GRAPH INDEX BASED QUERY-BY-EXAMPLE SEARCH ON A LARGE SPEECH DATA SET

Kazuo Aoyama, Atsunori Ogawa, Takashi Hattori, Takaaki Hori, and Atsushi Nakamura

NTT Communication Science Laboratories, NTT Corporation
2-4, Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237, Japan

ABSTRACT

This paper presents a neighborhood graph index approach for query-by-example search using dynamic time warping (DTW) on Gaussian mixture model (GMM) posteriorgram sequences. The approach is intended to achieve a significant speed-up of a spoken term detection (STD) task for resource-limited situations. The proposed method employs a degree-reduced k-nearest neighbor (k-*DR*) graph as an index. A set of k-*DR* graphs is pre-constructed *off-line* from a large number of GMM posteriorgram sequences. Given a query posteriorgram sequence, one k-*DR* graph is selected from the set as the index. By applying a newly introduced combination of greedy-search (*GS*) and breadth-first search (*BFS*) algorithms to the selected k-*DR* graph index, the proposed method efficiently achieves query-by-example STD. Experimental results on the MIT lecture corpus demonstrate that the proposed method works much faster than a state-of-art method by more than one order magnitude, keeping almost the same precision.

Index Terms— Neighborhood graph index, Spoken term detection, Query-by-example search, Dynamic time warping, Gaussian mixture model posteriorgram

1. INTRODUCTION

The volume of speech and audio data stored in repositories for public, commercial, and private uses has been growing continuously. Quickly finding required contents in the data has become an indispensable technologies [1, 2]. Spoken term detection (STD) is a key technology for performing the task [1, 3]. A common approach to STD is to employ an automatic speech recognition (ASR) system [4, 5] that has provided accurate recognition results for well-resourced tasks [6]. In contrast, in low-resource situations where limited speech data sets and transcriptions are available, it is difficult to build ASR systems.

To address this problem, query-by-example approaches have been reported recently [7, 8, 9]. Of these, the phonetic posteriorgram approach in [7] finds utterances containing posteriorgram sequences similar to a given query posteriorgram sequence using dynamic time warping (DTW). Without any transcription information, the Gaussian mixture model (GMM) posteriorgram approach in [8, 9] performs STD using DTW. These approaches perform *similarity search* for a query spoken term in speech data, where both query and data are represented by posteriorgram sequences.

Similarity search methods have been widely used for a variety of applications. Each utilizes an *index* built from a given data set *off-line* [10, 11]. The methods require some computational costs for building the index before search. However, once the index is built, the method can perform *fast* search owing to the index. The methods are very useful when a lot of similarity search tasks for distinct queries are performed in the same data set. Among them, a neighborhood graph index based method has high versatility in terms of the types of applicable data sets, that is, it is applicable to not only metric spaces but also general spaces with any dissimilarity [12, 13].

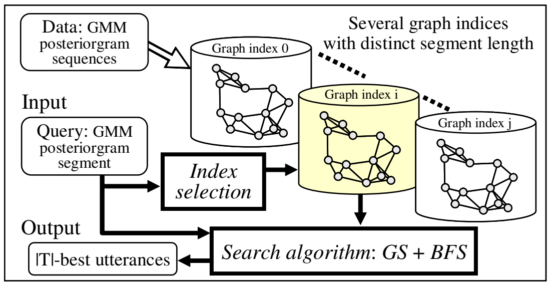

Data: GMM posteriorgram sequences
Several graph indices with distinct segment length
Graph index 0
Graph index i
Graph index j
Input
Query: GMM posteriorgram segment
Index selection
Output
|T|-best utterances
Search algorithm: *GS* + *BFS*

Fig. 1. Graph-index based query-by-example search system.

