SPEED IMPROVEMENTS TO INFORMATION RETRIEVAL-BASED DYNAMIC TIME WARPING USING HIERARCHICAL K-MEANS CLUSTERING

Gautam Mantena[1,2], Xavier Anguera[1]*

[1]Telefonica Research, Edicio Telefonica - Diagonal 00, 08019 Barcelona, Spain
[2]International Institute of Information Technology - Hyderabad, India
gautam.mantena@research.iiit.ac.in, xanguera@tid.es

ABSTRACT

With the increase in multi-media data over the Internet, query by example spoken term detection (QbE-STD) has become important in providing a search mechanism to find spoken queries in spoken audio. Audio search algorithms should be efficient in terms of speed and memory to handle large audio files. In general, approaches derived from the well known dynamic time warping (DTW) algorithm suffer from scalability problems.

To overcome such problems, an Information Retrieval-based DTW (IR-DTW) algorithm has been proposed recently. IR-DTW borrows techniques from Information Retrieval community to detect regions which are more likely to contain the spoken query and then uses a standard DTW to obtain exact start and end times. One drawback of the IR-DTW is the time taken for the retrieval of similar reference points for a given query point. In this paper we propose a method to improve the search performance of IR-DTW algorithm using a clustering based technique. The proposed method has shown an estimated speedup of 2400X.

Index Terms— Spoken term detection, audio search, query by example , indexing, retrieval.

1. INTRODUCTION

Query by example spoken term detection (QbE-STD) task is to detect a spoken query in spoken audio data (or reference data). With the increase in multi-media data over the Internet, a need for searching in audio databases has increased. One of the key aspects for QbE-STD is to enable voice search in very large multi-lingual data in real time.

QbE-STD task was previously attempted by first converting audio into a sequence of symbols and then performing a text based search. In [1, 2, 3], audio is first converted into sub-word like units using an automatic speech recognizer (ASR) and lattice-based search techniques are incorporated to further increase the performance of the system. These type of methods have become very popular as they enable fast retrieval by using text based search techniques.

One of the major issues of ASR based techniques, is the availability of labelled data for training the Hidden Markov Models (HMM). To overcome this problem, zero resource pattern matching techniques [4, 5, 6, 7] have been proposed. These techniques use similarity matrices (sparse/full) in detecting the spoken term. A popular technique for searching spoken queries in audio data is the segmental-DTW [4] which is slow and require a lot of memory as it computes a full similarity matrix. In [7, 8] faster variants of segmental-DTW were proposed to visit multiple cells in the similarity matrix only once, but the problem to handle large audio data still exists. In this paper we compare the performance results with the subsequence-DTW (S-DTW) [9] by using the implementation proposed in [7].

Alternative techniques have been proposed by [5] and [10] to search in large audio files in real time. [5] use randomized hashing algorithms in building a sparse matrix and then use image processing techniques in detecting the most likely matching segment.

IR-DTW technique [10] uses a fusion of information retrieval and DTW approaches. In IR-DTW, no similarity matrix needs to be constructed as the algorithm performs a sequential scan of the spoken query and accumulates information about the partial matching paths between the query and the audio reference data. IR-DTW then performs DTW alignment on the detected paths to get an accurate start and end points. In [10], it has been shown that IR-DTW is memory efficient as compared to S-DTW. One drawback of IR-DTW is that it uses an exhaustive search over all the reference data for a given query.

In this paper we propose an Indexing and Retrieval based approach using hierarchical K-Means clustering to speedup the IR-DTW. A similar kind of approach was used in large scale image recognition [11] and object and scene retrieval from videos [12]. The proposed algorithm shows as estimated speedup of 2400X and is faster than the one proposed in [13] using randomized hashing algorithms and image processing techniques, which has a speedup of 800X.

Section 2 gives a brief description of the IR-DTW. In section 3 we propose the indexing and retrieval of reference features (reference data) using hierarchical K-Means for IR-DTW and section 4 shows the performance of the system as compared to IR-DTW with exhaustive search and S-DTW.

