INTERSPEECH 2012
ISCA's 13th Annual Conference
Portland, OR, USA
September 9-13, 2012

Indexing Raw Acoustic Features for Scalable Zero Resource Search

Aren Jansen, Benjamin Van Durme

Human Language Technology Center of Excellence, Center for Language and Speech Processing
Johns Hopkins University, Baltimore, Maryland
aren@jhu.edu, vandurme@cs.jhu.edu

Abstract

We present a new speech indexing and search scheme called Randomized Acoustic Indexing and Logarithmic-time Search (RAILS) that enables *scalable* query-by-example spoken term detection in the zero resource regime. RAILS is derived from our recent investigation into the application of randomized hashing and approximate nearest neighbor search algorithms to raw acoustic features. Our approach permits an approximate search through hundreds of hours of speech audio in a matter of seconds, and may be applied to any language without the need of a training corpus, acoustic model, or pronunciation lexicon. The fidelity of the approximation is controlled through a small number of easily interpretable parameters that allow a trade-off between search accuracy and speed.

Index Terms: speech indexing, zero resource, query-by-example, spoken term detection, locality sensitive hashing

1. Introduction

Recent advances in fast spoken term detection (STD) technologies have relied on efficient lattice indexing methods, often based on finite state machines (FSM), to permit searches over hundreds of hours of speech audio in interactive time [1, 2]. These methods are typically made possible by phone, fragment, and/or word based recognizers that map each short-time acoustic feature vector time series to a hypothesized set of symbolic sequences of linguistic units that are more amenable to index construction. Even though the queries may be limited to a relatively small set of contentful terms, these recognizers attempt to interpret the speech in terms of the entire phonetic and lexical content of the language, and thus require large orthographically transcribed training corpora and pronunciation lexicons.

Meanwhile, there have been several recent efforts in query-by-example STD [3, 4, 5], where the search query is provided in the form of spoken example(s) as opposed to graphemic or phonetic form. The prevailing approach applies phonetic recognizers to the query and identifies regions of the search collection with phonetic content that is consistent. As in traditional STD, the query-by-example variant has also been implemented via FSM-based indexing with substantial success both in terms of detection accuracy and search speed [4]. However, such approaches still rely on access to in-language training material to train the phonetic recognizer. If we instead desire true language-independence for the zero resource regime, we must fall back on pattern matching schemes applied to the raw acoustics.

More recently, there have been ambitious efforts to build *zero resource* query-by-example search systems [6, 7, 8], including a multi-institution system evaluation [9]. The prevailing methods rely on dynamic time warping (DTW) based search algorithms applied to either raw acoustic features or unsupervised acoustic model posteriorgrams. Even with substantial optimization [10], these methods are universally linear time in the size of the search collection and several orders of magnitude slower than their logarithmic-time, index-based, supervised counterparts. Building on our recent exploration into the use of randomized hashing and approximate nearest neighbor search for speech audio [11], we propose a novel speech indexing and search scheme called Randomized Acoustic Indexing and Logarithmic-time Search (RAILS) for the task of query-by-example STD. At the core of RAILS, locality sensitive hashing [12] is used to map each feature vector to a sortable bit signature that can be recovered from a list of n frames in $O(\log n)$ time using a standard binary search. Combined with DTW-inspired methods to identify runs of frame-level matches, the overall search complexity is $O(m \log n)$, where m is the length of the query (in frames). The run time constants are set by the desired approximation fidelity, allowing a measured trade-off between search accuracy and real time factor.

2. Algorithms

In the query-by-example spoken term detection task we are presented with (i) a search collection, which we denote by the feature vector time series $X = x_1 \dots x_n$, where each $x_j \in \mathbb{R}^d$, and (ii) one or more spoken queries, each represented by a relatively short feature vector time series of the form $Q = q_1 \dots q_m$, where each $q_i \in \mathbb{R}^d$. The goal is to find all subseries of X that contain an occurrence of the keyword or keyterm uttered in Q. The RAILS approach consists of three steps. First, we map each feature vector sample of both the search collection and query to a fixed-length bit signature using locality sensitive hashing. The search collection signatures are sorted lexicographically and stored in the RAILS index to facilitate subsequent search. Second, we use the RAILS index to construct for each query frame-level nearest neighbors sets over the entire collection X in $O(\log n)$ time. Repeating this process for all query frames, it follows that we can construct in $O(m \log n)$ time a sparse approximate cosine similarity matrix $M \in \mathbb{R}^{m \times n}$ between Q and X of the form $M_{ij} \approx (q_i \cdot x_j)/(\|q_i\|\|x_j\|)$. Lastly, we use these similarity matrices to search for runs of frames in the search collection that match the given query using a two-pass, coarse-to-fine strategy made possible by sparse image processing techniques. For easy reference, Table 1 lists all parameters involved along with the default values used in the experiments described in Section 4 (except where noted).