In this paper, we apply the Graph-based Similarity Search (*GSS*) method to query-by-example STD using DTW on GMM posteriorgrams *for the first time*. Since the existing *GSS* methods cannot be directly used for this application (Section 4), we newly introduce two techniques to a *GSS* method. One is to prepare several graph indices with distinct segment lengths *off-line* and select an appropriate one *on-line* (Section 4.1). The other is to perform graph search with a combination of a greedy search (*GS*) algorithm and a breadth-first search (*BFS*) algorithm (Sections 4.2 and 4.3). A proposed query-by-example STD system based on a neighborhood graph index is shown in Fig. 1, which finds the $|T|$-best utterances (T denotes a set of resultant utterances) from a data set when a query segment (spoken term) is given. We demonstrate in experimental results on the MIT lecture corpus [14] that the proposed method achieves more than 10 times faster operation than a state-of-art method in [9], maintaining the same precision (Section 5).

2. RELATED WORK

We review two research topics: query-by-example STD with DTW on GMM posteriorgrams and similarity search with a graph index.

2.1. STD with DTW on GMM Posteriorgrams

Query-by-example STD systems described in [8, 9] are addressed here. The method in the STD system in [9], which we call *LB*, performs efficient search in a data set of GMM posteriorgrams; this is achieved by reducing the number of DTW-score calculations based on the *lower bound* on a DTW score evaluated *on-line* [9]. Hereinafter, a GMM posteriorgram is simply called a posteriorgram, and a sequence of posteriorgram corresponding to an utterance and its short subsequence are called a posteriorgram sequence and a posteriorgram segment, respectively.

LB finds a set T of the $|T|$-best utterances that contains posteriorgram segments $x = (f_1, \cdots, f_a)$ in a data set similar to a given query posteriorgram segment $q = (f_1^q, \cdots, f_b^q)$, where f_j

$(j=1,2,\cdots,a)$ and f_h^q $(h=1,2,\cdots,b)$ denote posteriorgrams [9]. *LB* first estimates the lower bound on a DTW score with a linear scan in a data set in a similar manner to that described in [15]. The DTW score is calculated within a warping path restricted by the Sakoe-Chiba band [16], whose width is denoted by R. Each local distance in the warping path is defined as $d(f_h^q, f_j) = -\log(f_h^q \cdot f_j)$ [7]. A posteriorgram segment in the data set is determined by the shift of a starting frame by one frame from the previous posteriorgram segment. Next, *LB* evaluates an exact DTW score of the posteriorgram segment whose lower bound is smaller than the current exact DTW score of the posteriorgram segment in the $|T|$th utterance (the $|T|$th-rank score). *LB* terminates when no posteriorgram segment exists whose lower bound is smaller than the $|T|$th-rank score.

The precision of the $|T|$-best utterances can be improved by the score fusion method in [7, 8] that uses multiple queries $Q = \{q_1, \cdots, q_m\}$, where each is an identical *keyword* but a distinct spoken instance (posteriorgram segment). The score fusion method unifies the normalized DTW scores $D(q_i, x)$ between q_i $(i=1,\cdots,m)$ and x by the number of frames (segment length) in q_i, and provides the fusion score $\bar{D}(Q,x)$ as follows.

$$\bar{D}(Q,x) = -(1/\alpha)\log\left((1/m)\sum_{i=1}^{m}\exp\left(-\alpha D(q_i,x)\right)\right), \quad (1)$$

where α denotes a non-negative parameter.

2.2. Similarity Search with a Graph Index

A graph is a general form that represents a relationship between a pair of objects corresponding to vertices with an edge. A neighborhood graph has been utilized as an index for similarity search, combined with several graph search algorithms. A directed k-nearest neighbor (k-*NN*) graph is one such neighborhood graph, where vertex x is connected with a directed edge to vertex y of which dissimilarity from x is within the kth smallest among those from x to other vertices [17, 18, 19, 20]. In contrast, an undirected k-*NN* graph uses an undirected edge instead of a directed edge in the directed k-*NN* graph [13, 21, 12, 22]. In this paper, an undirected k-*NN* graph is focused on and referred to as a k-*NN* graph for simplicity.

