ZERO-RESOURCE SPOKEN TERM DETECTION USING HIERARCHICAL GRAPH-BASED SIMILARITY SEARCH

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 fast zero-resource spoken term detection (STD) in a large-scale data set, by using a hierarchical graph-based similarity search method (*HGSS*). *HGSS* is an improved graph-based similarity search method (*GSS*) in terms of a search space for high-speed performance. Instead of a degree-reduced k-nearest neighbor (k-*DR*) graph for *GSS*, a hierarchical k-*DR* graph, which is constructed based on a cluster structure in the corresponding k-*DR* graph, is used as an index for *HGSS*. A search algorithm for the hierarchical k-*DR* graph effectively utilizes the cluster structure, resulting in the reduction of the search space. *HGSS* inherits the useful property of *GSS*; it is available for any data sets without limits on a data type nor a defined dissimilarity since a graph is a general expression of a relationship between objects. A vertex and an edge in the hierarchical graph correspond to a Gaussian mixture model (GMM) posteriorgram segment and the relationship between a pair of GMM posteriorgram segments, which is measured by dynamic time warping, respectively. Experimental results demonstrate that *HGSS* successfully reduces the computational cost by more than $40\,\%$ at nearly the same accuracy, compared to *GSS*.

Index Terms— Zero resource, Spoken term detection, Query-by-example search, Neighborhood graph index, Dynamic time warping

1. INTRODUCTION

Spoken term detection (STD) has attracted attention as the volume of speech data stored in repositories has been growing continuously [1, 2]. The predominant approach is to convert the speech data into linguistic representation in advance by automatic speech recognition (ASR), and then to execute search by a text query term [1]. This approach is useful for tasks which are well-resourced in terms of ASR model construction. In contrast, the demand for approaches in low-resource situations has increased recently, where ASR is not available, e.g., STD for a minor language [2]. Some of such approaches are aimed at achieving zero-resource STD that does not require any type of prior linguistic knowledge such as transcriptions, language models, or pronunciation dictionaries [3, 4, 5, 6].

Zero-resource STD, unfortunately, has drawbacks of rather low performance in accuracy and speed in spite of its potential availability. The low accuracy is due to the fact that the zero-resource STD must rely only on the acoustic information since the linguistic knowledge is not available. The low speed comes from the fact that an indexing and a search algorithm are still under development, which are comparable to well-established algorithms for text search employed in ASR-based STD systems.

To improve the accuracy, many efforts have been made regarding the development of an appropriate unsupervised learning algorithm and the selection of both an acoustic feature representation and a definition of a dissimilarity between features [3, 6, 7, 8]. For the high speed performance, a template matching approach based on dynamic time warping (DTW) has been proposed [9, 10, 11, 12, 13]. In particular, the methods reported in [9, 10, 13] share the common feature and dissimilarity; a Gaussian mixture model (GMM) posteriorgram for a feature of each time frame, and the negative logarithm of the probability that two posteriorgrams are generated from the same distribution for a dissimilarity between them [14]. Of these, we focus on the graph-based similarity search method (*GSS*) [13] that uses graph index information, due to its high-speed property and the versatility. Once a graph index is constructed, *GSS* can be performed *very fast*, which is advantageous when there are a lot of similarity search tasks for distinct queries to be performed in the same data set. *GSS* is also applicable to any data sets without limits on a data type nor a dissimilarity [15, 16, 17] (Section 4).

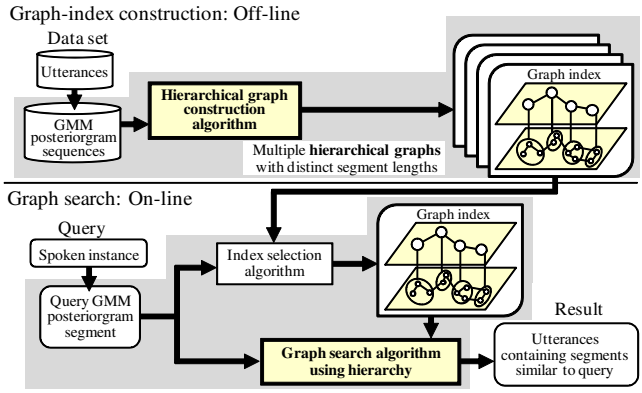


Fig. 1. Hierarchical graph-based similarity search system.

