# SEGMENTAL ACOUSTIC INDEXING FOR ZERO RESOURCE KEYWORD SEARCH

*Keith Levin, Aren Jansen, Benjamin Van Durme*

Human Language Technology Center of Excellence, Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218

## ABSTRACT

The task of zero resource query-by-example keyword search has received much attention in recent years as the speech technology needs of the developing world grow. These systems traditionally rely upon dynamic time warping (DTW) based retrieval algorithms with runtimes that are linear in the size of the search collection. As a result, their scalability substantially lags that of their supervised counterparts, which take advantage of efficient word-based indices. In this paper, we present a novel audio indexing approach called Segmental Randomized Acoustic Indexing and Logarithmic-time Search (S-RAILS). S-RAILS generalizes the original frame-based RAILS methodology to word-scale segments by exploiting a recently proposed acoustic segment embedding technique. By indexing word-scale segments directly, we avoid higher cost frame-based processing of RAILS while taking advantage of the improved lexical discrimination of the embeddings. Using the same conversational telephone speech benchmark, we demonstrate major improvements in both speed and accuracy over the original RAILS system.

***Index Terms*—** Zero resource, query-by-example search, speech indexing, fixed-dimensional embedding

## 1. INTRODUCTION

Keyword search, in which one must locate occurrences of an utterance in a collection of speech audio, has received increasing attention as speech data becomes ever more ubiquitous. Most approaches to date have employed lattice indexing techniques [1], enabling search of thousands of hours of speech in interactive time. Typical systems build a model to map sequences of frames to segmental units (e.g., phones or words) that are more amenable to standard lattice-based approaches. Unfortunately, these techniques require large collections of annotated speech audio, which are unavailable in most languages. As a result, the zero-resource setting, in which detailed annotations are unavailable and linguistic structure must be discovered without the aid of training data, has attracted attention both in the speech processing community [2] and among scientists interested in human language acquisition [3].

Query-by-example search, where search terms are presented as audio segments rather than in graphemic or phonetic form, has applications in probing large collections of unstructured audio data [4] and in voice interfaces [5]. The standard approach involves training a model to map query audio to a sequence of symbols (e.g., a phonetic representation) and searching for this sequence in a lattice built on the search collection [6]. Finite state automata techniques have made lattice search of this kind both fast and accurate [7], but the nature of the required training data makes these approaches infeasible in zero- and low-resource settings.

Dynamic time warping (DTW) has been effective in zero-resource query-by-example search [8, 9, 10]. Unfortunately, DTW sequence alignment requires time linear in the size of the search collection, which limits its scalability. While techniques such as those in [11, 12] have improved runtime by reducing the constants in this linear dependence, the Randomized Acoustic Indexing and Logarithmic-Time Search (RAILS) system introduced in [13] avoids this linear dependence altogether. RAILS operates by performing logarithmic-time approximate nearest-neighbor retrieval at the frame level to find likely matches of the query against the search collection. Subsequent processing extends frame-level matches to segmental-level matches of the query against the search collection using image processing techniques.

RAILS has two main limitations. First, its accuracy depends ultimately on DTW as a measure of segment-level similarity. Second, the process by which frame-level matches are extended requires computationally expensive digital image processing, which introduces a major runtime bottleneck. In the current work, we present S-RAILS, an extension of the RAILS methodology that avoids both of these shortcoming by performing search directly at the segment level using fixed-dimensional segmental embeddings presented in [14]. Such embedding techniques showed a marked improvement over a purely DTW-based approach as measured by performance on the evaluation task introduced in [15], and since search is performed at the segment level already, there is no need for extending frame-level matches as in the original RAILS system. We evaluate S-RAILS in a query-by-example keyword search task on a corpus of telephone speech, in which our system improves dramatically over the original RAILS system in both accuracy and runtime.

## 2. METHODS

The S-RAILS system is an adaptation of the RAILS query-by-example search system presented in [13]. In RAILS, indexing consists of building a structure to support fast approximate nearest-neighbor retrieval at the frame level using the point location in equal balls (PLEB) algorithm [16]. Given a query, the near neighbors of each frame in the query are retrieved from the index along with scores reflecting their similarity. These near neighbor frames along with their scores yield a sparse approximation to the frame-level similarity matrix, the entries of which correspond to similarities between frames in the query and frames in the search collection. Segments of the search audio that are similar to the query give rise to approximately diagonal lines in the similarity matrix. These diagonal lines appear as peaks in the Hough transform of the matrix, and thus can be quickly located.