A k-*NN* graph constructed from a data set based on a dissimilarity has useful properties for efficient search [12] that are similar to those observed in *small-world networks* [23, 24]: *homophily*, i.e., a tendency of *like to associate with like* [25] and a *very small average shortest path length*. These properties enable a search algorithm to reach a target vertex closest to a given query vertex from an initial vertex with a few steps. To further improve the search efficiency, a degree-reduced k-*NN* graph called a k-*DR* graph has been reported [12, 13], which is a subgraph of the corresponding k-*NN* graph, as described in Section 4.1. A similarity search method using the k-*DR* graph is suitable for performing fast search in a large-scale data set [12], and also can be applied to a wide variety of objects: documents [12], images [22], and GMMs estimated from speech signals [13]. Furthermore, it provides an approximate solution for the nearest neighbor to a given query object [22].

3. PRELIMINARIES

This section formulates a problem and introduces a naïve method (linear scan with caching) to create a baseline for search performance. Besides, Eq. (1) is modified so as to adapt to the problem.

3.1. Problem Formulation

Given a set X of posteriorgram sequences produced from utterance set U, multiple query posteriorgram segments q_i $(i=1,\cdots,m)$ for an identical query *keyword*, the definition of normalized DTW score $D(q_i,x)$ between q_i and x (x is a posteriorgram segment contained in a sequence in X), and the number $|T|$ of utterances required as

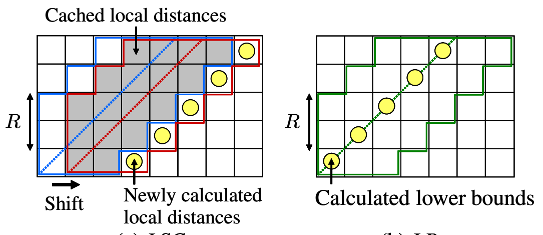


Fig. 2. Local calculations in warping path.

a resultant utterance set T_{q_i} $(|T_{q_i}| = |T| \ll |U|)$, *efficiently* find the $|T|$-best utterances based on a fusion score of the multiple query posteriorgram segments. Then a window of a posteriorgram segment in the data set is shifted *one frame by one frame* and a DTW score is calculated in a warping path restricted by the Sakoe-Chiba band; this setting is identical to that in [9]. The foregoing condition of $|T| \ll |U|$ is quite natural in practical use. For instance, as in the case of listing the 100-best utterances from given 3,000 utterances ($|T|=100$ and $|U|=3,000$).

3.2. Linear Scan with Caching (*LSC*)

A naïve method to solve the problem is to perform a linear scan of the posteriorgram sequences in the data set. Figure 2(a) shows an example of its operation with the query segment length of five and band width $R=2$. Since a window of a posteriorgram segment is shifted one frame by one frame, the difference between a current warping path and the preceding one is only the diagonal path (denoted by the circles in Fig. 2(a)); the remainder (denoted by the gray area) is overlapped. The method, which we call a linear scan with caching (*LSC*), stores the local distances in the overlapped warping path. The number of local-distance calculations in *LSC* is significantly reduced because only the local distances in the diagonal path are newly calculated. Note that the number of the newly calculated local distances is equal to the number of the lower-bound calculations of *LB* shown in Fig. 2(b) in this setting. *LSC* performing a fast linear scan without the lower-bound calculations is adopted as a method providing the baseline for search performance.

3.3. Score Fusion Method

Equation (1) holds under the condition that a resultant *utterance set* for each query q_i is the same, e.g., $T_{q_i} = U$. That is, $T_{q_i} = T_{q_j}$ $(q_i \neq q_j)$ and *permutations* of the sets in terms of the rank order may be different from each other. Since our problem has the restriction of $|T_{q_i}| = |T| \ll |U|$, we need to modify Eq. (1) so that a fusion score can be calculated even when each resultant set contains different utterances $(T_{q_i} \neq T_{q_j})$. Assume that utterance u is contained in the resultant set T_{q_i} for q_i but not in T_{q_j} for q_j $(q_j \neq q_i)$. Although an exact normalized DTW score of u to q_j is unknown, it is clear that the score is at least larger than the $|T|$th-rank score in T_{q_j}. Therefore we can set the $|T|$th-rank score in T_{q_j} for the score of u whose rank order is out of top $|T|$ for q_j. Then Eq. (1) is modified as follows.

