# Efficient Spoken Term Discovery Using Randomized Algorithms

Aren Jansen[1], Benjamin Van Durme[2]

*Human Language Technology Center of Excellence & The Center for Language and Speech Processing*
*Johns Hopkins University, 3400 N. Charles St., Baltimore, MD 21218 USA*
[1]aren@jhu.edu
[2]vandurme@cs.jhu.edu

***Abstract*—Spoken term discovery is the task of automatically identifying words and phrases in speech data by searching for long repeated acoustic patterns. Initial solutions relied on exhaustive dynamic time warping-based searches across the entire similarity matrix, a method whose scalability is ultimately limited by the $O(n^2)$ nature of the search space. Recent strategies have attempted to improve search efficiency by using either unsupervised or mismatched-language acoustic models to reduce the complexity of the feature representation. Taking a completely different approach, this paper investigates the use of randomized algorithms that operate directly on the raw acoustic features to produce sparse approximate similarity matrices in $O(n)$ space and $O(n \log n)$ time. We demonstrate these techniques facilitate spoken term discovery performance capable of outperforming a model-based strategy in the zero resource setting.**

## I. Introduction

Nearly all modern speech technologies rely on supervision at some stage of the pipeline, most commonly in the form of orthographic word or phoneme transcriptions. However, for the vast majority of the world's languages, little or no training resources are readily available. In the extreme *zero resource* setting, where we have no transcribed training data and no *a priori* linguistic knowledge for the language of application, the only recourse is to fall back on language independent technologies or fully unsupervised training methods.

Given a collection of speech in an unknown language, one point of entry is the automatic discovery of the linguistic structure present in the data. In the absence of supervision, intuition informs us that longer units (e.g. syllables, words, and phrases) will be easier to identify and catalog across speakers than short segmental units such phonemes. The spoken term discovery task attempts to identify lexical or phrasal units present in the audio by searching the audio for repeated trajectories in the acoustic feature space [1][2][3][4][5][6][7]. Even though they are unlabeled, the resulting automatically discovered units have clear practical utility, already having been demonstrated useful for summarization [6][8] and speech retrieval applications [9][10]. Moreover, it has been demonstrated that a sufficiently efficient term discovery algorithm can provide top-down constraints that enable fully unsupervised training of a *speaker independent* subword unit acoustic model [11].

To date the vast majority of spoken term discovery approaches have been built around dynamic time warping (DTW) based search of similarity matrices spanning the audio. Both the similarity matrix computation and the dynamic programming search are inherently $O(n^2)$ in space and time, limiting the size of the speech collection to which these methods can be applied. Recent efforts attempted to reduce the constants by mapping the original acoustic feature vectors to sparse representations that limit the necessary computation to pairs of frames with at least moderate degrees of similarity. In particular, both unsupervised [10][12] and mismatched-language [6] acoustic models have been used to produce both posteriorgrams or tokenizations to seed the search. In a subsequent study [13], it was demonstrated that raw acoustic features could provide more discriminative *cross-speaker* word DTW similarities than these zero resource model-based approaches. However, the denser raw acoustic features did not permit the same algorithmic speedups.

With the explicit goal of improving tractability of raw acoustic feature-based term discovery, this paper investigates the use of existing randomized approximation algorithms to circumvent the need for acoustic models, unsupervised or otherwise. In particular, we exploit recent algorithmic advances in using random projections to efficiently compute approximate nearest neighbors in the acoustic feature space, permitting sparse similarity matrix computation in $O(n)$ space and $O(n \log n)$ time. As with most randomized algorithms, these approaches involve a trade-off between speed and accuracy for which we provide a detailed experimental study. Having efficiently computed sparse similarity matrices, we proceed to apply our efficient two pass repeated interval search algorithm originally presented in [6], which was designed to operate on sparse similarity matrices of precisely the form produced by the randomized methods. We find this combination of approaches surpasses the mismatched acoustic model approach in both execution time and discovery performance. We begin with a complete description of the proposed solution.