2.1. Locality Sensitive Hashing

We apply locality sensitive hashing (LSH) to both Q and X in exactly the same manner used in [11] and provide a brief description here for completeness. LSH is a method of mapping high dimensional feature vectors, $x \in \mathbb{R}^d$, into low dimen-

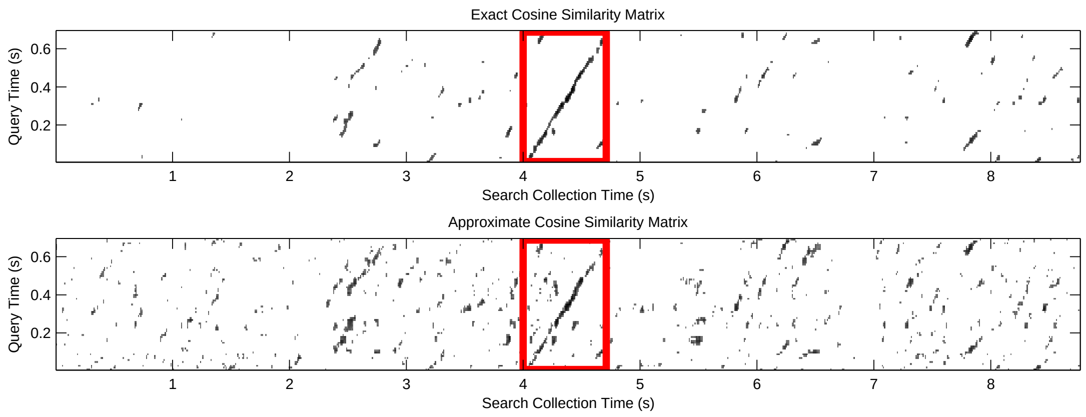

Figure 1: Exact (top) and approximate (bottom) similarity matrices, both thresholded at $\delta = 0.5$, for a single query example of the word "California" against 9 seconds of the search collection contain one instance at the 4 second mark (boxed).

Table 1: Algorithmic parameters and default values.

Parameter	Default Value
LSH signature size, b	64 bits
Cosine similarity threshold, δ	0.25
Nearest neighbor beam size, B	100000
Number of byte permutations, P	8
Diagonal probe length, D	20 frames (200 ms)
Percentile filter threshold, μ	50%
Percentile filter length, L	50 frames (0.5 s)
Gaussian filter width, σ	3 frames (30 ms)
Max DTW deviation, F	0.5

sional bit signatures, $h(x) \in \{0,1\}^b$, such that the Hamming distance between two such bit signatures can be used to approximate some distance metric in the original space. We consider the LSH variant that approximately preserves cosine similarity. This algorithm, defined by [13] as an extension of [12], proceeds by projecting each data point onto a collection of random vectors in $\mathbb{R}^d$ and uses each projection to determine a bit value by thresholding at zero. With each bit we encode the randomly oriented half-space that each point is contained in (assuming the boundary plane passes through the origin). This process can be implemented via matrix multiplication by generating a transformation matrix $\mathcal{T} \in \mathbb{R}^{d\times b}$, populated with random draws from $\mathcal{N}(0,1)$. The kth column, denoted $t_k \in \mathbb{R}^d$, defines the normal to the kth random hyperplane and the kth bit of the signature $h(x)$, denoted $h_k(x)$, takes value one if $x \cdot t_k \geq 0$ and zero otherwise. Each element of X and Q may by transformed into bit signatures via multiplication by $\mathcal{T}$ and thresholding at zero. Next, if we denote Hamming distance by $H(\cdot,\cdot)$, then the cosine similarity between acoustic feature vectors x_t and $x_{t'}$ can be approximated by (approaches equality as $b \to \infty$)

$$\cos(x_i, x_j) \approx \cos(\frac{H(h(x_i), h(x_j))}{b}\pi). \tag{1}$$

2.2. Binary Search for Nearest Neighbors

Given our efficient means to approximate cosine distances using LSH bit signatures, the next step is to use those signatures to generate sparse approximate similarity matrices M that characterize the pairwise cosine similarity between samples in Q and X. For the task of spoken term detection, our goal is to precompute offline an index that characterizes the acoustic content of the entire search collection X so that minimal effort is required for each query the system is presented with. Thus, we must modify the approach of [11], which was aimed at a one shot spoken term *discovery* task, though both methods are inspired by the Point Location in Equal Balls (PLEB) algorithm of [12].