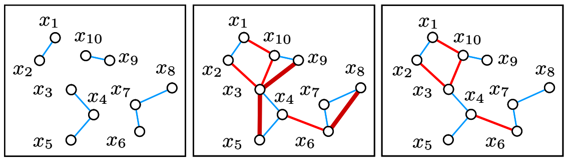
$$\bar{D}(Q,x) = -(1/\alpha)\log\left(F_{Q_{in}} + F_{Q_{out}}\right) \quad (2)$$

$$F_{Q_{in}} = (1/|Q_{in}|)\textstyle\sum_{q\in Q_{in}}\exp\left(-\alpha D(q,x)\right)$$

$$F_{Q_{out}} = \begin{cases} 0 & \text{if } Q_{out} = \emptyset \\ (1/|Q_{out}|)\sum_{q\in Q_{out}}\exp\left(-\alpha D_{max}(q)\right) & \text{otherwise} \end{cases}$$

$$D_{max}(q) = \max\{D(q,x')\},$$

where Q_{in} and Q_{out} denote sets of query posteriorgram segments q, whose element q's resultant set contains the utterance including posteriorgram segment x and does not, respectively, and $D(q,x)$ denotes a normalized DTW score of the utterance in T_q where contains x. Eq. (2) is completely identical to Eq. (1) if $Q_{out}=\emptyset$ like $T_q = U$.



(a) 1-*NN* graph (b) 2-*NN* graph (c) 2-*DR* graph

Fig. 3. Comparison of k-*NN* and k-*DR* graphs.

4. PROPOSED GRAPH-BASED SIMILARITY SEARCH

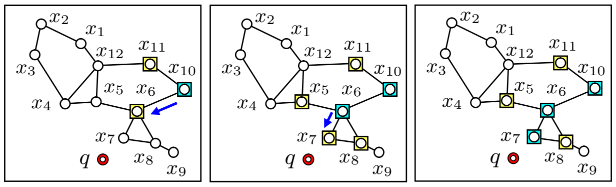
We solve the foregoing problem by a graph-based similarity search (*GSS*) method, that is, by indexing a data set of posteriorgram sequences *off-line* rather than by preprocessing it with various techniques such as *LB on-line*. The existing *GSS* methods, however, cannot be directly applied to the problem. This is because the methods require a valid object with a fixed unit for graph construction while a posteriorgram segment for the object in data posteriorgram sequences is *unknown* before a query posteriorgram segment is given. Besides, the methods are developed for efficiently finding the closest object to a given query or all the objects within a given dissimilarity from the query [22].

We propose a *GSS* method with two newly introduced techniques. One is to prepare several k-*DR* graph indices with distinct segment lengths *off-line* and select an appropriate one from them *on-line*. The other is to apply a combination of the greedy search (*GS*) algorithm with multiple initial vertices [22] and a breadth-first search (*BFS*) algorithm to the selected k-*DR* graph. The *GS* algorithm finds a set of vertices close to a given query, which contains the closest one with high probability [22]. The *BFS* algorithm collects the top-$|T|$ vertices by exploring the neighbors of the vertices found by the *GS* algorithm, where each in the top-$|T|$ vertices corresponds to the best posteriorgram segment contained in each in the top-$|T|$ distinct utterances. Note that only the posteriorgram segment with the best rank order in an utterance is kept if several posteriorgram segments in the same utterance are within the top $|T|$. Thus the *GSS* method can efficiently perform the query-by-example search. Each of the techniques is detailed below.