## II. The Algorithms

Our starting point is a raw acoustic feature vector time series of the form $X = x_1 x_2 \dots x_n$, where each $x_i \in \mathbb{R}^d$. Our proposed approach consists of three stages of approximation, each complete with parameters that allow for a controlled trade-off between speed and fidelity. The first stage maps samples in the time series $X$ into series of fixed-length bit signatures using a randomized hashing technique, permitting efficient estimation of cosine distances between points in the original space. The second stage uses the bit signatures to determine approximate nearest neighbors sets within $X$ in the service of computing sparse approximate similarity matrices for the form $M \in \mathbb{R}^{n \times n}$, where

$$M_{ij} = \frac{x_i \cdot x_j}{\|x_i\| \|x_j\|} \tag{1}$$

is the cosine similarity between $x_i$ and $x_j$. Finally, we use these similarity matrices to search for repeated acoustic patterns using a two-pass, coarse-to-fine strategy using efficient image processing techniques. In this section, we provide details for each of the three stages. For easy reference, Table I lists all parameters involved along with the default values used in the experiments described in Section III.

## A. Locality Sensitive Hashing

Locality sensitive hashing (LSH) refers to a family of algorithms for hashing high dimensional feature vectors, $x \in \mathbb{R}^d$, into low dimensional *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. The distance metric approximated is dependent on the hash function used. LSH has been used in a variety of image and language applications, including near-duplicate detection in web [14] and image search [15], building large-scale thesauri [16], and topic detection and tracking [17]. While LSH has not found its way into speech applications ([18] is one exception), it has been recently connected to compressive sensing, which has been a popular topic of study for speech coding and recognition applications with a different goal of constructing efficient and robust signal decompositions [19]. Here we employ the LSH routine of [20], an extension of [21], which approximately preserves cosine similarity.

The basic idea is to project each test sample onto a collection of random vectors in $\mathbb{R}^d$ and use each projection to determine a bit value by thresholding at zero. Alternatively, we may view each vector as defining a random hyperplane containing the origin, such that the corresponding bit vector indicates which half-space the test point lies. Formally, we first generate a transformation matrix $\mathcal{T} \in \mathbb{R}^{d \times b}$, populated with random draws from $\mathcal{N}(0, 1)$. If we assign the $k$th column of this matrix to the random vector $r_k \in \mathbb{R}^d$, then the $k$th bit of the signature $h(x)$, denoted $h_k(x)$, is given by:

$$h_k(x) = \begin{cases} 1 & \text{if } x \cdot r_k \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Note that this whole operation may be performed in one matrix multiplication of $X$ and $\mathcal{T}$, followed by 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

$$\cos(x_t, x_{t'}) \quad \approx \quad \cos\left(\frac{H(h(x_t), h(x_{t'}))}{b}\pi\right), \tag{3}$$

TABLE I
List of parameters used in the various algorithmic components, along with default experiment values.

| Parameter | Default Value |
|---|---|
| LSH signature size, $b$ | 32 bits |
| Cosine similarity threshold, $\delta$ | 0.5 |
| Nearest neighbor beam size, $B$ | 100 |
| Number of byte permutations, $P$ | 4 |
| Diagonal probe length, $D$ | 20 frames (200 ms) |
| Posterior threshold, $f$ | 0.1 |
| Percentile filter threshold, $\mu$ | 45% |
| Percentile filter length, $L$ | 50 frames (0.5 s) |
| Gaussian filter width, $\sigma$ | 3 frames (30 ms) |
| Max S-DTW deviation, $F$ | 10 |
| Dissimilarity budget, $C$ | 7 |
| Path trimming threshold, $T$ | 0.5 |