S-RAILS differs from the original RAILS system by indexing the acoustic features of whole word-sized segments directly, altogether avoiding both the intermediate step of frame-level indexing and the need to construct a similarity matrix. It operates as follows:

1. Voice activity detection (VAD) locates regions likely to contain speech.

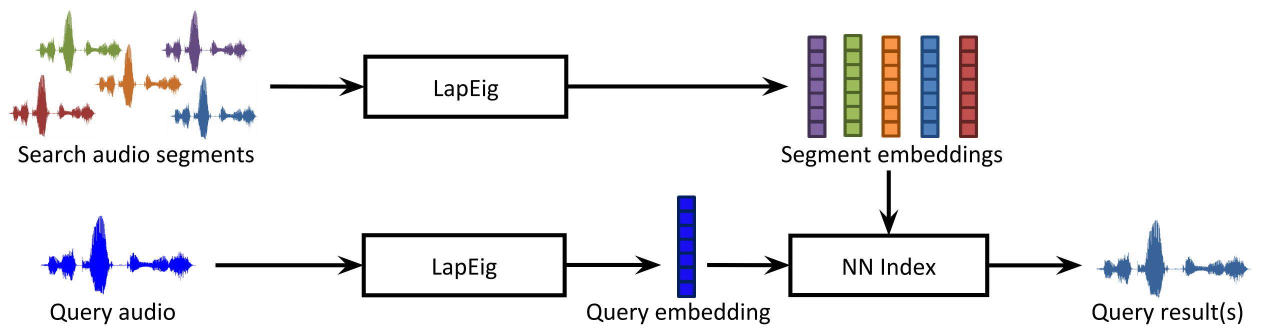

**Fig. 1**. Diagram of the S-RAILS audio search system.

2. Each VAD region is split into overlapping segments from some minimum duration to some maximum duration. Each segment is mapped to a fixed-dimensional vector using techniques from [14].
3. An index is constructed for randomized approximate nearest-neighbor retrieval [16] on the collection of fixed-dimensional embeddings. Each segment created in the previous step appears as an entry in the index.
4. At query time, a query segment is mapped to its fixed-dimensional representation and the near-neighbors of that representation are retrieved from the index.
5. Candidate matches to a query can be rescored after retrieval, e.g., by computing exact DTW scores as in [13].

## 2.1. Fixed-dimensional Segment Embeddings

To obtain fixed-dimensional representations of speech segments, we use the unsupervised Laplacian eigenmaps embedding described in Section 2.4 of [14], which we summarize here. Laplacian eigenmaps is a non-linear dimensionality reduction technique that maps a collection of objects into Euclidean space in such a way that the local geometry of the collection is preserved. The Laplacian eigenmaps framework is described in detail in [17], with an out-of-sample extension described in [18].

Let $\mathcal{X}$ be the set of all arbitrary-length feature vector time series, $\mathcal{X} = \{X = x_1 x_2, \dots, x_T : T \in \mathbb{Z}^+\}$, where each $x_i \in \mathbb{R}^p$ and $p$ is the dimensionality of a speech frame. Our goal is to find a mapping $h : \mathcal{X} \to \mathbb{R}^d$, where $d$ is the dimensionality of our embeddings, such that speech segments from $\mathcal{X}$ with similar content are mapped to nearby locations in $\mathbb{R}^d$. We are given a collection $\mathcal{Y} = \{X_1, X_2, \dots, X_n\} \subset \mathcal{X}$ from which to learn this mapping.

We construct a $k$-nearest neighbor graph with nodes corresponding to elements from $\mathcal{Y}$, measuring the distances between segments by their DTW alignment costs. The graph is represented by a binary-valued adjacency matrix $A \in \mathbb{R}^{n \times n}$, with $A_{ij} = 1$ if and only if $X_i$ is among the $k$ nearest neighbors of $X_j$ or vice versa. This allows us to construct the normalized graph Laplacian of $A$, $L = I - D^{-1/2} A D^{-1/2}$, where $D$ is a diagonal matrix with $D_{ii} = \sum_j A_{ij}$. The Laplacian eigenmaps out-of-sample extension, presented in [18], finds a set of projection maps $\{h_1, h_2, \dots, h_d\}$, with $h_j : \mathcal{X} \to \mathbb{R}$ for $j = 1, 2, \dots, d$, which are determined by the solutions to