The first step is to sort the collection of bit signatures derived from X lexicographically (alphabetical as bit strings). These sorted bit signatures form the backbone of the RAILS index (see Sec. 3 for a complete specification). Given this index, we can compute an approximate nearest neighbor set for query frame $q \in Q$ by performing a binary search (respecting lexicographic order) for the position $h(q)$ would take in the sorted list. Starting at this position, we test cosine similarity between B neighbors in the sorted list ($B/2$ in each direction), generating an approximate nearest neighbor set for q. Since we are only interested in a sparse similarity matrix with elements limited to good frame level matches, we require the approximate cosine similarity defined by Equation 1 to exceed a threshold δ for inclusion in the matrix (regardless of sorted list proximity).

Since the lexicographic sort places undue importance on the initial bits, this process is repeated for each $q \in Q$ under P bit order permutations to increase the likelihood that the true nearby neighbors of q will fall nearby in at least one of the sorted lists. The binary search completes in logarithmic time, but must be performed P times for each $q \in Q$, making the overall query time complexity $O(Pm \log n + PBm)$. On the surface the time complexity is logarithmic in the size of the search collection, but there is an important caveat. As n increases, the number of frames in the search collection similar to a given query frame will also increase. Thus, if B is held constant, the approximation fidelity will also decrease with increased collection size. In this way, the choice of B allows a very straightforward means to trade-off speed and accuracy.

Finally, in an effort to improve approximation accuracy with a limited increase of B, we augment the basic PLEB methodology with the diagonal probe of [11] that compares temporal neighbors of a match (probe length D). Figure 1 displays an exact and approximate (using both LSH and binary search method using default parameters of Table 1) cosine similarity matrix between a spoken query of the word "California" against 9 seconds of the Switchboard corpus. We find that a true occurrence of the query at the 4 second mark produces a clear diagonal line (boxed) in both images.

2.3. Two-Pass Repeated Trajectory Search

Given the sparse approximate similarity matrix M computed using the methods described above, we must now proceed to identify runs of frame level matches between regions of the search collection and the query. To accomplish this in time linear in the number of non-zero entries of M, we turn to a modified form of the two-pass search proposed in [14]. This method performs a coarse-to-fine search, first for syllable-size matches between the query and search collection with minimal relative time warping, followed by a DTW-based refinement step.

We borrow the image processing-based approach of [14] to recover syllable-sized diagonal lines present in the similarity matrix M as follows: (1) Transform M into binary matrix M' by thresholding similarities at δ. (2) Apply to M' a diagonal μ-percentile filter of length L with orientation *parallel* to the target line segments, allowing only diagonals of sufficient length and density to pass. (3) Apply a Gaussian filter of width σ with orientation *orthogonal* to the target line segments to allow variation in speaking rate. (4) Apply a classical one-dimensional Hough transform (i.e., r is varied, θ fixed at $-45\,\mathrm{deg}$) to the filtered image, amounting to a projection of the image onto the line $y = -x$. (5) Use the peaks of the Hough transform to define rays through which to search for line segments.

Our efficient implementation of the above steps for sparse matrices is of utmost importance. Since the sparse approximate similarity matrix will contain at most $mPBD$ non-zero elements (a constant in the face of increasing n), our implementation limits matrix processing to only those regions that contain this small proportion of non-zero elements. Without this consideration, the computational cost of this back-end processing would make the gains from LSH and PLEB irrelevant.

Once we have identified a collection of syllable-sized diagonal line matches, we can use them as starting points to search for an optimal time-warped alignment of the query to the local region of the search collection. For the spoken term *discovery* task considered in [11], our goal was to use segmental DTW (S-DTW) [15] to grow syllable-size matches out to arbitrarily long word or phrase level matches. However, in the present spoken term detection setting, we would like to match the entire query Q, no more no less, against however much of X that is necessary to produce a low DTW distance. This leads to a somewhat simpler refinement algorithm: given a detected line segment in the first pass, we apply S-DTW to find the curve that (i) passes through the midpoint of the line segment, and (ii) maximizes the similarity path integral that spans the entire query. To reasonably limit the search space, we introduce a single parameter F, which is the maximal allowable deviation of the optimal warp path from the diagonal relative to the query duration. In this way, with $F = 0.5$ (the value used in our experiments) the maximum allowable duration scaling is a factor of 2 (i.e., an additional $0.5m$ in each direction).

