*

	Median Example			**Best Example**			
System	**FOM**	**OTWV**	**P@10**	**FOM**	**OTWV**	**P@10**	**Query time (s)**
RAILS [8]	6.7	2.7	44.0	20.7	10.4	84.4	24.7
S-RAILS (baseline)	24.5	14.4	34.5	46.2	26.6	87.4	0.078
S-RAILS+NAWE (ours)	43.3	22.4	60.2	65.4	43.3	95.1	0.38

4.2. Evaluation set performance

Based on development results, we find that an operating point of 16 permutations, beamwidth of 2000, and signature length of 1024 is close to optimal, in terms of both performance and query speed, for both the baseline S-RAILS and S-RAILS+NAWE. We use these settings for final evaluation. For a qualitative view, Figure 4 visualizes several queries and their top hits in the evaluation collection. This visualization shows some expected properties. For example, the two “Massachusetts” queries and their top hits are embedded close together. Two of the false alarms for “Massachusetts” are the similar-sounding “messages” and “math is just”, while the somewhat more distant “math and science” is (correctly) not retrieved.

Final evaluation performance is shown in Table 4. Besides the S-RAILS baseline, we also compare to RAILS [8], a DTW-based system that is optimized for speed using LSH to get approximate frame-level near neighbor matches. RAILS evaluation scores are reproduced from [8]. We find that our approach improves significantly over both RAILS and S-RAILS in terms of all performance metrics at this operating point. Note that, based on Figure 3, the improvements should hold at most operating points, including ones with much higher query speeds. The biggest gains from S-RAILS+NAWE are seen in the Median Example results, where there is a relative improvement over S-RAILS of more than 55% across all measures. In terms of FOM and OTWV, we see relative improvements of over 40% in the Best Example case. Although the baselines obtain good P@10, we still find large improvements in this measure as well, from 87.1% to 95.1%.

5. Conclusion

We have presented an approach to query-by-examaple speech search using neural acoustic word embeddings, demonstrating the ability of these embedding models to improve over previous methods on a realistic task. The neural embeddings are learned from a very limited set of data; one interesting future direction is to study the limits of the approach as the amount of training data is varied, or to extend it to use no labeled data at all. Another interesting aspect of the approach is that the neural embeddings are learned from speech segments that have been pre-segmented at word boundaries, but they are then applied for embedding arbitrary segments that may or may not (and usually do not) correspond to words. It is encouraging that this approach works despite the lack of non-word examples in the training data, and an interesting avenue for future work is to attempt to further improve performance by explicitly training on both word and non-word segments. Additional future directions include training a QbE system end-to-end and extending our model to operate at the level of multi-word phrases.

6. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1433485 and by a Google faculty award.

7. References

[1] W. Shen, C. M. White, and T. J. Hazen, “A comparison of query-by-example methods for spoken term detection,” DTIC Document, MIT Lincoln Labs, Tech. Rep., 2009.

[2] C. Parada, A. Sethy, and B. Ramabhadran, “Query-by-example spoken term detection for oov terms,” in *Proc. ASRU*, 2009.

[3] I. Szöke, L. J. Rodríguez-Fuentes, A. Buzo, X. Anguera, F. Metze, J. Proenca, M. Lojka, and X. Xiong, “Query by example search on speech at mediaEval 2015.” in *Proc. MediaEval*, 2015.

[4] C. Allauzen, M. Mohri, and M. Saraclar, “General indexation of weighted automata: application to spoken utterance retrieval,” in *Proc. HLT-NAACL*, 2004.

[5] T. J. Hazen, W. Shen, and C. White, “Query-by-example spoken term detection using phonetic posteriorgram templates,” in *Proc. ASRU*, 2009.

[6] Y. Zhang and J. R. Glass, “Unsupervised spoken keyword spotting via segmental dtw on gaussian posteriorgrams,” in *Proc. ASRU*, 2009.

[7] ——, “A piecewise aggregate approximation lower-bound estimate for posteriorgram-based dynamic time warping.” in *Proc. Interspeech*, 2011.

[8] A. Jansen and B. Van Durme, “Indexing raw acoustic features for scalable zero resource search,” in *Proc. Interspeech*, 2012.

[9] G. Mantena and X. Anguera, “Speed improvements to information retrieval-based dynamic time warping using hierarchical k-means clustering,” in *Proc. ICASSP*, 2013.

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

[11] Y.-A. Chung, C.-C. Wu, C.-H. Shen, and H.-Y. Lee, “Unsupervised learning of audio segment representations using sequence-to-sequence recurrent neural networks,” in *Proc. Interspeech*, 2016.

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

[13] S. Settle and K. Livescu, “Discriminative acoustic word embeddings: Recurrent neural network-based approaches,” in *Proc. SLT*, 2016.

[14] W. He, W. Wang, and K. Livescu, “Multi-view recurrent neural acoustic word embeddings,” in *Proc. ICLR*, 2017.

[15] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, “End-to-end ASR-free keyword search from speech,” in *Proc. ICASSP*, 2017.

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

[17] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.

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

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

[20] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, “Signature verification using a “Siamese” time delay neural network,” *Int. J. Pattern Rec.*, vol. 7, no. 4, pp. 669–688, 1993.

[22] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” in *Proc. CVPR*, 2005.

[23] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng, “Grounded compositional semantics for finding and describing images with sentences,” *Trans. ACL*, vol. 2, pp. 207–218, 2014.

[24] P. Indyk and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” in *Proc. STOC*, 1998.

[25] M. Charikar, “Similarity estimation techniques from rounding algorithms,” in *Proc. STOC*, 2002.

[26] J. J. Godfrey, E. C. Holliman, and J. McDaniel, “SWITCHBOARD: Telephone speech corpus for research and development,” in *Proc. ICASSP*, 1992.

[27] H. Kamper, M. Elsner, A. Jansen, and S. J. Goldwater, “Unsupervised neural network based feature extraction using weak top-down constraints,” in *Proc. ICASSP*, 2015.

[28] A. Jansen, S. Thomas, and H. Hermansky, “Weak top-down constraints for unsupervised acoustic model training,” in *Proc. ICASSP*, 2013.

[29] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. ICLR*, 2014.

[30] D. R. H. Miller, M. Kleber, C. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz, and H. Gish, “Rapid and accurate spoken term detection,” in *Proc. Interspeech*, 2007.

[31] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.