2. INFORMATION RETRIEVAL-BASED DTW (IR-DTW)

The IR-DTW algorithm described in [10], borrows techniques from the Information Retrieval community and traditional real-valued dynamic programming algorithms to perform the matching between a spoken query and spoken reference audio. Its main advantages are the requirement of a small memory footprint and allowing for a fast implementation, as shown in this paper.

Most DTW-based algorithms like those proposed in [4, 7] build similarity matrices of size N x M, where N, M are the length of the query and the reference audio. This is a big limitation when requiring to search on big databases. Instead, IR-DTW performs the matching just using a vector memory structure, thus reducing the amount of total memory required, regardless of the size of the query

*Gautam Mantena performed this work while at Telefonica Research

and the reference data. In addition, by indexing the reference data (as shown in this paper) a considerable speedup is obtained. Next we briefly review the IR-DTW algorithm.

Let $\mathcal{Q} = \{q_1, q_2, \ldots, q_i, \ldots, q_N\}$ be a query audio file containing N feature vectors, and $\mathcal{R} = \{r_1, r_2, \ldots, r_j, \ldots, r_M\}$ be a reference audio file containing M feature vectors. The main steps of the IR-DTW are shown in Algorithm 1

Algorithm 1 Information Retrieval-based Dynamic Time Warping

Input: $\mathcal{Q}$, $\mathcal{R}$ time series, maxQDist parameter
Output: $\mathcal{P}$ set of matching paths

```
ΔT ← ⊘, P ← ⊘
for all q_i ϵ Q do
    R' ← best_points(R, q_i) {STEP 1}
    for all r_j ϵ R' do
        match_point ← {tq_i, tr_j, d(q_i, r_j)}
        ΔT ← InputMatch(match_point, maxQDist) {STEP 2}
    end for
end for
for all k ϵ ΔT do
    P ← P ∪ process&extract(ΔT[k]) {STEP 3}
end for
```

IR-DTW performs a sequential scan of all the query points q_i in $\mathcal{Q}$. In *STEP 1* of Algorithm 1 a search is performed for all the reference points $\mathcal{R}' \subseteq \mathcal{R}$ that are similar to a given query q_i. This can be done using an exhaustive comparison of all query-reference pairs or, as proposed in this paper, using a fast retrieval of the reference points (*STEP 1* in algorithm 1) using a hierarchical K-Means clustering based indexing and retrieval approach. From this point on, only matching pairs are used to find final matching subsequences. For this reason, a system parameter *maxQDist* is introduced as the maximum allowed distance between any two matching points within a good matching subsequence.

From the retrieved reference points $\mathcal{R}'$, a *match_point* element is defined as a triplet composed of tq_i, tr_j and $d(q_i, r_j)$. tq_i and tr_j are the time offsets of the query and the reference data measured from the start of their respective temporal sequences. $d(i, j)$ is the distance measure between query q_i and reference r_j points, given by a cosine similarity in logarithmic scale. The core of the IR-DTW algorithm (in *STEP 2* of Algorithm 1) combines all *match_point* elements into matching subsequences of low overall distance between query and reference that follow some local and global warping constraints.

In *STEP 2* of Algorithm 1 a vector data structure called $\triangle$T is used to store the matching subsequences formed by concatenating the *match_point* elements. The location in $\triangle$T where subsequences are created and updated is based on the offset calculated from the time stamps of query and reference points in each *match_point* element. For every query-reference matching point $t_{offset} = tr_j - tq_i$.

Would there be a diagonal match between the query and some portion in the reference, t_{offset} would be constant across that region in the reference. Due to the variation in the speaking rate and acoustic conditions this is seldom true, even for the same speaker. Following what is done in a standard DTW implementation, in the IR-DTW algorithm a range of values around t_{offset} are checked to find the best matching subsequence to contain each *match_point* element. The range of locations is given by $t'_{offset} = [t_{offset} - WRange, t_{offset} + WRange]$, where $WRange = \frac{maxQDist}{2}$ for a warping constraint allowing for one subsequence to be as much as double the other.

The input *match_point* element is appended to the subsequence found to be the best among those within range. This decision can be taken based on the normalized score (similar to DTW) or on the subsequence length. Instead, if there is no existing subsequence at any location in the range defined by t'_{offset} then the *match_point* element is used to create a new subsequence at location t_{offset}.