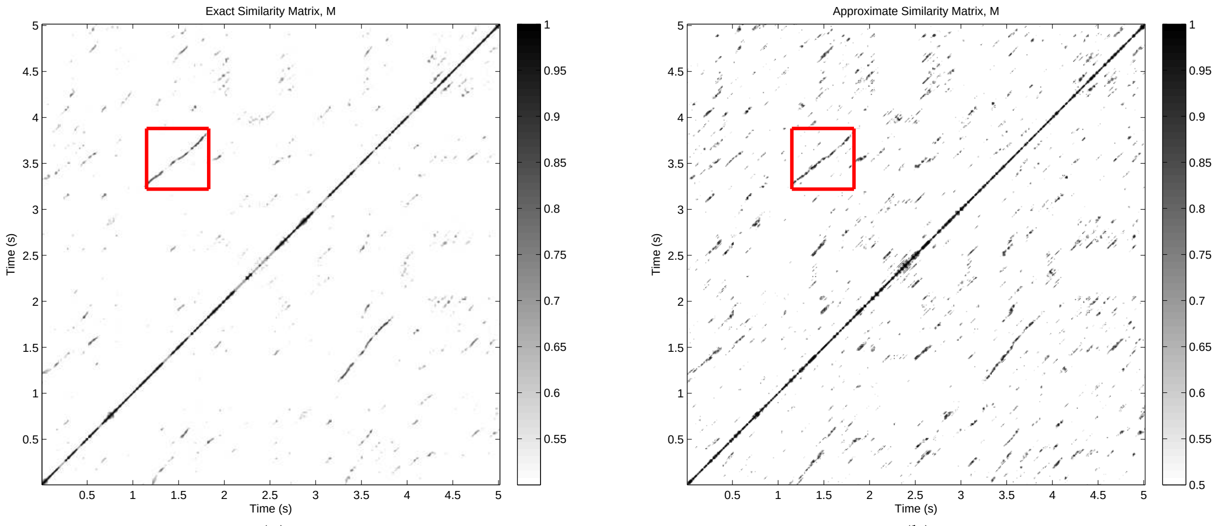
a relation that approaches equality as $b$ goes to infinity. Here, the intuition is that increased numbers of random projections specify in finer detail the ray each test point is contained in, providing an increasingly accurate approximation of the angle between pairs of points.

## B. Point Location in Equal Balls

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 points in $X$. We apply the Point Location in Equal Balls (PLEB) algorithm of [21], a descendant of [22], to perform approximate nearest neighbor search over speech frames. By limiting cosine similarity computation to the nearest neighbor set of a given test point, we can avoid spending time in regions of $M$ with negligible values. The savings can be dramatic as PLEB operates over a collection of LSH bit signatures, running in time $O(n \log n)$, rather than the exhaustive $O(n^2)$.

PLEB consists of two steps, which may be performed one or more times: (i) sort the collection of bit signatures *lexicographically*, which amounts to an alphabetical sort of the bit strings; and then (ii) linearly sweep through the sorted list, comparing each element to a fixed width *beam* of signatures falling nearby under the sort. When performing multiple iterations, the sort considers the bit signatures via a *permuted* order: as there is nothing distinguished by the position of a bit in an LSH signature ("all bits are created equal"), then by permuting the order of the bit signatures and resorting, one is increasing the chance of signatures with low Hamming distance to fall nearby each other. Performing $P$ iterations of this algorithm requires $O(Pn \log n)$ sorting operations and $O(PBn)$ neighbor comparisons, where $B$ is the beam width of the search. In the analysis one might treat $P$ and $B$ as constants to be disregarded in the face of $n \log n$. In practice we find that $P$ directly leads to a linear multiplier on run-time, while $B$ is often set to be larger than $\log n$, and thus the linear scan dominates the search.

In our application, we are only interested in producing sparse similarity matrices $M$, as low similarity regions are unlikely to contribute to the subsequent search for repeated trajectories. Thus, as we compute approximate cosine similarities, we need only include them in $M$ if they exceed some



(a) (b)

Fig. 1. Exact (a) and approximate (b) similarity matrices, both thresholded at 0.5, computed on 5 seconds of Switchboard speech. The boxed diagonal line segment corresponds to two occurrences of the word "recyclable" spoken by the same speaker. For the approximate case, both LSH and PLEB are used with the default values of Table I.

threshold $\delta$. Now, $O(PBn)$ neighbor comparisons can produce at most $O(n)$ entries of $M$, so as $\delta$ is lowered $B$ must be increased to prevent a substantial loss in the approximation. Moreover, as the problem size grows, our linear space approximation will incur an increasingly substantial loss in $M$ if $B$ is not increased accordingly. In Section III, we consider approximation accuracy as a function of the PLEB parameters on actual speech data.

Finally, in an effort to improve approximation accuracy with a limited increase of $B$, we augment the basic PLEB routine with an additional processing step tailored to the spoken term discovery task at hand. Since our acoustic feature vectors have a temporal ordering and we are searching for runs of similar, near contiguous frames, finding an isolated pair of highly similar frames will provide insufficient evidence for further action downstream. Thus, given such a pair, we can perform a constant number of comparisons in the temporal vicinity to probe whether a run of similar frames are also present (as compared to the sortal vicinity under stage (i) of basic PLEB). We limit this search to diagonal probes in the similarity matrix that assume perfect linear alignment of any repeated utterance, if present. We introduce an additional parameter, $D$, that refers to the *total diagonal length* in frames that we search upon finding a pair of points whose similarity exceeds our similarity threshold, $\delta$ (i.e., we test $D/2$ both forward and backward in time). In the best case, we would need our approximate search to find only a single pair of frames across a repeated utterance to provide an anchor around which we could search for additional matches nearby in the original temporal order.