In this paper, we propose a hierarchical graph-based similarity search method (*HGSS*), an improved *GSS*, for further high-speed performance. *HGSS* first constructs a hierarchical degree-reduced k-nearest neighbor graph (k-*DR* graph) as an index from a posteriorgram sequences corresponding to utterances based on the dissimilarity by a novel unique technique (Section 5.1). This technique finds clusters in the k-*DR* graph and stratifies the graph by exploiting the cluster structure. Given a query posteriorgram segment, *HGSS* efficiently finds a set of posteriorgram segments similar to the query from the posteriorgram sequences by a graph search algorithm which limits a search space by exploiting the hierarchy of the k-*DR* graph effectively (Section 5.2). The proposed STD system based on *HGSS* is shown in Fig. 1. We demonstrate in experimental results on the MIT lecture corpus [18] that *HGSS* successfully reduces the computational cost by over 40% than *GSS* at nearly the same accuracy (Section 6).

2. RELATED WORK

We review the three related topics: fast zero-resource STD, similarity search using a neighborhood graph index, and graph clustering.

2.1. Fast Zero-Resource Spoken Term Detection

A method in [9], which we call *LB*, performs fast STD by reducing the number of DTW-score calculations based on the lower bound on a DTW score that is evaluated *on-line*. By applying a piecewise aggregation approximation technique to *LB*, the *LB* achieved the speed-up by almost $30\,\%$ [10]. Most recently, two fast STD methods have

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

been proposed which exploit an index built *off-line* [11, 12]. One utilizes locality-sensitive hashing (indexing and approximately similarity search), and executes a trajectory search to find a similar region in a data set to the query segment [11]. The other method employs a k-means tree as an index [12]. Note that the parameter k in the k-means algorithm denotes the number of centroids in clusters, and differs from graph structural parameter k in the k-*DR* graph in Section 1. Both the methods have the following restrictions on a search space which stem from the index construction algorithms. An object is represented by a feature vector and only a distance can be used, which satisfies the distance axiom. This means that these can not be applied to all data sets, e.g., a data set with a Kullback-Leibler divergence as a dissimilarity. In contrast, the proposed method has no restriction on a data type or a dissimilarity, since a k-*DR* graph, which is used as an index, is a general expression of a relationship between objects and is constructed based on a rank order.

2.2. Similarity Search Using a Neighborhood Graph Index

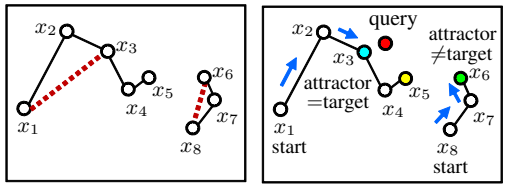
Neighborhood graphs have been used as useful structures in various research fields such as computational geometry, computer vision, pattern classification, and search [19, 20]. In particular, a k-nearest neighbor (k-*NN*) graph has been studied as a search index [15, 21, 22, 23, 24]. A k-*NN* graph has an edge between a pair of vertices x and y if x is among the k closest vertices to y or vice versa. Note that the term "close" means a concept measured by not only a distance but also dissimilarity. A k-*NN* graph has useful properties for similarity search [15] which resemble those observed in *small-world networks* [25, 26]: *homophily*, i.e., a tendency of *like to associate with like* [27] and a *very small average shortest path length*. These properties enable a search algorithm to reach the vertex closest to a given query vertex from an initial vertex chosen at random with a few steps. Moreover, a k-*DR* graph for improving search performance has been applied to a variety of data sets: large-scale documents [15], images [17], GMMs with a Kullback-Leibler divergence as a dissimilarity [16], and GMM posteriorgram sequences [13]. The similarity search using the k-*DR* graph is described in Section 4.