$$h^* = \arg \min_{\substack{h \in \mathcal{H}_K, \\ \mathbf{h}^T\mathbf{h}=1, \\ \mathbf{h}^T\vec{\mathbf{1}}=0}} \mathbf{h}^T L \mathbf{h} + \xi \|\mathbf{h}\|_K^2, \tag{1}$$

where $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive semi-definite kernel function, $\mathcal{H}_K$ is the reproducing kernel Hilbert space for $K$, $\mathbf{h}$ is the vector with $\mathbf{h}_i = h(X_i)$, and $\xi \geq 0$ is a regularization parameter. Our $j$-th projection map applied to segment $X \in \mathcal{X}$ is then given by

$$h_j^*(X) = \sum_{i=1}^{n} \alpha_i^{(j)} K(X_i, X), \tag{2}$$

where the $\alpha_i^{(j)}$ terms are the solutions to the generalized eigenvector problem $(LK + \xi I)\alpha = \lambda K \alpha$, where $K_{ij} = K(X_i, X_j)$. As in [14], we use a kernel function given by

$$K(X_i, X_j) = \exp\left\{ \frac{-\max\{0, \mathrm{DTW}(X_i, X_j) - \eta\}}{2\sigma^2} \right\},$$

where $\mathrm{DTW}(X_i, X_j)$ denotes the DTW alignment cost of segment $X_i$ with segment $X_j$, and $\eta, \sigma \in \mathbb{R}$ with $\sigma > 0$. We measure distance between embeddings using cosine distance, since [14] shows this to yield good results on a word discrimination task.

## 2.2. Near-neighbor retrieval

A crucial step in both RAILS and S-RAILS consists of retrieving a set of embeddings that are similar to a query embedding. Our goal is to build an index which, given a query vector, returns vectors from the index that are near to the query vector under cosine distance. To solve this problem, RAILS used an implementation of point location in equal balls (PLEB) as presented in [16]. PLEB makes use of locality sensitive hash (LSH) functions, which capture the geometric proximity of pairs of items in the sense that nearby items are likely to be hashed to the same value and distant items are unlikely to be hashed to the same value.

The LSH variant used here is the same as that used in the original RAILS system [13]. We map vectors to binary strings of length $S$, which we call *signatures*. This mapping is chosen such that cosine distance between two vectors can be approximated by some function of the Hamming distance between their respective signatures. These signatures are generated by randomly choosing a set of $S$ hyperplanes through the origin in the vector space. Each bit of a vector's signature is determined by which side of a corresponding hyperplane it falls on. Pairs of vectors with small cosine distance are unlikely to be separated by a randomly-chosen hyperplane, and thus their signatures are likely to be similar. This permits fast retrieval of the approximate near neighbors of a given query vector by computing its signature and returning all vectors from the search collection whose signatures are at a small Hamming distance from it.

The near-neighbor retrieval algorithm used in S-RAILS is discussed in detail in [19], and we summarize it here. We let $B$ denote the *beamwidth*, a parameter that controls the number of near

**Table 1**. S-RAILS performance on the *development* search collection, averaged over all query types as a function of signature length $S$ for fixed number of permutations $P = 8$ and beamwidth $B = 10,000$. All scores are percentages.

| | Median Example | | | Best Example | | |
|---|---|---|---|---|---|---|
| S | FOM | OTWV | P@10 | FOM | OTWV | P@10 |
| 64 | 22.3 | 9.8 | 9.1 | 48.7 | 26.2 | 45.4 |
| 128 | 27.5 | 11.4 | 11.4 | 56.0 | 30.4 | 55.1 |
| 512 | 30.4 | 14.0 | 14.4 | 57.7 | 33.8 | 59.1 |
| 1024 | 30.2 | 13.9 | 14.8 | 58.3 | 35.0 | 60.7 |

**Table 2**. S-RAILS performance on the *development* search collection, averaged over all query types as a function of number permutations $P$ for fixed beamwidth $B = 100,000$ and signature length $S = 512$. All scores are percentages.