4.1. k-DR Graph Index

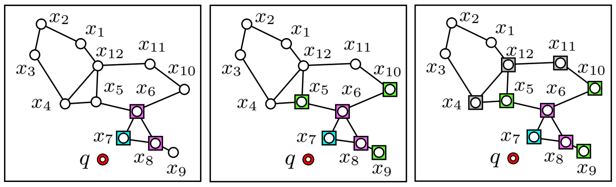
A k-*DR* graph enables a *GS* algorithm (Section 4.2) to operate more efficiently than the corresponding k-*NN* graph (Section 2.2) [12]. This is because the k-*DR* graph has fewer edges than the k-*NN* graph, maintaining the reachability between vertices in the k-*NN* graph. To intuitively understand a k-*DR* graph, we show an example in Fig. 3. The 1-*DR* graph with 10 vertices in Fig. 3(a) is identical to the 1-*NN* graph. Compared with the 2-*NN* graph in Fig. 3(b), the 2-*DR* graph in Fig. 3(c) has three fewer edges (denoted by thick solid lines in Fig. 3(b)), keeping the reachability. For instance, the edge between x_3 and x_5 in the 2-*NN* graph is deleted in the 2-*DR* graph since the *GS* algorithm starting from x_3, which is the second closest vertex from x_5, reaches x_5 via x_4. Thus the edge from vertex x to the kth closest vertex in the k-*DR* graph is deleted if the *GS* algorithm reaches x from the kth closest vertex along the existing edges.

Next, we explain how to construct k-*DR* graphs and select one from them. For graph construction, a query segment length, which is unknown *off-line*, is required. Based on a fact that a query is a spoken term whose segment length is limited from a few dozen to several hundreds, we determine several segment lengths between an estimated maximum and minimum segment lengths. The proposed method constructs several k-*DR* graphs by regarding a posteriorgram segment with the determined segment length as an object with a fixed unit. Given a query posteriorgram segment, the method selects the k-*DR* graph constructed by using the segment length that is a minimum integer exceeding the query segment length.



(a) 1st step (b) 2nd step (c) Termination

Fig. 4. GS operation in 3-*DR* graph with 12 vertices.



(a) 1st depth (b) 2nd depth (c) 3rd depth

Fig. 5. BFS operation in 3-*DR* graph with the 12 vertices.

4.2. Greedy search algorithm with multiple initial vertices

The proposed method first executes a greedy search (*GS*) algorithm to find the closest vertex (target) to given query vertex q. We call the closest vertex to q at the time of a *GS*-algorithm termination an *attractor* (x_a). The *GS* algorithm reaches the target by repeating the selection of the closest adjacent vertex of a current vertex and the movement to it until a current vertex becomes the closest vertex. As an example, the simple *GS* operation is shown in Fig. 4. Given the 3-*DR* graph with the 12 vertices, query q, and initial vertex x_{10}, the *GS* algorithm first evaluates a DTW score between q and x_{10} and does DTW scores of x_{10}'s adjacent vertices x_6 and x_{11}. The *GS* algorithm moves to x_6 closest to q in Fig. 4(a). Next, the *GS* algorithm evaluates vertices x_5, x_7, and x_8 regarding their DTW scores to q, and selects x_7 as the next *expanded vertex* in Fig. 4(b). The *GS* algorithm terminates since the DTW score of the current vertex x_7 is smallest in Fig. 4(c). An attractor is not always the vertex closest to q in all the vertices. It depends only on an initial vertex whether the *GS* algorithm successfully finds the closest vertex to q. To improve an expected search success rate, a *GS* algorithm starting from multiple initial vertices is employed [22]. The *GS* algorithm with L initial vertices ($L > 1$) returns a set of attractors. Then one of the attractors is the closest vertex with a high probability depending on L and graph structural parameter k [22].