Once a sequential scan of all the query points is complete, $\triangle$T contains all the possible matching subsequences between the query and the reference data. At this point we perform *STEP 3* to select those subsequences with a minimum length (set here to half of the query length) and the net score below a defined threshold. Such a threshold is applied to remove any spurious paths that might have been detected.

Although the IR-DTW is already finished and all matching subsequences have been found, in practice a standard DTW algorithm is then applied in the detected regions to get an accurate start and end time stamps and a score dependent on all frame pairs (not only those selected as *match_point* elements). A more detailed description of the IR-DTW algorithm is given in [10].

Due to the sequential scan of the query q_i and building the partial matches, IR-DTW is memory efficient as compared to S-DTW. While *STEP 1* can be performed through exhaustive search, doing so would have an important overhead in terms of computation. In section 3 we propose an indexing and retrieval approach to IR-DTW to obtain much greater performance in terms of speed.

3. INDEXING AND RETRIEVAL USING HIERARCHICAL K-MEANS

In this paper, we focus on an indexing and retrieval technique for obtaining the reference points for a given query. In standard IR-DTW we compute distances for a given query q_i with all the reference points and compare the scores to a given threshold θ to select the closest reference points. When dealing with large scale databases, computing distances for all the reference points is time consuming and is a hindrance for real time systems.

In this paper, we propose an indexing type of approach to overcome the problems with the exhaustive search. We use a hierarchical K-Means clustering based approach to cluster all the reference data points. A given query point q_i traverses the tree to retrieve the closest reference points. In this approach the query point $q_i$is no longer compared with each of the reference points but only with a few cluster centers. Cluster centers which are closest to the query point q_i, based on a threshold, are selected and all the reference points belonging to those clusters are retrieved. A detailed description of the indexing and retrieval is given in sections 3.1 and 3.2.

3.1. Indexing the reference vectors

The reference points are indexed using hierarchical K-Means. Each node in the tree corresponds to a cluster. Each cluster can be further divided into sub-clusters based on some criteria. All the leaf nodes contain the indices of the reference points. The goal is to index the data in the form of a tree where similar reference points lie within the same node (or cluster).

Splitting each of the nodes in the tree is based on the number of points present in the given node. This is to make sure that the nodes do not contain too many reference points, which would result in increased false alarms.

In building the hierarchical K-Means tree, we define *K* which is the maximum number of children a given node can have (provided it satisfies the splitting criteria). Each of the inner nodes will have

a minimum of 2 children and a maximum of *K*. For simplicity K-Means initialization was done using random selection of the data points.

At times, a given node has a large number of data points but the cluster is very dense to make any further partitions. For such clusters, the clustering algorithm is performed again by choosing *K=2*. For these nodes, the first center initialization is randomly selected and the second center is chosen by selecting a point farthest from the first. This is to ensure that the clusters are as small as possible. A minimum cluster size of 200 reference points is considered to make sure that the clusters have some minimum number of points.

Although the initial building of the hierarchical K-Means clustering and indexing of the reference data is done offline, the standard K-Means is quite slow on large databases. We have used a variant of K-Means clustering called mini-batch K-Means algorithm [14] to solve this problem. In mini-batch K-Means clustering for each iteration we randomly select a subset of data and perform a standard gradient descent method to converge to a local minimum. In [14], it was shown that mini-batch K-Means converged to a local minimum with several orders of magnitude faster than the standard K-Means.

In our implementation of the hierarchical K-Means for a large collection of reference points, we have randomly selected 10,000 reference points to perform the gradient descent approach. This selection and updating of the centers was done for 100 iterations and whenever the number of reference points fall below 10,000 a standard K-Means was performed.