| | Median Example | | | Best Example | | |
|---|---|---|---|---|---|---|
| P | FOM | OTWV | P@10 | FOM | OTWV | P@10 |
| 4 | 31.3 | 13.6 | 15.2 | 60.7 | 34.1 | 58.7 |
| 8 | 33.1 | 14.5 | 15.4 | 63.0 | 35.2 | 59.6 |

neighbors that we retrieve. Retrieval is performed by sorting the signatures in the search collection and returning signatures that share a prefix with the query signature. Given a collection of signatures $\mathcal{Z} = \{z_1, z_2, \ldots, z_N\}$ with each $z_i \in \{0,1\}^S$, we sort the elements of $\mathcal{Z}$ in lexicographic order. Let $\pi$ be a permutation of the integers $1, 2, \ldots, N$ such that $z_{\pi(1)}, z_{\pi(2)}, \ldots, z_{\pi(N)}$ is the lexicographic sort of the elements of $\mathcal{Z}$. Given a query signature $q \in \{0,1\}^S$, we find via binary search the location where $q$ belongs in the sorted list and return the $B$ signatures before that position and the $B$ signatures after that position. That is, if $q$ belongs between $z_{\pi(i)}$ and $z_{\pi(i+1)}$ in the sorted list, we return the set $\{z_{\pi(a)}, z_{\pi(a+1)}, \ldots, z_{\pi(b)}\}$, where $a = \max\{1, i - B + 1\}$ and $b = \min\{N, i + B\}$. Of course, in this lexicographic sorting scheme, a given ordering of the signature bits means that bits appearing early in the signature have a greater influence over which pairs of signatures are considered similar. This problem is mitigated by performing several of these searches under different permutations of the signature bit ordering. We denote by $P$ the number of such permutations that we use. In practice, rather than repeatedly permuting and sorting the signature list, we keep $P$ separate lists of the search collection signatures, each sorted according to a different one of the $P$ permutations. Retrieval of near-neighbors under this scheme requires time logarithmic in $N$ and linear in both $P$ and $S$. We have observed in our experiments that runtime depends only weakly on $S$ compared to dependence on $P$ and $N$.

## 3. EXPERIMENTS

Our experiments follow those presented in [13]. We evaluated our system in a query-by-example keyword search task on the Switchboard corpus, a collection of conversational telephone speech. A 37-hour collection was set aside from which to draw query terms, a 48-hour development search collection was used to explore the effect of different parameters on the system's performance, and a 433-hour evaluation set was used to obtain final performance metrics. Query word types were chosen to have corpus-wide median duration of at least 0.5 seconds and orthographic representation at least six characters long. This resulted in a collection of 43 query word types:

absolutely basically benefit bottles business California college community companies control crimes definitely deterrent employees expenses expensive important individual insurance interesting mandatory Massachusetts newspaper organization performance plastic policy positive process program punishment recently recycle recycling retirement salary savings situation society understand unfortunately university vacation

Each query type appeared between 20 and 162 times in the query set, between 2 and 188 times in the development search collection, and between 39 and 1386 times in the evaluation collection. More than half of the selected query types had median duration less than 0.55 s and all query types had median duration less than 0.75 s. We considered three common keyword search metrics:

(1) *Figure-of-merit* (FOM), the average recall over the 10 operating points at which the false alarm rate is $1, 2, \ldots, 10$ false alarms per hour of search audio.

(2) *Oracular term weighted value* (OTWV), a weighted difference between the system's recall and false alarm rate. The oracular variant of this metric assumes an optimal query-specific threshold. See [1] for a detailed account of this metric.

(3) *Precision at 10* (P@10), the fraction of the top ten ranked candidate matches that are correct.

Metrics were computed separately for each query type, and are reported as unweighted averages over all 43 query types. Performance is sensitive to the specific query example. Thus, for each metric, we report both (i) the median query example performance, and (ii) the best query example performance.

### 3.1. Selecting Index Parameters

Table 1 shows the effect of signature length on system performance for fixed beamwidth $B = 10,000$ and number of permutations $P = 8$. Performance saturates at a signature length of 512 bits. These signatures are larger than the 64-bit signatures used in RAILS owing to the fact that RAILS indexes 39-dimensional feature vectors while S-RAILS indexes 1000-dimensional fixed-dimensional embeddings. As a result, a larger number of bits are required to achieve suitably high fidelity in approximating cosine distance between vectors. Table 2 shows system performance as a function of the number of permutations for fixed beamwidth $B = 100,000$ and signature length $S = 512$. We see that $P = 8$ yields a non-negligible performance gain over $P = 4$ in the best-example case, though median performance is largely insensitive to $P$. These two tables jointly suggest that performance saturates at a signature length of 512 bits and $P = 8$. We use these parameters in the remainder of our evaluation.

