MEMORY EFFICIENT SUBSEQUENCE DTW FOR QUERY-BY-EXAMPLE SPOKEN TERM DETECTION

Xavier Anguera and Miquel Ferrarons

Telefonica Research, Edificio Telefonica-Diagonal 00, Barcelona, Spain
xanguera@tid.es

ABSTRACT

In this paper we propose a fast and memory efficient Dynamic Time Warping (MES-DTW) algorithm for the task of Query-by-Example Spoken Term Detection (QbE-STD). The proposed algorithm is based on the subsequence-DTW (S-DTW) algorithm, which allows the search for small query sequences of feature vectors within a much longer reference sequence by considering fixed start-end points in the query and discovering optimal matching subsequences within the reference. The proposed algorithm applies some modifications to S-DTW that make it better suited for the QbE-STD task, including a way to perform the matching with virtually no system memory, optimal when querying large scale databases. We also describe the system used to perform QbE-STD, including an energy-based quantification for speech/non-speech detection and an overlap detector for matches. We test the system proposed using the Mediaeval 2012 spoken-web-search dataset and show that, in addition to the memory savings, the proposed algorithm brings an advantage in terms of matching accuracy (up to 0.235 absolute MTWV increase) and speed (around 25% faster) in comparison to the original S-DTW.

Index Terms— One, two, three, four, five

1. INTRODUCTION

Query-by-Example (QbE) algorithms are used to search for audio patterns within audio documents by using audio examples. Within a Spoken Term Detection (STD) task the example is a speech query, which is searched for within a speech corpus to retrieve all the realizations of that query, regardless of the speaker that uttered them or the recording acoustic conditions. Although this task could be approached by first decoding the data into phoneme lattices and then searching for the decoded query phonetic string, doing so would require prior knowledge about what is being decoded and searched for, which is not always available. Recently, a family of algorithms based on pattern matching techniques have shown their effectiveness in performing this task while not using any prior knowledge of the language, or of the content being spoken nor any pre-trained acoustic models [1].

Most pattern matching algorithms inherit from the well known Dynamic Time Warping (DTW) algorithm [2] where an optimal non-linear alignment is found between two sequences of feature vectors through dynamic programming techniques. Although Hidden Markov Models (HMM) made DTW obsolete in the 80's by using statistical processes to learn acoustic properties from many labelled examples, recently DTW-based algorithms are becoming again useful for zero-resource settings, where no (or very little) labeled data is available for training. Among the first to propose novel adaptations to DTW is the Segmental-DTW algorithm first introduced in [3] for the task of unsupervised word pattern discovery in speech. The Segmental-DTW algorithm applies successive DTW steps within overlapping parallelogram-shaped global constraints on the data to find matching sequences of acoustic frames. In [4] the authors improved upon the previous work by using posterior probabilities from a Gaussian Mixture Model (GMM) as features, useful when comparing multiple speakers. A variation of such posterior features, as described in [5], have been used in this work. Also using posteriors, [6] modifies the DTW algorithm for QbE. The segmental DTW algorithm has also been used for the QbE task by forcing the start-end points in the query, although it can be quite slow to compute.

In [7], the authors propose a QbE system using a variation of the DTW algorithm and the use of an image algorithm in a post-processing step. Also using image processing for improved performance, [8] proposes a spoken term discovery system that has been later applied to QbE in [9]. This system has been designed for application to large scale databases, by using information retrieval and image processing algorithms to make the search very fast, which has traditionally been one of the major drawbacks of DTW systems. Alternatively, in [10] QbE is performed by comparing audio at the segment level, where the segmentation is obtained through agglomerative hierarchical clustering of similar acoustic frames.

The subsequence-DTW algorithm (S-DTW) initially proposed by Müller in [11] for QbE on music stands out from the previous due to its simplicity and effectivity. In S-DTW a standard DTW is applied which does not penalize insertions in the first and last frames of the query when analyzing any query-reference position. As we will see in the experimental section, the original S-DTW is already quite competitive with regard to computational complexity when compared to most of the approaches mentioned above, with the exception of [8]. Still, S-DTW and many other algorithms are quite memory inefficient. For example, for a query of length N and a reference of length M, S-DTW involves storing a matrix of NxM values, which can become prohibitive when M is quite large. Note that although the main goal in QbE amounts to simply finding the locations (and maybe the score) of all reference segments matching a query, these algorithms additionally output the exact alignment between matching sequences, at the expense of considerably increasing the memory required. We will see that by not storing such an alignment we will be able to dramatically reduce the memory required.