3.2. Fast retrieval of reference vectors

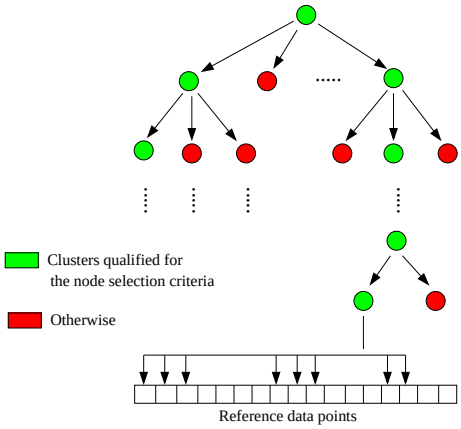

Fig. 1. Tree traversal in hierarchical K-Means tree to retrieve the reference data.

All the reference vectors are indexed using a hierarchical K-Means tree. In the tree data structure the leaf nodes contain the indices for all the reference data vectors. Tree traversal is based on the same thresholding condition, θ, used in algorithm 1. Each internal node of the tree only stores the centers obtained by K-Means clustering and the width of the cluster (node). Let q_i be a query point and c_j be any cluster center and the distance between them is given by $d(q_i, c_j)$. We also define the width of a node c_i as the maximum distance of the cluster center and the points lying inside the given cluster, and is represented as $w(c_i)$. The node selection criteria for retrieval is as shown in equation 1

$$\frac{d(q_i, c_j)}{w(c_j) + \theta} \leq \beta \tag{1}$$

where β is called the overlapping factor. The smaller the value of the overlapping factor the more likely the selection of the node will be. β plays a role in maintaining a balance between speed and accuracy. As shown in tables 2 and 4, smaller values of β indicate a strict selection process of the node and increases the speed of IR-DTW while larger values of β show better performance.

As shown in figure 1, breadth first traversal is performed on the hierarchical K-Means tree to obtain the required leaf nodes. For a given query q_i, whenever the condition for node selection fails, all its children nodes are rejected and are not traversed. All the leaf nodes that have passed the node selection criteria are selected and the reference points belonging to these leaf nodes are retrieved to populate the $\triangle$T. During retrieval we do not compute the distances with any of the reference points but only verify with the cluster centers of the leaf nodes using the node selection criteria.

4. EXPERIMENT AND RESULTS

The proposed algorithm is evaluated using MediaEval 2012 data which is a subset of Lwazi database [15]. The data consists of audio recorded via telephone in four of 11 South African languages available in the Lwazi database. We have considered two data sets, development (dev) and evaluation (eval) with a common set of query audio data for both of them. The statistics of the audio data is as shown in table 1

Data	# Utts	Total (h)	Average (sec)
dev	1580	3.69	8.42
eval	1660	3.87	8.40
query	200	0.08	1.47

Table 1. Statistics of the development (dev), evaluation (eval) and audio query used to validate the performance of Hierarchical K-Means based IR-DTW.

All the evaluations were performed using the NIST evaluation [16] criteria and the corresponding max term weighted values (MTWV) are reported. A more detailed information on the evaluation is provided in [17].

39 dimensional mel-frequency cepstral coefficients (MFCC) were extracted with 25 ms window length and 10 ms window shift. These 39 dimensional MFCC were used to train 128 modified Gaussian mixture model (GMM) [7]. The MFCC's are then transformed to their corresponding Gaussian posteriorgrams. All the experiments were performed using such Gaussian posteriorgrams.

For the K-Means clustering the maximum value of *K* was set to 8. To avoid clusters with very few points, the minimum number of points in a cluster is set to 200 data points. A constant frame threshold of $\theta = 4$ was considered for all the experiments. The results reported in tables 2 and 3 using S-DTW, exhaustive and tree-based IR-DTW were implemented in Python[1] using Numpy package [18].

In table 2, MTWV scores indicate that the IR-DTW with exhaustive search performs better than the S-DTW, and that the proposed tree-based IR-DTW shows a similar performance to that of exhaustive IR-DTW for $\beta = 0.5$.