2.3. Graph Clustering

Graph clustering (or community detection) has become an important technique due to the growing demand for analyzing graph data sets from social networks to biological networks [28, 29, 30, 31]. In most of graph clustering techniques, a cluster in a graph is defined as a *dense* subgraph which has more within-cluster edges than between-cluster edges. A partitioning approach finds cluster boundaries by using a graph feature such as *edge-betweenness centrality* and produces clusters by partitioning the whole graph along the cluster boundaries [28]. In contrast, an agglomerative approach joins clusters in pairs by evaluating some measure such as *modularity* from an initial state where each vertex is the sole member of each cluster [29]. Furthermore, faster algorithms in the approach have been proposed [30, 31]. The computational costs which these techniques require are not low yet for a large-scale data set. Our unique technique that produces a cluster structure without an extra high computational cost is detailed in Section 5.1.

3. PROBLEM FORMULATION

We deal with zero-resource STD as the following problem identical to that in [13] (For details to [13]). Given a set of GMM posteriorgram sequences produced from an utterance set, multiple query GMM posteriorgram segments q_i $(i = 1, \cdots, m)$ produced from spoken instances for an identical query *keyword*, a normalized DTW score $D(q_i, x)$ between q_i and x (x is a segment in the GMM posteriorgram sequence), and integer T which denotes the number of resultant utterances, *efficiently* find the T-best utterances based on a fusion score of the multiple query GMM posteriorgram segments.



(a) 2-*DR* and 2-*NN* graph (b) *MSGS* and *BFS* algorithm

Fig. 2. *GSS* operations: (a) Graph construction for the 2-*DR* and the 2-*NN* graph with the eight vertices $(x_1, \cdots, x_8)$. The 2-*DR* graph does not have the two edges depicted by the dashed lines in the 2-*NN* graph. (b) Graph search in the 2-*DR* graph. The *MSGS* algorithm starting at x_1 and x_8 terminates the attractors x_3 (target) and x_6, respectively. The *BFS* algorithm finds x_5 which the *MSGS* did not.

4. GRAPH-BASED SIMILARITY SEARCH

A graph-based similarity search method (*GSS*) for solving the foregoing problem is characterized by the following three points; preconstructing multiple k-*DR* graphs as index candidates for a query segment with any length, selecting an appropriate index from the candidates when a query is given, and performing graph search with a multi-start greedy search and a breadth-first search algorithm. A k-*DR* graph construction and a graph search algorithm, which are subject to improvement in Section 5, are described below.

4.1. k-DR Graph Construction

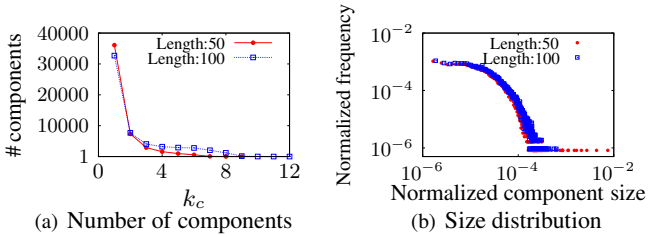
A k-*DR* graph is a subgraph of a k-*NN* graph, and has a smaller degree, i.e., fewer edges of a vertex, than the k-*NN* graph [15] as shown in Fig. 2(a). A k-*DR* graph is recursively constructed by referring to a K-*NN* list ($K \geq k$) that consists of top-K vertices closest to each vertex. A k-*DR* graph construction algorithm first connects each vertex to its closest vertex with an edge, i.e., a 1-*DR* graph is the 1-*NN* graph. For $k \geq 2$, the k-*DR* graph is constructed from the $(k-1)$-*DR* graph by the following rule. An edge from vertex x to its k-th closest vertex $x^{[k]}$ is generated only if a greedy search (*GS*) algorithm starting from $x^{[k]}$ does not reach x. The *GS* algorithm moves a current vertex to the adjacent vertex closest to a query vertex after evaluating the dissimilarities between the query and all the adjacent vertices. By this rule, the k-*DR* graph with fewer edges achieves the same reachability by the *GS* algorithm as the k-*NN* graph. Since reducing the degree leads to decreasing the evaluation cost, the k-*DR* graph enables the *GS* algorithm to work efficiently.

4.2. Graph Search

Top-T vertices closest to a query vertex are found in the constructed k-*DR* graph by a combination of two algorithms; a multi-start greedy search (*MSGS*) and a breadth-first search (*BFS*) algorithm [16].