4.3. Breadth-first search algorithm

A breadth-first search (*BFS*) algorithm is successively executed after the *GS* algorithm to improve top-$|T|$ vertices closest to q. Figure 5 shows the simple *BFS* operation after the *GS* algorithm terminated in Fig. 4. Assume that a set P of six vertices closest to q is required as the result. The *BFS* algorithm uses the attractor x_7 found by the *GS* algorithm as the root vertex, and then the current P contains $x_7, x_8, x_5, x_6, x_{10}$, and x_{11} in the search path of the *GS* algorithm. First, the vertices x_6 and x_8 at the first depth from the root are candidates for evaluation regarding their DTW scores to q in Fig. 5(a). If a DTW score of a vertex was already evaluated, the evaluation is skipped. If an evaluated DTW score is smaller than those of the current six vertices, P is updated. In Fig. 5(a), the evaluation is skipped. Next, the vertices x_5, x_9, and x_{10} at the second depth are candidates for the evaluation in Fig. 5(b). Then, x_{11} is removed from P and x_9 is added. The procedure is repeated until there is no vertex at the next depth, whose DTW score to q is smaller than that of the top sixth-rank vertex in P. The vertices at the third depth x_4, x_{12}, and x_{11} are the candidates in Fig. 5(c). Then, $P \leftarrow (P \setminus \{x_{10}\}) \cup \{x_4\}$.

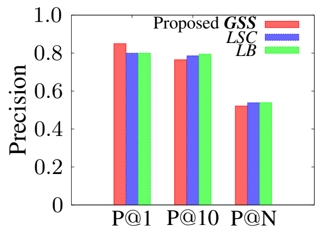


Fig. 6. Comparison of the average precisions of *GSS*, *LSC*, and *LB*.

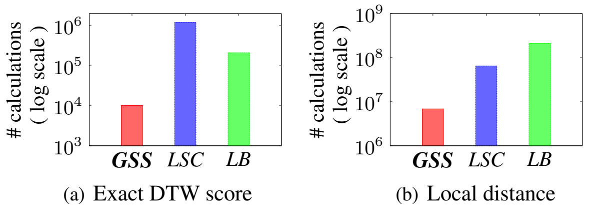


Fig. 7. Computational costs of *GSS*, *LSC*, and *LB*: Average number of (a) exact DTW-score calculations, (b) local-distance calculations.

Since the DTW scores of x_3 and x_1 at the fourth depth are larger than that of any vertex in P, the *BFS* algorithm terminates.

An actual *BFS* algorithm receives a set of attractors found by the *GS* algorithm with multiple initial vertices. The attractor x_a is picked up in ascending order of its DTW score, and newly added to P only if $x_a \notin P$ and its DTW score is smaller than those of vertices in P. When x_a is added to P, the *BFS* algorithm is executed by setting x_a at the root vertex. This procedure is repeated until the *BFS* algorithm uses all the attractors as the root-vertex candidates.

5. EXPERIMENTS

5.1. Experimental Settings

We prepared the MIT lecture corpus [14] for an utterance set that contained 54,581 utterances for a training set and 3,000 utterances for a test set used for search performance evaluation. Although the test-set size is smaller than that in [8], the number of actual posteriorgram segments for search reaches around 1.2×10^6, which is comparable to the size of a large-scale data set. A set of 13-dimensional Mel-Frequency Cepstral Coefficients (MFCCs) was extracted from the utterance set. The sampling rate was 16 kHz and the frame size and shift were 25 ms and 10 ms, respectively. A GMM with 50 mixture components was estimated from the set of the MFCC vectors in the training set. Based on the GMM, a 50-dimensional posteriorgram for each frame in the utterance set was produced.

As queries for search, we picked up 20 *keywords* where for each we used the 10 distinct spoken instances to make the similar setting to that in [8]. For each spoken instance (query segment) of a query *keyword*, we obtained the 100-best utterances that contain segments closest to a query segment regarding a normalized DTW score with the bandwidth $R=6$ as in [8] by the three search methods: proposed *GSS*, *LSC*, and *LB*. For each query *keyword*, we made a list of the utterances sorted in ascending order of the fusion score calculated by using Eq. (2), where $\alpha=0.5$ as in [8].

For the *GSS* method, we constructed 11 distinct *k-DR* graphs from the test set, where the segment lengths were set at $10, 20, \cdots,$ $100, 110$ and the graph structural parameter k was fixed at 100. In the *GS* algorithm with multiple L initial vertices, $L=30$.