### 3.2. Constructing the Index

To segment the search collection, candidate segment boundaries were placed at 3-frame intervals in all VAD regions. Resulting segments with duration at least 40 frames (400 ms) and at most 100 frames (1 s) were included in the index. To construct Laplacian eigenmaps embeddings, we used a set of 10,383 unlabeled word examples from the Switchboard corpus to define our similarity graph. As discussed in [14], the process of constructing Laplacian eigenmaps embeddings is slow, since a single embedding requires computing a DTW alignment of a segment with every segment in the similarity graph. Indeed, this process is currently the major bottleneck in constructing an index. In order to speed up the embedding process, rather than explicitly computing $\mathrm{DTW}(X, X_i)$ for all $i$ as in (2), we performed a spectral clustering of the 10,383-segment similarity graph and selected a representative segment (the medoid)

**Table 3**. S-RAILS performance on the evaluation search set, averaged over all query types as a function of beam width $B$ for fixed number of permutations $P = 8$ and signature length $S = 512$. All scores are percentages except Real Time Speedup, which is the ratio of search collection duration to the average time required to perform a single query.

| | **Median Example** | | | **Best Example** | | | |
|---|---|---|---|---|---|---|---|
| B | **FOM** | **OTWV** | **P@10** | **FOM** | **OTWV** | **P@10** | **Real Time Speedup** |
| 100 | 7.6 | 6.0 | 39.3 | 19.8 | 15.5 | 85.3 | 307,000,000 |
| 1,000 | 15.0 | 9.7 | 38.3 | 34.1 | 21.8 | 87.4 | 40,800,000 |
| 10,000 | 26.0 | 12.7 | 38.6 | 47.7 | 26.3 | 91.6 | 5,770,000 |
| 100,000 | 37.3 | 15.1 | 38.6 | 56.9 | 29.6 | 89.3 | 510,000 |

**Table 4**. Baseline RAILS performance (reproduced from [13]) on the *evaluation* search set averaged over all query types as a function of beam width $B$. All scores are percentages except Real Time Speedup, which is the ratio of search collection duration to the average time required to perform a single query.

| | **Median Example** | | | **Best Example** | | | |
|---|---|---|---|---|---|---|---|
| B | **FOM** | **OTWV** | **P@10** | **FOM** | **OTWV** | **P@10** | **Real Time Speedup** |
| 500 | 0.8 | 0.9 | 21.0 | 3.6 | 2.8 | 58.4 | 620,000 |
| 5,000 | 6.7 | 2.7 | 44.0 | 20.7 | 10.4 | 84.4 | 63,000 |
| 50,000 | 19.0 | 4.7 | 49.2 | 39.9 | 16.5 | 88.4 | 7,000 |
| 100,000 | 20.2 | 4.8 | 49.8 | 41.1 | 16.6 | 88.1 | 3,600 |

from each cluster. Given a segment $X \in \mathcal{X}$ to embed, its DTW alignment was computed with each cluster representative. For representatives whose alignment cost was above some threshold, we set $K(X, X_i) = 0$ for all $X_i$ in the corresponding cluster rather than computing exact alignment costs. Experiments showed that 550 clusters with a threshold of 0.17 yielded a very good approximation to the true values of the kernel function. This approximation yielded a factor of 6 speedup with respect to the exact computation, but even with this speedup, computing fixed-dimensional embeddings of speech audio is approximately 130 times slower than real time on current hardware. This process produced approximately 30 million 1,000-dimensional embeddings in the case of the development search collection and approximately 280 million in the case of the evaluation search collection, which became the input to the index.

By the nature of the Laplacian eigenmaps embedding, word examples that are not similar to any words in the reference set are mapped to locations near the origin. At query time, when similarity search is performed under cosine distance, many of these small-norm embeddings are retrieved as candidate matches. This results in many false positives, reflected in the low median precision at 10 scores in Tables 1 and 2. To reduce this effect, we removed from the index all embeddings with norm less than a set threshold. We found a threshold of 0.06 to be best, though performance was comparatively flat for thresholds between 0.01 and 0.1. In experiments on the development search set, this resulted in 50% to 70% relative improvements in median precision at 10, as well approximately 8% relative improvement in maximum precision at 10 and, somewhat surprisingly, small improvements on all other metrics.