In this paper we propose a memory efficient implementation of the S-DTW that does not return the unnecessary frame-by-frame alignment between query and found reference sequences, but only outputs the matching start-end points and their matching scores. The algorithm proposed (which we call MES-DTW for Memory Efficient S-DTW) uses a sliding window to compute the optimum alignment and just requires the storage of around $Nx2$ values in memory at a time. In addition, in this paper we also propose a speedup to S-DTW and experiment with two different normalization techniques that are able to boost S-DTW performances. As a result, the proposed MES-

DTW is shown to be much faster than many pattern-matching implementations while requiring very little memory. These properties make MES-DTW suitable for application in embedded devices and for searching on mid-to-large scale databases.

In addition, we also describe the QbE-STD system that allows for the search of an input query signal in a reference speech database by using the MES-DTW algorithm. The system is composed of a feature extraction front end to extract posteriorgram features, a speech/non-speech detector that uses a quantization-based thresholding, the MES-DTW algorithm and an overlap detector to postprocess all matches corresponding to very similar matching sequences.

The remaining of this paper is structured as follows. First, in Section 2, we review the S-DTW algorithm, as it is the baseline of our proposed algorithm. Then, Section 3 describes the three improvements of S-DTW that conform the proposed MES-DTW algorithm. Then we describe the QbE system that uses the MES-DTW algorithm in Section 4. Finally, in Section 5 we evaluate the system performance and in Section 6 some conclusions are drawn.

2. SUBSEQUENCE DYNAMIC TIME WARPING

In this section we review the S-DTW algorithm first introduced by Müller in [11] as a modification to the classical DTW algorithm to find matching subsequences between a query feature sequence $X := (x_1, x_2, \dots, x_N)$ and a reference feature sequence $Y := (y_1, y_2, \dots, y_M)$ where N and M are the length of each sequence and usually $N \ll M$. This algorithm is suitable to be used in query-by-example applications, where the start-end points of the query are fixed to the extremes of X, and one wants to find the subsequence in Y (i.e. $Y(a^* : b^*) := (y_{a^*}, y_{a^*+1}, \dots, y_{b^*})$) that optimally matches X. More specifically, we are looking for the locations (a^*, b^*) in Y, where $1 \leq a^* \leq b^* \leq M$, that satisfy equation 1 where $DTW()$ is the global time-warped distance between two sequences of feature vectors.

$$(a^*, b^*) := \underset{(a,b):1 \leq a \leq b \leq M}{\arg\min} (DTW(X, Y(a : b))) \tag{1}$$

S-DTW finds the optimal solution to equation 1 by applying a standard DTW where insertions in the alignment paths are not penalized when they appear at the beginning or the end of X. The S-DTW algorithm implements this in two steps. First, the accumulated cost matrix ($D(n, m)$ where $1 \leq n \leq N$ relates to frames in X, and $1 \leq m \leq M$ relates to frames sin Y) is modified so that $D(1, m) := c(x_1, y_m)$ for $1 \leq m \leq M$, where $c()$ indicates the distance between a given query frame and a reference frame. Such modification allows the dynamic programming process to consider all points $(1, m)$ as possible starting points for the optimal matching subsequence. Then, once we have computed the entire matrix $D(n, m)$, the optimum matching subsequence ending point b^* is selected following equation 2.

$$b^* := \underset{b \epsilon [1:M]}{\arg\min} D(N, b) \tag{2}$$

This second step also differs from the classic DTW implementation in that we allow the optimum matching subsequence to finish in any position along the reference subsequence that follow equation 2. This is equivalent to a DTW implementation where the cost of insertion in the first and last query rows is 0. Once the last point in the matching subsequence (N, b^*) has been found, [11] performs a trackback dynamic programming process (inverse to that used to find b^*) to find the optimum starting point $(1, a^*)$ and therefore the subsequence $Y(a^*, b^*)$ optimally matching X.