In practice, it is often desirable to compute self-similarity matrices as well as similarity matrices between distinct vector times series (i.e. comparing two separate audio files). In this case, the PLEB procedure needs to be outfitted for parallel lexicographic sorts of LSH signatures of the two audio files separately. The beam search and subsequent similarity comparisons are conducted across the two sorted lists, with extra bookkeeping required to maintain two distinct scan positions.

Figure 1 displays an exact and approximate (using both LSH and PLEB) cosine similarity matrix of 5 seconds of the Switchboard corpus. In both cases, all cosine similarity values below 0.5 are discarded. We find that the repeated utterance of the word "recyclable" produces a clear diagonal line (boxed in red). We find that the randomized algorithms introduce a noticeable level of background noise, though the diagonal correlation of that noise tends to be sufficiently limited. Thus, sufficiently long diagonal lines arising from repeated words and phrases may still be easily identified.

### C. *Two-Pass Repeated Trajectory Search*

Given our sparse approximate similarity matrix $M$ computed with LSH and PLEB, we must now search for repeated acoustic trajectories that manifest themselves as closely matching DTW paths. Performing exhaustive DTW search on a large similarity matrix can become intractable quickly given the $O(n^2)$ nature of the algorithm. Moreover, the sparse matrix format does not lend itself to this traditional dynamic programming solution. As such, we turn to the two-pass search proposed in [6], which (i) performs a fast diagonal line segment search for syllable-sized units in the sparse matrix and (ii) performs a segmental DTW [1] search restricted to those regions that had a sufficiently confident diagonal match.

*1) Fast Diagonal Line Search:* Treating our similarity matrix as a real valued image, we can use common image processing techniques to recover any present diagonal lines. We borrow the method of [6], reproduced below for completeness:

1. Transform the matrix $M$ into binary matrix $M'$ by applying a similarity threshold $\delta$ such that $M'_{ij} = 1$ if $M_{ij} > \delta$ and 0 otherwise.
2. Apply to $M'$ a diagonal $\mu$-percentile filter of length $L$ with orientation *parallel* to the target line segments ($45^\circ$

relative to the $x$-axis of the image), allowing only diagonal line structures of sufficient length (and thus sufficient repeated trajectory duration) and density to pass.

3. Apply a Gaussian filter of width $\sigma$ with orientation *orthogonal* to the target line segments ($-45°$ relative to the $x$-axis of the image) to the thresholded image, transversely smearing the remaining pixels to allow some variation in speaking rate.
4. Apply a classical one-dimensional Hough transform (i.e., $r$ is varied, $\theta$ fixed at $-45°$) 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.

One complication to this simple approach is we must adapt these techniques to the sparse matrix setting, which introduces some extra bookkeeping. The interested reader may find more details on this procedure in [6].

*2) Match Refinement with Segmental DTW:* While diagonal line search can be efficiently performed using the Hough transform, natural prosodic variation across acoustic realizations of a given word type (both within and across speakers) can easily break the warp-free assumption. The original spoken term discovery approach addresses this reality by performing an exhaustive segmental DTW (S-DTW) search for all possible warped paths in the image [1]. However, the computational cost of applying this globally was high, requiring 33 machine-hours to process just one hour of speech.

We base our two-pass approach on the assumption that if a strong nonlinearly warped match is present, it might still contain a shorter, warp-free, syllable-sized line segment that the first pass can discover. Thus, if we operate the first pass in high recall, low precision mode, we can limit our application of S-DTW to local regions around each diagonal line segment match. In particular, 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.

Our implementation of the S-DTW algorithm involves three additional parameters. The first, meant to reasonably limit the search space, is the maximal allowable deviation $F$ (in frames) of the optimal warp path from the diagonal. Next, to impose the constraint that our optimal warp path passes through the first-pass line segment, we start from the segment midpoint and apply S-DTW both forward and backward in time with a dissimilarity budget of $C$ in each direction (i.e. we stop growing the match when the path integral of $(1 - M_{ij})/2$ exceeds $C$). Finally, we trim the ends of the optimal path by removing all initial or final points in the path that have a similarity less than $T$.