The *MSGS* algorithm plays a role of finding a *target*, which is the closest vertex to a query, at a high search success rate. To achieve the high search success rate, the *MSGS* algorithm starts from multiple initial vertices chosen at random because each *GS* algorithm terminates at an *attractor*, which is a current vertex closer to the query than its adjacent vertices, regardless of whether the attractor is the target as shown in Fig. 2 (b). The *MSGS* algorithm collects a tentative set of the top-T vertices (a tentative top-T set) on search paths, and provides both the tentative top-T set and the attractors to the successive *BFS* algorithm.

The *BFS* algorithm determines the top-T set by updating the tentative top-T set as follows. The *BFS* algorithm repeatedly sets an attractor in the tentative top-T set at a root (initial) vertex in ascending order of its dissimilarity to the query as shown in Fig. 2(b). Each iteration terminates if the tentative top-T set is not updated at a current depth.



(a) Number of components (b) Size distribution

Fig. 3. Characteristics of connected components that are generated in graph construction process in the case of two distinct GMM posteriorgram segment lengths, 50 and 100: (a) number of connected components with k_c, (b) normalized component size distribution.

5. PROPOSED HIERARCHICAL GSS

Our goal is the speed-up of *GSS* to accomplish fast zero-resource STD in a large-scale data set. The main idea is to limit a search space, i.e., the number of vertices that are subject to the dissimilarity calculation, by exploiting a *latent cluster structure* in a k-*DR* graph for *GSS*. In the setting described in Section 3, several consecutive posteriorgram segments (vertices) in an utterance are often similar (close) to each other. These vertices form a dense subgraph (cluster) described in Section 2.3. By regarding the dense subgraph as one vertex, a supernode, we can construct a hierarchical k-*DR* graph. We first introduce a unique clustering technique, and then detail an *MSGS* algorithm and a *BFS* algorithm using a cluster structure.

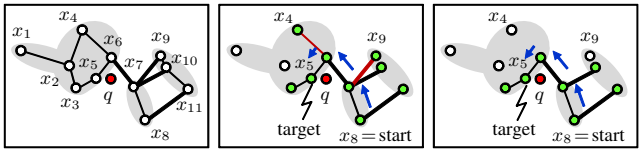
5.1. Clustering for Hierarchical Graph Construction

We focus attention on connected components that are generated in the k-*DR* graph construction process described in Section 4.1. A k-*DR* graph is constructed by increasing the graph structural parameter k_c one by one from 1 to k. In this process, vertices are agglomerated with k_c and form several connected components. We utilize the connected components in the k_c-*DR* graph as clusters rather than those obtained by the existing graph clustering techniques, which are applied to the completed k-*DR* graph, since there is no need for an additional operation.

Figure 3(a) shows the foregoing process when we used as a vertex two distinct lengths of the posteriorgram segments, 50 and 100, where the number of segments is almost 1×10^6. Suppose that $k_c=0$ (Not illustrated in Fig. 3). Then, the graph has no edge, and the numbers of both the isolated components and the vertices in the graph are the same. Next, when k_c increases to 1, each vertex is connected to its closest vertex, and many connected components appear. At some value of k_c, each and every vertex in the graph become a member of a one and only connected component. The growth of the components with k_c depends on the data set, and is shown in Fig. 3(a).

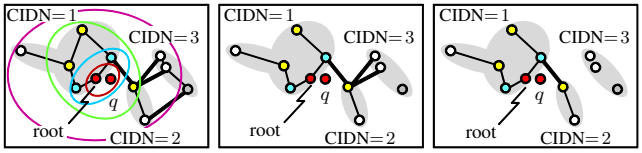
We exploited the connected components that were generated when $k_c = 1$ since the number of the connected components and the component size distribution were suitable to similarity search. It is desired for our efficient search that a graph has an appropriate number of uniformly-sized connected components, i.e., no giant component. Figure 3(b) shows the distribution of the connected component sizes normalized by the number of all the segments when $k_c=1$. We observed that the maximum normalized component size was less than one-hundredth even when the segment length was short such as 50 and there is no giant component in the 1-*DR* graph.