In standard IR-DTW we do an exhaustive search for all the reference points whose similarity measure is less than θ. Using hierarchical K-Means we retrieve all the points in a given node that satisfy the condition given in equation 1. Table 2 scores show that the

[1] http://www.xavieranguera.com/resources/TREE-IR-DTW.zip

Exp.	S-DTW	IR-DTW	tree-based IR-DTW		
			$\beta = 0.4$	$\beta = 0.5$	$\beta = 0.6$
dev	0.336	0.364	0.295	0.364	0.339
eval	0.337	0.357	0.269	0.334	0.335

Table 2. Maximum term weighted values (MTWV) of S-DTW, exhaustive IR-DTW and tree-based IR-DTW for various values of β.

tree-based IR-DTW for $\beta = 0.5$ performs better than for $\beta = 0.6$ because for $\beta = 0.6$ there are a lot of reference points which are not similar to a given query point. In the proposed method, unlike the exhaustive search, some of the reference points retrieved have a distance measure for a given query q_i greater than θ. This is because we do not validate any reference points but only verify whether the cluster center of the leaf node passes the node selection criteria or not.

To validate the performance of the system we need to define the terms coverage, relevance and miss percentage:

Coverage: Percentage of reference points retrieved from the complete reference data set.

Relevance: Percentage of reference points retrieved which are relevant, i.e. the percentage of retrieved reference points whose distance from a given query q_i is below a defined threshold θ.

Miss: Percentage of reference points which are closer to the query but have not been selected. This occurs when a particular cluster fails the node selection criteria but has reference points close to the given query point.

Table 3 shows the average coverage, relevance and miss percentages computed for the dev and eval data for the various β values.

A lower coverage percentage indicates that fewer reference points have been retrieved which results in a speedup as there are less points to be inserted into $\triangle$T. Low coverage reflects a high relevance percentage because many points retrieved from the leaf node cluster are closest to the query and subsequently giving a high miss percentage and resulting in a poor performance. A similar explanation holds for high coverage resulting in a poor performance in terms of accuracy and speedup. From tables 2 and 3, we conclude that tree-based search performs better for $\beta = 0.5$. Even though the proposed approach is missing 44.9% of the points for $\beta = 0.5$ it is still able to find the matching paths.

β	Coverage	Relevance	Miss Percentage
0.4	2.76%	96.12%	60.97%
0.5	4.59%	84.06%	44.91%
0.6	6.54%	69.53%	34.85%

Table 3. Average coverage and relevance scores computed for the dev and eval data sets.

	$\beta = 0.6$		$\beta = 0.5$		$\beta = 0.4$	
Exp.	R'	Alg.1	R'	Alg.1	R'	Alg.1
dev	869	1.5	1200	2.2	1840	4.4
eval	832	1.5	1224	2.5	1670	3.8

Table 4. Speed improvements for retrieval and in Algorithm 1 using tree-based over exhaustive search IR-DTW

In table 4, for each β value we compute R' and Alg.1 to compare the computation time of tree-based IR-DTW over exhaustive where R' is the ratio of computation time in retrieving the reference points (*STEP 1*) and Alg.1 is the ratio of total computation in retrieving the reference points and updating the matching paths in $\triangle$T (*STEP1* and *STEP 2* in Algorithm 1). There is a very big improvement in the computation time as we are only comparing the query q_i with the cluster centers to retrieve all the relevant points.

In table 4, lower values of β indicate a lower coverage percentage resulting in the speedup in the retrieval of the reference points and subsequently a speedup in the algorithm 1. On the other hand, low values of $beta$ have a higher miss percentage resulting in a poor performance and is shown in table 2.

Algorithms	Speedup	
	dev	eval
S-DTW (Python)	13	14
IR-DTW (Python, exhaustive search)	7	7
IR-DTW (Python, tree-based search)	21	20
S-DTW (C++)	630	680
IR-DTW (C++, exhaustive search)	840	860
IR-DTW (C++, tree-based search, estimated)	2400	2450

Table 5. Speedup of S-DTW, exhaustive IR-DTW and tree-based IR-DTW ($\beta = 0.5$) in Python and C++ implementation.

The speedup for IR-DTW is computed by the ratio of database size (in sec.) and search time (in sec) per size of queries. Table 5 shows that the estimated speedup of tree-based IR-DTW is faster than the one proposed in [13] using randomized hashing algorithms and image processing techniques, which has a speedup of 800X. Based on the increment shown for the Python implementation, the estimated speedup for the tree-based C++ implementation is 2400X.