## III. Experiments

Given we are proposing multiple stages of approximation, we evaluated the performance of our approach at each point in the pipeline from acoustic features to discovered terms. We begin with a specification of the evaluation data and acoustic front ends considered.

### A. Evaluation Data and Representations

Our evaluation was limited to subsets of the Switchboard conversational telephone speech corpus that, while consisting of English speech, can be treated as a zero resource language by abstaining from applying supervised techniques. We borrowed the evaluation paradigm of [11] and [13], which characterizes a representation's ability to associate examples of the same word type while simultaneously preventing incorrect matching across word types. Using time aligned word transcripts for the Switchboard corpus, we extracted a set of spoken word examples with a minimum duration of 0.5 seconds and minimum character length of 5 as text. Given each pair of word instances, which can be drawn from the same or different speakers, we computed their DTW distance using cosine distance (one minus cosine similarity) as the frame-level distance metric, using both the exact and LSH/PLEB approximation. Each example pair is either of the same or different word type, setting up a natural information retrieval task of isolating correct from incorrect word matches, for which we can sample a precision-recall curve across the full range of DTW distance thresholds. We characterized the quality of each representation and similarity matrix computation strategy by the resulting average precision (AP). For each experiment, we report both overall AP and the AP restricted to word example pairs from the same speaker.

All of our raw acoustic feature-based experiments (both exact and approximate) were performed on top of the short-time frequency domain linear prediction (FDLP-S) acoustic front-end [23]. FDLP-S (39-dimensions, including velocity and acceleration, sampled every 10 ms) was demonstrated in [13] to outperform standard PLP and MFCC speech features for this task, making them a natural starting point for our investigation into approximation methods. Note that the features were globally normalized to zero mean and unit variance in each dimension. In Section III-C, we compare the end-to-end term discovery performance of our proposed approximation methods to both matched (English) and mismatched-language (German, Spanish) multilayer perceptron (MLP) acoustic models using the architecture of [24].

### B. Similarity Matrix Approximation

Our first set of experiments examined the approximation fidelity of the LSH and PLEB algorithms as a function of the various algorithmic parameters. For each of these experiments, we limited evaluation to a set of 10 Switchboard conversation sides (approximately one hour of speech), containing 79,800 word example pairs, 646 of which are the same word type (the target pairs), and 118 of those are also uttered by the same speaker. Table II lists the average precision performance (both within speaker and overall) for the exact similarity computation as a function of cosine similarity threshold $\delta$. These performance values serve as the reference for the approximation experiments that follow. The listed speedups were all benchmarked comparing a single pair of 10 minute conversation sides ($n = 60,000$); thus, the exact similarity

TABLE II
AVERAGE PRECISION AS A FUNCTION OF COSINE DISTANCE THRESHOLD $\delta$ USING EXACT DISTANCES BETWEEN THE ORIGINAL FDLP-S FEATURES.

|  | **Average Precision (%)** |  |
|---|---|---|
| $\delta$ | **Same spkr** | **Overall** |
| 0.75 | 46.9 | 13.2 |
| 0.50 | 71.4 | 38.3 |
| 0.25 | 73.8 | 45.2 |
| 0.00 | 73.5 | 45.3 |
| -0.50 | 74.0 | 45.8 |
| -1.00 | 74.3 | 45.8 |

TABLE III
AVERAGE PRECISION AS A FUNCTION OF COSINE DISTANCE THRESHOLD $\delta$ USING LSH/PLEB WITH $P = 4$, $B = 100$, $D = 20$, $b = 32$.

|  | **PLEB** |  | **Average Precision (%)** |  |
|---|---|---|---|---|
| $\delta$ | **Speedup** | $N_{comp}$ | **Same spkr** | **Overall** |
| 0.75 | 14.9 | 265 M | 55.8 | 25.4 |
| 0.50 | 10.5 | 353 M | 60.9 | 33.9 |
| 0.25 | 8.4 | 423 M | 62.3 | 36.3 |
| 0.00 | 7.6 | 448 M | 62.2 | 36.1 |
| -0.50 | 7.4 | 455 M | 62.1 | 36.6 |
| -1.00 | 7.4 | 456 M | 62.2 | 36.6 |

matrix computation required $n^2 = 3.6$ billion frame-level comparisons. For each approximate similarity matrix computation, we also list the number of frame-level comparisons performed, denoted $N_{comp}$, to be compared with the 3.6 billion of the exact method.