Using S-DTW for query-by-example applications where multiple matching subsequences might be found in reference Y for a given query X is straightforward and also covered by [11]. Once the first matching subsequence has been found, all values around point (N, b^*) in the last query row in matrix $D(n, m)$ are set to a high value, and equation 2 is run again to find the next best ending point (N, b_2^*). The process is then repeated iteratively while the accumulated distance of selected matching subsequences remains below a given threshold τ, $D(N, b_i^*) < \tau$. In our implementation of the algorithm we ignore all points between the selected local minima and adjacent local maxima.

3. MEMORY EFFICIENT SUBSEQUENCE DYNAMIC TIME WARPING

In this section we describe the memory efficient dynamic time warping algorithm (MES-DTW), derived from the S-DTW algorithm described in Section 2. The three main novelties of MES-DTW w.r.t. S-DTW are the use of a lookup table to perform a fast trackback to determine the value of a^* once b^* is found, two alternative normalizations of matching paths, and a memory efficient implementation that dramatically reduces the required system memory. Next we describe each of these novelties in detail.

3.1. Matching Subsequence Fast Trackback

In the original definition of S-DTW, upon finding the optimum subsequence ending point (N, b^*) we need to perform a dynamic programming trackback through the cumulative cost matrix $D(n, m)$ to find the optimum starting point a^*. This step can be easily sped up by storing the alignment steps taken locally by the forward dynamic programming process in an additional matrix $S(n, m)$ so that the trackback can be performed without recomputing the comparisons between nodes. Alternatively, the processing can be further reduced by instead storing in $S(n, m)$ the optimal starting points in Y at each step of the dynamic programming process, so that once b^* is found, one just needs to lookup in $S(N, b^*)$ the value of a^*. Note that this modification to the algorithm requires that the matrix $S(n, m)$ be stored in memory, which can become a burden in certain cases. We will see, though, that by using the memory efficient implementation explained below, this memory increase is greatly reduced.

3.2. Matching Path Normalization

When applying dynamic programming techniques to find the optimal alignment between two sequences, each local alignment decision is taken based on the comparison of the cumulative cost values of nearby points, chosen according to predefined local constraints. In this setting, diagonal alignments are favored as less steps are required to align both sequences when these are constrained to contain the whole query sequence. In order to control the importance of each alignment path some research [12] has addressed the application of weights together with local constraints. Similarly, in [13] a local normalization of points by their path lengths is made before the selection of the best local alignment. Alternatively, the final distance of the matching paths can be globally normalized by the matching length in order to make the result comparable among matches of different lengths.

In S-DTW the normalization of scores is quite relevant as paths starting in different starting points compete for the optimal alignment at each point, where diagonal alignments are also favored. Unlike in [11] where no normalization is applied, we experiment in this paper with both local and global normalizations. At the local level we normalize the cumulative costs before deciding the local alignment. At the global level we normalize the final alignment so that global distances are comparable between different-length sequences. Note that the theoretical assumptions that make dynamic programming suitable for the alignment of two sequences are still valid when applying such normalizations over the cumulative distance matrix.

In this paper we use the same local constraints as proposed in [11], i.e. single-frame insertion, deletion and alignment. In this setting, local normalization modifies the constraints as shown in equation 3, where $D(n,m)$ is the accumulated cost matrix and $C(n,m)$ stores the length of the best alignment path leading to each point (n,m). In this work we measure the path length by using the Manhattan distance.

$$D(n,m) = min \begin{cases} \frac{D(n-1,m)+d(n,m)}{C(n-1,m)+1} \\ \frac{D(n-1,m-1)+d(n,m)}{C(n-1,m-1)+2} \\ \frac{D(n,m-1)+d(n,m)}{C(n,n-1)+1} \end{cases} \tag{3}$$

Note how this normalization requires the storage of an additional matrix $C(n,m)$ to keep track of the path length for each point. In section 3.3 we see how to minimize the effect of this memory increase.