3. RAILS Index Structure

Given the typical structure of a speech corpus and the requirement of multiple bit signature permutations, the actual implementation of the RAILS index requires a fair amount of additional bookkeeping beyond the LSH signatures themselves. The contents of the RAILS index for a search collection of n frames (split into N files) include:

1. Number of bytes per signature, $b/8$ (1 `unsigned int`)
2. Number of speech files, N (1 `unsigned int`)
3. For each speech file: file name length (1 `unsigned int`), file name (array of `char`), frames in file (1 `unsigned int`)
4. n signature structures, each containing: speech file identifier (1 `unsigned int`), location (frame) in speech file (1 `unsigned int`), signature value (array of $b/8$ `unsigned char`)
5. Number of permutations, P (1 `unsigned int`)
6. For each permutation: byte-level permutation ($b/8$ `unsigned int`), sorted order (n `unsigned long`)

The application of LSH is linear in the search collection size, but is dwarfed by the front-end processing time for standard features like PLP and MFCC. The 433 hour evaluation set index using the default parameters in Table 1 completes in approximately 1 hour using a single processor (comparable to standard front end processing time). The total index size is 7GB and can be loaded from disk into memory in less than 1 minute. Note that the raw wave files alone (2-byte samples) would fill 25 GB. The index size is linear in both n and P, but independent of B.

4. Experiments

We evaluated the RAILS indexing and search method on a large-scale query-by-example spoken term detection task using the Switchboard conversational telephone speech corpus. We split the corpus into a 37 hour query set (from which we draw our query examples), a 48 hour development set used to understand the behavior of the algorithms, and a 433 hour evaluation set. A short-time frequency domain linear prediction acoustic front-end [16] (39 dimensional, including velocity and acceleration, sampled every 10 ms) was used for all experiments due to their demonstrated success in the zero resource regime [11]. Each feature dimension was normalized to zero mean and unit variance before application of LSH. We defined our keyword set as follows (each with median duration greater than 0.5 seconds and greater than 5 characters as text):

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 keyword type had from 20-162 query examples in the query set and from 40-1386 occurrences in the evaluation set. The median query example duration was less than 0.55 s for over half the keyword types (the maximum was 0.75 s).

We evaluated performance using three ubiquitous spoken term detection metrics, computed separately for each keyword type and reported as averages over the 43 keyword set (uniformly weighted): (1) figure-of-merit (FOM) is the average recall over the 10 operating points where the false alarm rate is $1, 2, \ldots, 10$ false alarms per hour; (2) oracular term weighted value (OTWV, defined in [2]) is a weighted difference between recall and false alarm rate–the oracular variant assumes we can choose an optimal threshold that maximizes this difference; (3) precision at 10 (P@10) is the fraction of the top ten ranked putative hits that are correct. For each keyword type, we considered both the median performance across the individual query examples and performance of the best single query example (before averaging over keyword type for both cases).

Table 2 lists the spoken term detection performance for various values of B. Performance strongly depends on B but nearly saturates at $B = 100\text{k}$ (see [11] for an examination of other parameters). Here, performance is comparable to previous

Table 2: Performance averaged over the 43 query types as a function of search beam width B (all scores in percent).

	Median Example			Best Example			Real Time Speedup	
B	FOM	OTWV	P@10	FOM	OTWV	P@10	Matrix	Overall
1,000	0.8	0.9	21.0	3.6	2.8	58.4	1,500,000	620,000
5,000	3.7	1.8	38.3	13.4	8.0	82.8	310,000	130,000
10,000	6.7	2.7	44.0	20.7	10.4	84.4	160,000	63,000
20,000	10.9	3.6	48.1	28.6	13.4	86.5	81,000	33,000
50,000	16.1	4.4	49.2	36.6	15.6	87.0	34,000	13,000
100,000	19.0	4.7	49.2	39.9	16.5	88.4	18,000	7,000
200,000	20.2	4.8	49.8	41.1	16.6	88.1	9,200	3,600

Table 3: Performance with example combination, both before and after match rescoring with exact DTW similarities.