### 3.3. Post-processing of query results

Owing to the segmentation scheme used in S-RAILS, the index contains many entries corresponding to overlapping segments, and our embedding technique causes these segments to be mapped to similar fixed-dimensional vectors. The result is that at query time, if one of these segments is retrieved, many other overlapping segments are likely to be retrieved, as well. To eliminate this redundancy, we performed a post-processing step in which retrieved segments whose midpoints were within a given number of frames of one another were greedily merged by discarding the segment with lower score. This operation was repeated until no further merge operations could be performed. We found that merging pairs of segments whose midpoints were within 10 frames of one another proved effective.

### 3.4. Results

Table 3 shows system performance on the evaluation search set as a function of beamwidth for fixed number of permutations $P = 8$ and signature length $S = 512$. Table 4 shows performance of the original RAILS system for comparison. We note that values of $B$ in RAILS and S-RAILS are not directly comparable, since the two systems operate on different objects, though both systems' runtimes depend linearly on the parameter. Comparing the best performance of the two systems, we see that S-RAILS achieves more than 80% relative improvement over RAILS in median example FOM and upwards of 200% relative improvement in median example OTWV. In the case of best example performance, S-RAILS exhibits approximately 78% relative improvement in OTWV and 38% relative improvement in FOM performance. P@10 scores are less decisive. S-RAILS improves marginally on RAILS in best example P@10, but lags by a non-negligible margin in median example P@10. As alluded to previously, this is due to a small number of particularly high-scoring false alarms introduced by the embedding process. This issue might be ameliorated by a suitable rescoring procedure.

Comparing system runtimes paints a more impressive picture. S-RAILS tends to achieve a speedup of between two and five orders of magnitude with respect to RAILS at any given performance level. To take a particularly striking example, S-RAILS with $B = 100$ achieves better median OTWV performance than RAILS with $B = 200,000$ while running more than 85,000 times faster.

## 4. CONCLUSION

We have presented a novel segment-based indexing and retrieval method for query-by-example search that operates at scale in the zero-resource setting. Our approach makes substantial improvements relative to the accuracy and query processing time of the RAILS system presented in [13]. Directions for future work include exploring how selection of elements for use in the Laplacian eigenmaps similarity graph effects system performance, exploring methods to further speed up the construction of fixed-dimensional embeddings, and rescoring candidate matches retrieved by S-RAILS.

## 5. REFERENCES

[1] 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.

[2] Jim Glass, "Towards unsupervised speech processing," in *Proc. ISSPA*, 2012.

[3] A. Jansen, E. Dupoux, S. Goldwater, M. Johnson, S. Khudanpur, et al., "A summary of the 2012 CLSP workshop on zero resource speech technologies and models of early language acquisition," in *Proc. ICASSP*, 2013.

[4] X. Anguera, F. Metze, A. Buzo, I. Szöke, and L. J. Rodriguez-Fuentes, "The spoken web search task," in *Proc. MediaEval 2013 Multimedia Benchmark Workshop*, 2013.

[5] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Proc. ICASSP*, 2014.

[6] C. Parada, A. Sethy, and B. Ramabhadran, "Query-by-example spoken term detection for OOV terms," in *Proc. ASRU*, 2009.

[7] C. Allauzen, M. Mohri, and M. Saraclar, "General indexation of weighted automata– application to spoken utterance retrieval," in *Proc. workshop on interdisciplinary approaches to speech indexing and retrieval at HLT-NAACL*, 2004.

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

[9] A. Jansen, K. Church, and H. Hermansky, "Towards spoken term discovery at scale with zero resources," in *Proc. Interspeech*, 2010.

[10] X. Anguera and M. Ferrarons, "Memory efficient subsequence dtw for query-by-example spoken term detection," in *Proc. ICME*, 2013.

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

[12] Y. Zhang and J. Glass, "A piecewise aggregate approximation lower-bound estimate for posteriorgram-based dynamic time warping," in *Proc. Interspeech*, 2011.

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

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

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

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

[17] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Represenation," *Neural Computation*, vol. 16, pp. 1373–1396, 2003.

[18] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold Regularization: A Geometric Framework for Learning from Examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[19] A. Jansen and B. Van Durme, "Efficient spoken term discovery using randomized algorithms," in *Proc. ASRU*, 2011.