5. CONCLUSIONS AND FUTURE WORK

With the increase in multi-media data over the Internet, QbE-STD has become important in providing a search mechanism to find spoken queries in a spoken audio. Due to the limitations of training Hidden Markov Models for automatic speech recognition for low-resource languages, zero resource pattern matching techniques were proposed. In general, approaches derived from the well known DTW algorithms suffer from scalability problems. The proposed method has shown an estimated speedup of 2400X.

Recently in [10] it has been shown that IR-DTW outperforms the standard S-DTW in terms of memory (more than 10-fold). The memory performance of IR-DTW has shown promising results and is applicable for performing QbE-STD over large audio databases. In this paper we proposed a hierarchical K-Means clustering based approach to increase the performance of IR-DTW over exhaustive search.

The number of tree leaves are important in determining the coverage, relevance and miss percentage during retrieval of the nodes. Due to the compactness of Gaussian posteriorgrams node splitting was not possible beyond 200 clusters and as future work we plan on looking at how to effectively split clusters with a high density of very similar points.

6. REFERENCES

[1] I. Szoke, M. Fapso, L. Burget, and J. Cernocky, "Hybrid word-subword decoding for spoken term detection," in *Workshop on Searching Spontaneous Conversational Speech*, 2008.

[2] Murat Saraclar and Richard Sproat, "Lattice-based search for spoken utterance retrieval," in *HLT-NAACL*, 2004, pp. 129–136.

[3] David R. H. Miller, Michael Kleber, Chia-Lin Kao, Owen Kimball, Thomas Colthurst, Stephen A. Lowe, Richard M. Schwartz, and Herbert Gish, "Rapid and accurate spoken term detection," in *INTERSPEECH*, 2007, pp. 314–317.

[4] Yaodong Zhang and James R. Glass, "Unsupervised spoken keyword spotting via segmental DTW on gaussian posteriorgrams," in *ASRU*, 2009, pp. 398–403.

[5] Aren Jansen and Benjamin Van Durme, "Efficient spoken term discovery using randomized algorithms," in *ASRU*, 2011, pp. 401–406.

[6] Chun-an Chan and Lin-Shan Lee, "Unsupervised spoken-term detection with spoken queries using segment-based dynamic time warping," in *INTERSPEECH*, 2010, pp. 693–696.

[7] Xavier Anguera, "Speaker independent discriminant feature extraction for acoustic pattern-matching," in *ICASSP*, 2012, pp. 485–488.

[8] Vikram Gupta, Jitendra Ajmera, Arun Kumar, and Ashish Verma, "A language independent approach to audio search," in *INTERSPEECH*, 2011, pp. 1125–1128.

[9] Meinard Muller, *Information Retrieval for Music and Motion*, chapter Dynamic Time Warping, pp. 69–84, Springer, 2007.

[10] Xavier Anguera, "Method and system for improved pattern matching," Patent EP12382508, Telefonica Research, Barcelona, Spain, 2012.

[11] David Nister and Henrik Stewenius, "Scalable recognition with a vocabulary tree," in *Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2161–2168.

[12] Josef Sivic and Andrew Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *International Conference on Computer Vision*, 2003.

[13] Aren Jansen, Benjamin Van Durme, and Pascal Clark, "The JHU-HLTCOE spoken web search system for MediaEval 2012," in *MediaEval*, 2012.

[14] D. Sculley, "Web-scale k-means clustering," in *International Conference on World Wide Web*, 2010, pp. 1177–1178.

[15] Etienne Barnard, Marelie H. Davel, and Charl Johannes van Heerden, "ASR corpus design for resource-scarce languages," in *INTERSPEECH*, 2009, pp. 2847–2850.

[16] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddington, "Results of the 2006 spoken term detection evaluation," in *Workshop on Searching Spontaneous Conversational Speech*, 2007, pp. 45–50.

[17] Florian Metze, Etienne Barnard, Marelie H. Davel, Charl Johannes van Heerden, Xavier Anguera, Guillaume Gravier, and Nitendra Rajput, "The spoken web search task," in *MediaEval*, 2012.

[18] Paul F. Dubois, Konrad Hinsen, and James Hugunin, "Numerical Python," *Computers in Physics*, vol. 10, no. 3, 1996.