We assign a cluster identification number (CIDN) to each connected component (cluster), and attach a CIDN to the vertex that belongs to the corresponding cluster with the CIDN. We regard a set of vertices whose CIDNs are the same as a supernode, and use the edges in a k-*DR* graph as the edges between the supernodes, i.e., maintain the edge structure. Thus we can construct a hierarchical k-*DR* graph based on the connected components (clusters) generated with $k_c=1$. Hereafter, we call k_c the clustering parameter.



(a) 3-*DR* graph (b) *GS* algorithm (c) Improved *GS*

Fig. 4. *GS* operations in the 3-*DR* graph with the 11 vertices denoted by x_i, $i=1,\cdots,11$. The clusters are illustrated with the gray areas and the query is denoted by q. The improved *GS* algorithm starting at x_8 in (c) omits the dissimilarity calculations of x_4 and x_9 while the *GS* in (b) executes them.



(a) 3-*DR* graph (b) *BFS* algorithm (c) Improved *BFS*

Fig. 5. *BFS* operations in the same setting as Fig. 4. CIDNs are assigned to the three clusters each and the *BFS* algorithm starts at the root. The depths from the root are illustrated with the contour lines in (a) and the vertices in the same depth are given the same color. The improved *BFS* algorithm in (c) does not evaluate the vertices in the cluster with CIDN$=3$ while the *BFS* in (b) does. Note that the termination condition is not used because of too few vertices.

5.2. Graph Search Using Cluster Structure

A graph search algorithm in *HGSS*, which consists of the *MSGS* and the *BFS* algorithm using a cluster structure, reduces the number of dissimilarity calculations by selecting vertices based on their CIDNs. For simplicity, we explain the *GS* algorithm instead of the *MSGS* algorithm, and only the *BFS* algorithm starting at the target.

The conventional *GS* algorithm sets the next current vertex at the closest vertex to a query in all the adjacent vertices of a current vertex. To determine the closest vertex, *all the adjacent vertices* are evaluated regarding their dissimilarities to the query. In contrast, if there are more than one adjacent vertex that shares the same CIDN, *only the closest vertex* to the current vertex among the adjacent vertices is evaluated in the improved *GS* algorithm in *HGSS*. Only if a current vertex is an attractor, the *GS* algorithm evaluates the remainders in each cluster, i.e., all the adjacent vertices. The *GS* algorithm terminates if no adjacent vertex is closer to the query, i.e., the current vertex is the attractor again. The limited dissimilarity evaluation leads to the reduction of the computational cost. For intuitive understanding of the difference between the *GS* and the improved *GS* algorithm, their comparison is shown in Fig. 4.

The conventional *BFS* algorithm requires a high computational cost because it calculates all the vertices at each depth from a root (target). The improved *BFS* algorithm evaluates only the vertices whose CIDNs are the same as those of the vertices within the *depth of two* from the target. There are two reasons for selecting the depth of two. One comes from the k-*DR* graph structure. Because of the graph construction rule in Section 4.1, the k'-th closest vertex ($k' \leq k$) to the target is not always connected to the target directory. Instead of only the adjacent vertices, we regard the vertices at the depth of two as the candidates. The other is inspired by a *small-world network* like [32]. A *small-world network* has the tendency that a neighbor's neighbor (a vertex at the depth of two) is also likely to be a neighbor as described in Section 2.2. This means that a vertex at the depth of two may be close to the target. We regard the vertices with the same CIDNs as those at the *depth of two* as the candidates. Thus, the improved *BFS* algorithm limits the search space. The operation of the improved *BFS* algorithm is shown in Fig. 5.

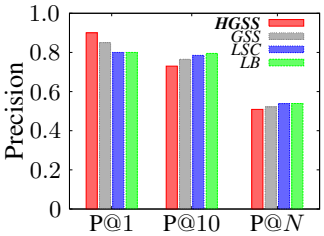


Fig. 6. Average precisions of *HGSS*, *GSS*, *LSC*, and *LB*.

6. EXPERIMENTS

6.1. Experimental Setup

We used the MIT lecture corpus [18] for an utterance set to carry out search performance evaluation. The utterance set consisted of 54,581 utterances for a training set and 3,000 utterances for a test 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 the frame 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. A 50-dimensional posteriorgram for each frame in the utterance set was produced from the GMM.