Table III lists the LSH/PLEB approximation as a function of the cosine similarity threshold, and is directly comparable to Table II. Using the default LSH/PLEB parameter settings, we achieved 88% of the overall AP of the exact computation in less than 1/10 the runtime. Interestingly, for high similarity thresholds (e.g. 0.75), the approximation actually outperformed the exact computation, as the noise induced by LSH allowed more comparisons to exceed this overly strict threshold, favoring same-type matching in subsequent DTW computations. When considering absolute speedup values, it is important to note that, in theory, the PLEB speedups listed would grow linearly with $n$, making it beneficial to concatenate conversations (at the expense of extra bookkeeping downstream). However, it is important to also realize that the number of comparisons performed by the PLEB approximation is linear in $n$. Thus, if the similarity matrix is not sufficiently sparse, $B$ will have to grow with $n$ to maintain the same fidelity of approximation.

Table IV lists the performance for two LSH signature sizes, 32 and 64 bits. Here, LSH speedup is the runtime savings resulting from using the approximate cosine distance calculation to perform a full $n^2$ similarity matrix computation, while PLEB speedup is the combined savings of the sparse approximation. We find that the extra bits led to only a minor LSH slowdown, but actually sped the PLEB portion up overall by wasting less time on frame-level comparisons resulting from LSH approximation error. Still, the extra bits had little impact on the overall AP performance. Interestingly, we observed a larger relative gain in the same speaker case, as acoustic variation across speakers prevented cross speaker matching from benefiting from the extra resolution.

TABLE IV
SPEEDUP AND AVERAGE PRECISION AS A FUNCTION OF SIGNATURE SIZE $S$ USING LSH/PLEB WITH $\delta = 0.5$, $P = 4$, $B = 100$, $D = 20$.

|  | **Speedup** |  |  | **Average Precision (%)** |  |
|---|---|---|---|---|---|
| $S$ | **LSH** | **PLEB** | $N_{comp}$ | **Same spkr** | **Overall** |
| 32 | 1.4 | 10.5 | 353 M | 60.9 | 33.9 |
| 64 | 1.3 | 11.2 | 299 M | 63.6 | 34.6 |

Next, we consider the competing role of the beam width $B$ and permutation count settings $P$ of the PLEB algorithm, with results in Table V. First, we observed the clear linear dependence on $B$ and $P$ both in runtime and number of frame-level comparisons made. We also observed a strong dependence of AP performance on these settings, though saturation is evident at $B = 100$ and $P = 4$. Indeed, when we increased the beam width to 1000, our performance can actually drop, as the noise from LSH approximation error can artificially improve DTW scores between word examples of different types (all at a significant increase in runtime). Finally, in Table VI we examine the interdependence of beam width $B$ and diagonal probe length $D$. Since the diagonal probe was only performed for beam search points exceeding threshold $\delta$, runtime grows more slowly in $D$ than it does $B$. However, increasing $D$ still provided significant AP improvements, though they largely saturate at $B = 100$ and $D = 20$. Note that diagonal probing is the simplest means of secondary local search; indeed, a greedy DTW search in the vicinity may prove more effective and will be implemented in future efforts.

## C. *Spoken Term Discovery*

Since the spoken term discovery algorithm produces a limited set of predicted matches, the above-defined evaluation strategy must be modified slightly. First, each discovered repetition produced by our term discovery algorithm was assigned a score equal to the DTW score between the two intervals of speech. For each pair of word examples in our evaluation set, we assigned a discovered repetition to it if it overlapped at least 50% of the duration of both examples in the evaluation pair. The score for the evaluation example pair was subsequently defined as the DTW distance for the assigned discovered repetition; if multiple discovered repetitions overlap the evaluation pair, we assigned the lowest possible DTW distance, and if no overlapping discovered repetitions were produced, we assigned a distance of infinity. Having assigned each evaluation word example pair a distance score in this fashion, we can sample a precision-recall curve as before. However, given the inherent match duration thresholds built into the term discovery procedure of Section II-C, average precision ceases to be a natural metric. Thus, we resort to $R$-precision (a.k.a. precision-recall breakeven) to characterize end-to-end discovery performance.