5.2. Search Performances

We evaluated the proposed *GSS* comparing with *LSC* and *LB* regarding two performance measures: the precisions of search results and the *online* computational costs. The latter consists of the numbers of

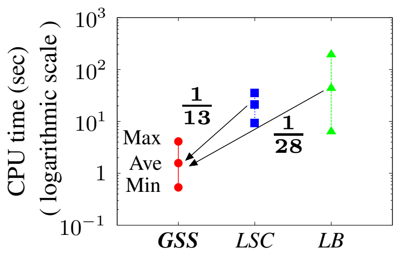


Fig. 8. CPU time of proposed *GSS*, *LSC*, and *LB*.

exact DTW-score and local-distance calculations (Sections 2.1 and 3.2) and CPU time. These results are shown in Figs. 6, 7 and 8.

Figure 6 shows the average precisions of a set of top-X utterances to the ground truth (denoted by P@X, X = 1, 10, N) of *GSS*, *LSC*, and *LB*. The P@X was obtained by using all the precisions of the 20 query *keywords*. The precision of each query *keyword* was calculated by using the results with the score fusion. The values of P@10 and P@N were almost 0.8 and 0.5, respectively. These values are higher than the corresponding values 0.683 and 0.393 shown in [8]. This is because the number of test utterances in our settings were smaller than that in [8]. The tendencies of the average precisions of the three methods were almost the same although the search algorithm in the *GSS* method is an approximate one.

Figures 7(a) and (b) show the average numbers of DTW-score calculations and local-distance calculations with logarithmic scale, respectively. Regarding the average number of exact DTW-score calculations, *LSC*'s number is a baseline because *LSC* performs a linear scan of all the segments in the test set. *LB* reduced the number of the calculations by around one-sixth of the baseline with the lower-bound estimation. The proposed *GSS* drastically reduced the number of the calculations by two orders of magnitude. Furthermore, *GSS* outperformed the others in the average number of local-distance calculations that affected to the CPU time for search. The number of the calculations of *GSS* was only about one-tenth of that of *LSC*.

Figure 8 shows the CPU time for search of each method with logarithmic scale. The CPU time was measured on a computer system equipped with an Intel Xeon E7-4870 2.4 GHz. *LB*'s CPU time is the sum of values for calculating the exact DTW scores and for estimating their lower bounds. The three points of each method in Fig. 8, which are denoted by Max, Ave, and Min, correspond to the maximum, the average, and the minimum CPU time of all the spoken queries. *GSS* operated much faster than *LSC* and *LB* by nearly 13 times and 28 times, respectively. Besides *GSS* achieved the smallest variance of the CPU time among the methods.

6. CONCLUSION AND FUTURE WORK

To speed up a query-by-example spoken term detection method using DTW on GMM posteriorgram sequences, we presented the Graph-based Similarity Search (*GSS*) method. The proposed *GSS* method employs as an index a *k-DR* graph that is appropriately selected from *k-DR* graphs pre-constructed from data posteriorgram sequences *off-line*. The *GSS* method efficiently explores the selected *k-DR* graph with the newly introduced search algorithm; the combination of the *GS* and the *BFS* algorithms. The experimental results on the MIT lecture corpus demonstrated that the *GSS* method accomplished nearly 28 times faster than *LB* using the lower bound on the exact DTW score *on-line*, keeping almost the same precisions.

We will apply the *GSS* method with some extensions to ASR-based STD. Besides the *GSS* method will be performed by parallel processing for faster search just as *LB* has been extended for parallel processing on graphical processing units (GPU) [26].

7. REFERENCES

[1] J. Tejedor, M. Fapšo, I. Szöke, J. H. Černocký, and F. Grézl, "Comparison of methods for language-dependent and language-independent query-by-example spoken term detection," *ACM Trans. Information Systems*, vol. 30, no. 3, August 2012.

[2] F. Metze, N. Rajput, X. Anguera, M. Davel, G. Gravier, C. van Heerden, G. V. Mantena, A. Muscariello, K. Prahallad, I. Szöke, and J. Tejedor, "The spoken WEB search task at MediaEval 2011," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, March 2012, pp. 5165–5168.