For STD, we chose 20 keywords and picked up 10 distinct spoken instances for each keyword as the query from the training set. This was done to set up the similar conditions to those in [5] and [13]. We employed a DTW score normalized by a query segment length as the dissimilarity, and obtained the 100-best utterances which contained the most similar 100 distinct segments to the query segment. In the DTW, the warping path was limited within the bandwidth $R = 6$ as in [5]. The 100-best-utterance lists, where each list consisted of 100-best utterances for each query segment, were merged into a single list for each keyword. The utterances in the merged single list were sorted in ascending order of the fusion score calculated likewise to previous studies; it was done in the same manner as [13] and adopted the fusion parameter $\alpha = 0.5$ equally to [5].

We compared the proposed *HGSS* with three existing methods: *GSS* [13], *LSC* [13], and *LB* [9]. The comparison was carried out regarding two search performance criteria; the accuracy and the *online* computational costs, i.e., speed. The accuracy was evaluated by the average precisions of the sets of top-X utterances to the ground truth for all the keywords; $X = 1, 10, N$, where N denotes the number of the correct utterances. The speed was measured by three ways: the number of the DTW score calculations, the number of local dissimilarity calculations, and CPU time. We performed the experiments on a computer system equipped with an Intel Xeon E7-4870 2.4 GHz.

In the *HGSS* and the *GSS* method, we employed 11 distinct k-*DR* graphs constructed from the test set. The k-*DR* graph's 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 was set at 30. Furthermore, for the *HGSS* method, we set the clustering parameter k_c as $k_c = 1$.

6.2. Search Performance: Accuracy and Speed

The proposed *HGSS* accomplished the zero-resource STD at around twice the speed and also at nearly the same accuracy as the *GSS*. Besides, the *HGSS* operated much faster than both *LSC* and *LB* by almost 25 times and 50 times, respectively.

Figure 6 shows the average precisions of the sets of top-X utterances, which are denoted by P@X, $X = 1, 10, N$, of *HGSS*, *GSS*, *LSC*, and *LB*. The average precisions of the four methods were nearly the same at each P@X.

Figures 7(a) and (b) show the average numbers of DTW-score calculations and local-dissimilarity calculations with *logarithmic scale*, respectively. Compared to the *GSS*, the proposed *HGSS* reduced the numbers of the DTW-score calculations and the local-dissimilarity calculations by 45%, respectively. These numbers were much smaller than those of *LSC* and *LB* by one order to two orders of magnitude. *HGSS* selectively calculates a DTW-score between a query segment (query vertex) and a segment (vertex) in an utterance by using the segment's CIDN. In the experiments, the vertex-selection strategy in the *BFS* algorithm was, in particular, useful to reduce the search space. Consequently, the number of the DTW-score calculations in the *BFS* algorithm decreased, and the required CPU time was reduced.

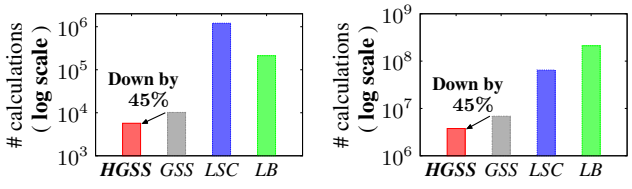


Fig. 7. Average number of (a) exact DTW-score calculations, (b) local-dissimilarity calculations of *HGSS*, *GSS*, *LSC*, and *LB*.

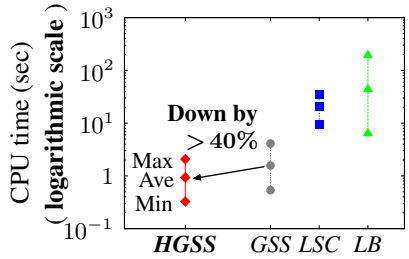


Fig. 8. CPU time of *HGSS*, *GSS*, *LSC*, and *LB*.

Figure 8 shows the CPU time, which the four methods required, with *logarithmic scale*. 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. Owing to the reductions of the number of the DTW-score calculations and the local-dissimilarity calculations, the *HGSS* successfully reduced more than 40% from the average CPU time (Ave) required by the *GSS* which was the best method of the existing three.