For our spoken term discovery evaluation, we expanded the Switchboard evaluation set to approximately 4 hours of speech (40 conversation sides, distinct from the 10 used earlier), containing a total of 944,625 word example pairs (2021 same word type and 396 of these also uttered by the same speaker) meeting the requirements for inclusion in the

TABLE V
SPEEDUP AND AVERAGE PRECISION FOR VARIOUS BEAM WIDTHS $B$ AND PERMUTATION COUNTS $P$ USING LSH/PLEB ($\delta = 0.5$, $D = 20$, $b = 32$).

| | | **PLEB** | | **Average Precision (%)** | |
|---|---|---|---|---|---|
| $B$ | $P$ | **Speedup** | $N_{comp}$ | **Same spkr** | **Overall** |
| 10 | 1 | 361.5 | 10.3 M | 39.3 | 14.5 |
| 10 | 2 | 184.5 | 20.6 M | 47.7 | 19.1 |
| 10 | 4 | 91.8 | 40.5 M | 52.0 | 23.8 |
| 100 | 1 | 40.6 | 92.0 M | 54.1 | 24.7 |
| 100 | 2 | 20.6 | 181 M | 58.6 | 29.9 |
| 100 | 4 | 10.5 | 353 M | 60.9 | 33.9 |
| 1000 | 1 | 4.8 | 774 M | 50.6 | 26.8 |
| 1000 | 2 | 2.4 | 1.53 B | 60.9 | 33.5 |
| 1000 | 4 | 1.3 | 2.93 B | 60.2 | 32.9 |

evaluation. Table VII lists the total term discovery runtime across the 4 hours of speech (in machine-hours; for wall clock time, divide by your compute cluster size) and $R$-precision for three model-based approaches (all using the posteriorgram-based speedups of [6]) and the proposed randomized algorithm methods. The English MLP entry is the matched acoustic model case and sets a supervised performance ceiling for the task. The Spanish and German MLPs are two mismatched acoustic models, and provide two zero resource baselines. We find that the proposed method, operating on raw features, outperforms both mismatched acoustic model systems in $R$-precision and represents the best published runtime for spoken term discovery task in the zero resource setting. As an external point of comparison, note that the exhaustive S-DTW approach of [1] would require approximately 528 machine-hours to process these four hours of speech.

## IV. CONCLUSIONS

We have presented a solution to the spoken term discovery task capable of operating on arbitrary raw acoustic features and posts the best reported execution time to date in the zero resource setting. We found that the randomized LSH and PLEB algorithms provide a suitable strategy for efficient computation of sparse approximate similarity matrices over large collections of speech data, a fact that can potentially have ramifications more broadly for speech applications involving large similarity matrices. Finally, we have demonstrated that our proposed method outperforms a mismatched acoustic model-based strategy for efficient term discovery in both runtime and accuracy.

## ACKNOWLEDGMENTS

The authors would like to thank Samuel Thomas of the Center for Language and Speech Processing at Johns Hopkins University for assistance in preparing the FDLP-S acoustic features and MLP acoustic models used in our experiments.

TABLE VI
SPEEDUP AND AVERAGE PRECISION FOR VARIOUS BEAM WIDTHS $B$ AND DIAG. PROBE LENGTHS $D$ USING LSH/PLEB ($\delta = 0.5$, $P = 4$, $b = 32$).

| | | **PLEB** | | **Average Precision (%)** | |
|---|---|---|---|---|---|
| $B$ | $D$ | **Speedup** | $N_{comp}$ | **Same spkr** | **Overall** |
| 10 | 0 | 966.5 | 2.40 M | 29.1 | 8.4 |
| 10 | 10 | 187.6 | 19.5 M | 49.3 | 21.3 |
| 10 | 20 | 91.9 | 40.5 M | 52.0 | 23.8 |
| 100 | 0 | 195.0 | 24.0 M | 52.4 | 23.3 |
| 100 | 10 | 23.1 | 171 M | 61.3 | 33.3 |
| 100 | 20 | 10.5 | 353 M | 60.9 | 33.9 |