Score Type	FOM	OTWV	P@10
Combined	56.3	22.4	77.9
Rescored, Median	21.0	6.4	58.5
Rescored, Best	42.5	20.1	90.7
Rescored, Combined	61.8	27.4	68.4

zero resource spoken term detection efforts [6, 9] on a conversational telephone speech task that is at least as difficult acoustically and of unprecedented scale. Even at $B = 100$k, our real time speedup is 25 times that of the high water mark for exact DTW-based search [10], which itself performs a lower-bound calculation for substantial speedup. If some degradation in accuracy is acceptable for a given use case, we can gain orders of magnitude additional speedup.

Performance also strongly depends on the length of the query. While many of our query types straddle the minimum 0.5 s duration, for longer keywords the performance values are substantially higher. For example, for one of the longer keywords, "California" (median query example duration of 0.66 s), the *median example* FOM, OTWV, and P@10 with $B = 200k$ were 49.5%, 19.9%, and 95%, respectively. As reported in previous studies [4, 5] and evidenced by the median-best gap, the performance strongly depends on the particular query example used. However, when presented with a representative query example, the system can produce near-perfect precision in the top 10 ranked hits and OTWVs comparable to out-of-vocabulary performance of early high-resource STD systems [2].

While the query set was extracted entirely from conversations not contained in the evaluation set, there was an overlap of speakers between them. This means that the evaluation is not strictly speaker independent and it has been clearly demonstrated in [11] that the within-speaker performance of our proposed method is substantially better than across-speaker performance. Several recent efforts [6, 7, 17, 8] that address speaker independence in the zero resource regime would all be compatible with the proposed indexing method.

Finally, as considered in [5], significant improvements can be had by combining the results from several query examples of the same keyword type. We implement a simple sum combination of the DTW similarities (defined as 1-DTW distance) clamped to a minimum value of 0.82 (determined empirically on a heldout set). We also considered rescoring all matches using the DTW distance scores computed from the original features, i.e. before LSH. Table 3 lists performance values for these cases (defaults of Table 1 used). We see that query combination outpaces the best query example alone on both FOM and OTWV, demonstrating the usefulness of multiple query examples when they are available. Exact DTW rescoring also makes a significant improvement in all cases, indicating that increased signature length (b) would improve performance (at the expense of larger index size, but no significant increase in run time).

5. Conclusions

We have presented a scalable method for query-by-example spoken term detection that operates with no language-specific assumptions or resources whatsoever (beyond the front end selection). Compared with previously proposed approaches to zero resource query-by-example spoken term detection, RAILS enables orders of magnitude speedup, capable of matching query response times of high resource counterparts that employ efficient finite state machine-based indexing. Various configurations provide search accuracies comparable to both high resource and zero resource predecessors.

6. References

[1] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proc. SIGIR*. ACM, 2007.

[2] 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.

[3] W. Shen, C. White, and T. Hazen, "A comparison of query-by-example methods for spoken term detection," in *Proc. Interspeech*, 2009.

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

[5] J. Tejedor, I. Szoke, , and M. Fapso, "Novel methods for query selection and query combination in query-by-example spoken term detection," in *Proc. SSCS*, 2010.

[6] Y. Zhang and J. Glass, "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams," in *ASRU*, 2009.

[7] A. Muscariello, G. Gravier, and F. Bimbot, "Zero-resource audio-only spoken term detection based on a combination of template matching techniques," in *Proc. Interspeech*, 2011.

[8] X. Anguera, "Speaker independent discriminant feature extraction for acoustic pattern matching," in *Proc. ICASSP*, 2012.

[9] F. Metze et al., "The spoken web search task at MediaEval 2011," in *Proc. ICASSP*, 2012.

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

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

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

[13] M. Charikar, "Similarity estimation techniques from rounding algorithms," in *STOC*, 2002.

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

[15] A. Park and J. R. Glass, "Unsupervised pattern discovery in speech," *IEEE T-ASLP*, vol. 16, no. 1, pp. 186–197, 2008.

[16] S. Thomas, S. Ganapathy, and H. Hermansky, "Recognition of reverberant speech using frequency domain linear prediction," *IEEE Signal Processing Letters*, pp. 681–684, 2008.

[17] A. Jansen and K. Church, "Towards unsupervised training of speaker independent acoustic models," in *Interspeech*, 2011.