[3] W. Shen, C. M. White, and T. J. Hazen, "A comparison of query-by-example methods for spoken term detection," in *Proc. Interspeech*, September 2009, pp. 2143–2146.

[4] C. Chelba, T. J. Hazen, and M. Saraçlar, "Retrieval and browsing of spoken content," *IEEE Signal Process. Mag.*, vol. 25, no. 3, pp. 39–49, May 2008.

[5] C. Parada, A. Sethy, and B. Ramabhadran, "Query-by-example spoken term detection for OOV terms," in *Proc. Int. Workshop on Acoustic Speech Recognition & Understanding.*, 2009, pp. 13–17.

[6] 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*, August 2007, pp. 314–317.

[7] T. J. Hazen, W. Shen, and C. White, "Query-by-example spoken term detection using phonetic posteriorgram templates," in *Proc. Int. Workshop on Acoustic Speech Recognition & Understanding.*, 2009, pp. 421–426.

[8] Y. Zhang and J. R. Glass, "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams," in *Proc. Int. Workshop on Acoustic Speech Recognition & Understanding.*, 2009, pp. 398–403.

[9] Y. Zhang and J. R. Glass, "An inner-product lower-bound estimate for dynamic time warping," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, 2011, pp. 5660–5663.

[10] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, "Searching in metric spaces," *ACM Comp. Surveys*, vol. 33, no. 3, pp. 273–321, September 2001.

[11] G. R. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces," *ACM Trans. Database System*, vol. 28, no. 4, pp. 517–580, December 2003.

[12] K. Aoyama, K. Saito, T. Yamada, and N. Ueda, "Fast similarity search in small-world networks," in *Complex Networks: Int. Workshop on Complex Networks*, R. Menezes et al., Ed. 2009, pp. 185–196, Springer.

[13] K. Aoyama, S. Watanabe, H. Sawada, Y. Minami, N. Ueda, and K. Saito, "Fast similarity search on a large speech data set with neighborhood graph indexing," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, March 2010, pp. 5358–5361.

[14] J. Glass, T. J. Hazen, L. Hetherington, and C. Wang, "Analysis and processing of lecture audio data: Preliminary investigations," in *Proc. HLT-NAACL*, 2004, pp. 9–12.

[15] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.

[16] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Accoustics Speech Signal Process.*, vol. 26, no. 1, pp. 43–49, 1978.

[17] M. T. Orchard, "A fast nearest-neighbor search algorithm," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, 1991, vol. 4, pp. 2297–2300.

[18] T. B. Sebastian and B. B. Kimia, "Metric-based shape retrieval in large databases," in *Proc. Int. Conf. Pattern Recognition*, 2002, vol. 3, pp. 291–296.

[19] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zong, "Fast approximate nearest-neighbor search with k-nearest neighbor graph," in *Proc. Int. Joint Conf. Artificial Intelligence*, 2011, pp. 1312–1317.

[20] J. Wang and S. Li, "Query-driven iterated neighborhood graph search for large scale indexing," in *Proc. ACM Multimedia*, October 2012.

[21] J. Sakagaito and T. Wada, "Nearest first traversing graph for simultaneous object tracking and recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2007, pp. 1–7.

[22] K. Aoyama, K. Saito, H. Sawada, and N. Ueda, "Fast approximate similarity search based on degree-reduced neighborhood graphs," in *Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, 2011, pp. 1055–1063.

[23] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[24] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proc. ACM Symp. Theory of Computing*. SIGACT, May 2000, pp. 163–170.

[25] Ö. Şimşek and D. Jensen, "Decentralized search in networks using homophily and degree disparity," in *Proc. Int. Joint Conf. Artificial Intelligence*, July 2005, pp. 304–310.

[26] Y. Zhang, K. Adl, and J. R. Glass, "Fast spoken query detection using lower-bound dynamic time warping on graphical processing units," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, 2012, pp. 5173–5176.