TABLE VII
RUNTIME AND DISCOVERY $R$-PRECISION (RANKED BY MATCH DTW DISTANCE) FOR ENGLISH PHONE POSTERIORGRAMS (MATCHED), GERMAN/SPANISH PHONE POSTERIORGRAMS (MISMATCHED), AND THE PROPOSED SYSTEM USING THE DEFAULT PARAMETERS OF TABLE I.

| | | **$R$-Precision (%)** | |
|---|---|---|---|
| **Features** | **Runtime** | **Same spkr** | **Overall** |
| English MLP | 2.51 m-hr | 59.8 | 40.2 |
| Spanish MLP | 5.43 m-hr | 29.4 | 7.3 |
| German MLP | 3.26 m-hr | 48.4 | 18.9 |
| FDLP-S LSH | 2.71 m-hr | 52.3 | 19.0 |

## REFERENCES

[1] A. Park and J. R. Glass, "Unsupervised pattern discovery in speech," *IEEE T-ASLP*, vol. 16, no. 1, pp. 186–197, 2008.
[2] L. ten Bosch, H. V. hamme, and L. Boves, "A computational model of language acquisition: Focus on word discovery," in *Interspeech*, 2008.
[3] L. ten Bosch and B. Cranen, "A computational model for unsupervised word discovery," in *Interspeech*, 2007.
[4] A. Muscariello, G. Gravier, and F. Bimbot, "Audio keyword extraction by unsupervised word discovery," in *Interspeech*, 2009.
[5] G. Zweig, "New methods for the analysis of repeated utterances," in *Interspeech*, 2009.
[6] A. Jansen, K. Church, and H. Hermansky, "Towards spoken term discovery at scale with zero resources," in *Interspeech*, 2010.
[7] X. Anguera, R. Macrae, and N. Oliver, "Partial sequence matching using an unbounded dynamic time warping algorithm," in *Proc. of ICASSP*, 2010.
[8] R. Flamary, X. Anguera, and N. Oliver, "Spoken wordcloud: Clustering recurrent patterns in speech," in *Proc. of CBMI*, 2011.
[9] M. Dredze, A. Jansen, G. Coppersmith, and K. Church, "NLP on spoken documents without ASR," in *Proc. of EMNLP*, 2010.
[10] M.-H. Siu, H. Gish, A. Chan, and W. Belfield, "Improved topic classification and keyword discovery using an HMM-based speech recognizer trained without supervision," in *Proc. of Interspeech*, 2010.
[11] A. Jansen and K. Church, "Towards unsupervised training of speaker independent acoustic models," in *Interspeech*, 2011.
[12] Y. Zhang and J. R. Glass, "Towards multi-speaker unsupervised speech pattern discovery," in *Proc. of ICASSP*, 2010.
[13] M. Carlin, S. Thomas, A. Jansen, and H. Hermansky, "Rapid evaluation of speech representations for spoken term discovery," in *Proc. of ICASSP*, 2011.
[14] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," *Computer Networks and ISDN Systems*, vol. 29, pp. 1157–1166, 1997.
[15] Y. Ke, R. Sukthankar, L. Huston, Y. Ke, and R. Sukthankar, "Efficient near-duplicate detection and sub-image retrieval," in *In ACM Multimedia*, 2004, pp. 869–876.
[16] D. Ravichandran, P. Pantel, and E. Hovy, "Randomized algorithms and NLP: Using locality sensitive hash functions for high speed noun clustering," in *ACL*, 2005.
[17] S. Petrovic, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to Twitter," in *NAACL*, 2010.
[18] J. Labiak and K. Livescu, "Nearest neighbors with learned distances for phonetic frame classification," in *Interspeech*, 2011.
[19] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk, "Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors," *Arxiv preprint arXiv:1104.3160*, 2011.
[20] M. Charikar, "Similarity estimation techniques from rounding algorithms," in *STOC*, 2002.
[21] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality." in *STOC*, 1998.
[22] R. M. Karp, O. Waarts, and G. Zweig, "The bit vector intersection problem," in *Proc. of FOCS*, 1995.
[23] S. Thomas, S. Ganapathy, and H. Hermansky, "Recognition of reverberant speech using frequency domain linear prediction," *IEEE Signal Processing Letters*, pp. 681–684, 2008.
[24] ——, "Phoneme recognition using spectral envelope and modulation frequency features," in *Proc. of ICASSP*, 2009.