Alternatively, a global normalization is performed by using non-normalized local constraints and normalizing the cumulative cost for $n = N$ as shown in equation 5 before performing the search for the optimum ending point b^*. This is equivalent to the score we obtain at that location by applying local normalization, except that local decisions here have been taken without any normalization. In fact, when applying a global normalization we can avoid building a distance matrix $C(n,m)$ as we can obtain the normalizing factor as $C(N,m) = N + (b^* - a^*)$.

$$D_{norm}(N,m) = \frac{D(N,m)}{C(N,m)} \tag{4}$$

3.3. Memory Efficient Implementation

The implementation of any DTW-based algorithm usually involves computing first a similarity matrix between all frames in the two sequences being compared and then performing dynamic programming steps to find the optimal alignment between the sequences. In S-DTW this typical approach requires the storage of the accumulated cost matrix $D(N,M)$ in memory, occupying $N \cdot M$ real-valued positions. In addition, in order to speedup the trackback of the best matching sequence start a^* for an optimum ending point b^* we create matrix $S(n,m)$ with an extra $N \cdot M$ values. Finally, if a local normalization by the path length is to be applied, yet another $N \cdot M$ values need to be stored for matrix $C(n,m)$. Overall, the amount of memory required if the two modifications proposed above are applied is around $3 \cdot N \cdot M$ Which can cause a burden in the memory required to run S-DTW if M (reference data) is very big.

Given that for a typical QbE search the goal is usually only to find the optimum locations (a^*, b^*) in Y and their alignment score, there is no need to store the local alignment steps or the values to track them back once the optimum ending points have been found for the whole reference sequence. We can obtain the exact same results by storing in memory only the values needed to make local decisions and to find the optimum starting point a^* once the optimum ending point b^* has been found. We achieve this through the processing of the dynamic programming through a sliding window on the reference data, as shown in Algorithm 1.

Algorithm 1 Memory Efficient Subsequence DTW

```
Input: $X, Y$ sequences of feature vectors
Output: $Y(a_i^* : b_i^*); i = 1 : K$ matching subsequences
  Initialize vectors $D'_1 \leftarrow \infty, C'_1 \leftarrow 0, S'_1 \leftarrow 0$
  for $m = 1$ to $M$ do
    for $n = 1$ to $N$ do
      if $n == 1$ then
        $D'_2(1) \leftarrow d(n,m), C'_2(1) \leftarrow 1$
      else
        Apply local constraints to $D'_2(n)$ {equation 3}
      end if
    end for
    if $D'_1(N)/C'_1(N)$ is a local minima then
      Consider match at $b^* = m$ and retrieve $a^* = S(N, b^*)$
    end if
    swap vectors $D'$, $C'$ and $S'$
  end for
  return top K matching subsequences
```

To find matching subsequences using MES-DTW, instead of the full accumulated cost matrix $D(n,m)$, we require only the storage of two vectors corresponding to the two columns in $D(n,m)$ involved in the local decisions $D'_{1:2} = D(n, m-1 : m)$. The same applies to the $C(n,m)$ and $S(n,m)$ matrices. In addition, we also store the single value $D(N, m-2)$ in order to decide on the existence of a local minima in $D'_1(N)$.

Initialization of the algorithm is done by setting $D'_1 \leftarrow \infty, C'_1 \leftarrow 0, S'_1 \leftarrow 0$. This is similar to what is done in [11] for an extended row 0 in the accumulated cost matrix D. Then, for each processing step (i.e. each reference frame) we apply the local constraints shown in 3, except for $n = 1$ where all paths are set to start from (i.e. no insertion penalty is applied).

After each step we compare the ratio $D'_1(N)/C'_1(N)$ to its neighboring normalized values (all with $n = N$) and whenever we find a local minima we consider that frame to be an ending point b^* of an optimum matching subsequence. As our objective is to return the best K matching subsequences, at this step we just store the triplet $m, S'_1(m), D'_1(m)$ for later processing. Before we move onto the next reference frame we swap the contents of both vectors (optimally implemented through memory pointers) so that D_1, C_1 and S_1 contain data for the step we just finalized and D_2, C_2 and S_2 become available to be filled in the next step.

After all processing of the reference data has been performed, we rank the stored triplets according to the matching scores obtained and return the K-results with the smallest values.

Over all, the required memory for the dynamic programming step with all proposed modifications is roughly $3 \cdot 3 \cdot N$, which is significantly smaller that the memory required in the “full-memory” version, which would be roughly $3 \cdot N \cdot M$.