7. CONCLUSION AND FUTURE WORK

To accomplish fast zero-resource spoken term detection (STD) in the large-scale data set, we developed a hierarchical graph-based similarity search method (*HGSS*). Compared to the conventional *GSS*, the proposed *HGSS* achieved the zero-resource STD at around twice the speed and also at nearly the same precisions; P@1, P@10, and P@N. This is because the graph search algorithm successfully limited the search space by effectively exploiting the clusters in the hierarchical k-*DR* graph constructed from the data set.

HGSS can be combined with any method for extracting acoustic features or determining a dissimilarity between features since a graph is a general expression of a data set with a dissimilarity and the hierarchical k-*DR* graph is constructed based on a rank order regarding a dissimilarity. If such a method achieving high accuracy is developed, a zero-resource STD system with the combination of the method and *HGSS* can accomplish high speed and accuracy at once.

We will develop a graph clustering algorithm that generates a structure appropriate to search, e.g., a structure where cluster sizes are uniform, instead of the proposed technique using a k_c value controlled by the graph construction algorithm. Besides, search algorithms will be developed which utilize the cluster structure more effectively.

8. REFERENCES

[1] 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.

[2] 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. Inform. Syst.*, vol. 30, no. 3, August 2012.

[3] F. Metze, X. Anguera, E. Barnard, M. Davel, and G. Gravier, "The spoken WEB search task at MEDIAEVAL 2012," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, May 2013, pp. 8121–88125.

[4] A. Jansen, E. Dupoux, S. Goldwater, M. Johnson, S. Khudanpur, K. Church, N. Feldman, H. Hermansky, F. Metze, R. Rose, M. Seltzer, P. Clark, I. McGraw, B. Varadarajan, E. Bennett, B. Borschinger, J. Chiu, E. Dunbar, A. Fourtassi, D. Harwath, C.-Y. Lee, K. Levin, A. Norouzian, V. Peddinti, R. Richardson, T. Schatz, and S. Thomas, "A summary of the 2012 JHU CLISP workshop on zero resource speech technologies and models of early kanguage acquisition," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, May 2013, pp. 8111–8115.

[5] 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.

[6] C.-A. Chan, C.-T. Chung, Y.-H. Kuo, and L.-S. Lee, "Toward unsupervised model-based spoken term detection with spoken queries without annotated data," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, May 2013, pp. 8550–8554.

[7] S. Soldo, M. Magimai.-Doss, J. Pinto, and H. Bourlard, "Posterior features for template-based ASR," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, May 2011, pp. 4864–4867.

[8] H. Wang, T. Lee, C.-C. Leung, B. Ma, and H. Li, "Using parallel tokenizers with DTW matrix combination for low-resource spoken term detection," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, May 2013, pp. 8545–8549.

[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] Y. Zhang and J. R. Glass, "A piecewise aggregate approximation lower-bound estimate for posteriorgram-based dynamic time warping," in *Proc. Interspeech.*, 2011, pp. 1909–1912.

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

[12] G. Mantena and X. Anguera, "Speed improvements to information retrieval-based dynamic time warping using heirarchical k-means clustering," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, May 2013, pp. 8515–8519.

[13] K. Aoyama, A. Ogawa, T. Hattori, T. Hori, and A. Nakamura, "Graph index based query-by-example search on a large speech data set," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, May 2013, pp. 8520–8524.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] J. W. Jaromczyk and G. T. Toussaint, "Relative neighborhood graphs and their relatives," *Proc. IEEE*, vol. 80, no. 9, pp. 1502–1517, September 1992.

[20] G. Karypis, E.-H. S. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, August 1999.

[21] 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.

[22] 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.

[23] 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.

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

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

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

[27] 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.

[28] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Natl. Acad. Sci.*, vol. 99, no. 12, pp. 7821–7826, June 11 2002.

[29] M. E. J. Newman, "Fast algorithm for detecting community structures in networks," *Phys. Rev. E*, vol. 69, 066133, 2004.

[30] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, 066111, 2004.

[31] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech.*, P10008, 2008.

[32] W. Dong, M. Charikar, and K. Li, "Efficient K-nearest neighbor graph construction for generic similarity measures," in *Proc. Int. Conf. World Wide Web*, 2011, pp. 577–586.