4. QUERY-BY-EXAMPLE SPOKEN TERM DETECTION SYSTEM

One of the most common current applications of the subsequence-DTW algorithm is for Query-by-Example (QbE) Spoken Term Detection (STD) where a given acoustic query is searched for within a corpus of reference audio recordings. The objective consists in determining with high accuracy in which locations in the corpus each query appears. Figure 1 shows the steps followed to perform QbE-STD using the subsequence matching algorithm proposed above.

First, we extract MFCC-39 (13 cepstral + 13 derivatives + 13 accelerations) features from the acoustic data (both reference and queries) in a standard way (10ms scroll in 25ms window). Then, like in standard ASR-based systems, we account for the non-speech frames present in the reference and query data. Every query will usually have an unknown amount of non-speech both at the beginning and the end, in the case of a single-word query, and anywhere else for multiple words. It is desirable to detect and not consider such frames in the matching, as non-speech frames always obtain high matching scores and can turn the system unusable. In order not to depend on external data to pre-train an acoustic model to detect such frames, we implement an energy-based system that is trained on the reference data itself. Note that in a QbE-STD application the reference data is considered known by the system before performing the test with unknown queries, therefore training better acoustic models with this data should not be considered over-fitting.

The speech/non-speech model is trained by using only the energy values of the signal. First, we gather the 10% of acoustic frames with lowest energy from our reference dataset. With these frames we train a one Gaussian non-speech model and with the rest we train a 4-Gaussian speech model. Then we iteratively decode and retrain the models using the same data. This usually increases the number of frames in the silence model to around 30%, as we do not impose any minimum speech or non-speech duration. We stop after 20 iterations or when the difference in number of frames between two consecutive iterations is small. We store the Gaussians in the speech model ordered by their mean energy.

Queries and reference data are then labeled using this model. Instead of just labeling each frame with the most likely model, we record which of the Gaussian mixtures is the closest one. This produces a frame-level energy-based labeling with label 0 for non-speech and 1 through 4 for speech frames. This was done because we found a big correlation between frame energy and matching errors. Before performing any matching between query and reference data we eliminate those frames below a certain level (from 1 to 4). The higher the level, the fewer the frames that will be matched, and therefore the system will run faster and with more reliable data, but also more false alarms can result when queries become too small. In our experiments we eliminated all frames assigned to levels 0 and 1.

Next, 128-dimensional posterior probabilities are obtained from speech frames. Posterior probabilities have been successfully used in pattern matching for some time [4]. They have been shown to be a robust representation of the acoustic signal when attempting to match audio from different speakers. Several methods have been proposed in the literature to obtain the posterior probabilities. These include posteriors obtained from Gaussian Mixture Models (GMM) [4], from phonetic decoders [7] or from models derived automatically from the data through acoustic segment models [14]. In our system we use Gaussian posteriors obtained from a modified GMM model [5] trained on all available reference data (i.e. development and testing data). This modified GMM training combines the use of EM and K-means iterations in order to maximize the discovery and separation of automatically generated acoustic regions in the acoustic space.

Comparison between any two feature vectors is done by means of the distance proposed in equation 6. Note that we do not apply any post-processing to the posterior probability features (e.g. imposing a minimum posterior value) like in [6], as we found no advantage in doing so.

$$d(\mathbf{x}, \mathbf{y}) = -\log\left(\frac{\mathbf{x}\cdot\mathbf{y}}{||\mathbf{x}||||\mathbf{y}||}\right) \tag{5}$$

Posterior-probability features are the input to a DTW-based algorithm, which can either be the MES-DTW, S-SDW or others. In the current implementation the reference data is first loaded into memory and then each query (much smaller in size) is used to retrieve the K-best subsequences of the reference matching the query. Note that in QbE-STD applications the number of matches is not known, and can range from none to several hundreds, depending on the length of the reference. Depending on the particular use-case scenario that the algorithm is applied to one needs to decide on the value of K. Given that in this paper we are not interested in any particular use-case, but in measuring retrieval accuracy, we set K to a high value (i.e. $K = 1000$) and later apply a score thresholding on the final resulting matches to find the optimum balance between precision and recall.

Given a list of reference subsequences optimally matching the query, the next step is to post-process this list to eliminate subsequences that are highly overlapping. Even though the algorithms described in this paper forbids for multiple ending points b^* to be very close to each other, in reality there are still multiple matching subsequences being returned by the algorithm. This is due to the nature of speech, where nearby acoustic frames are acoustically very similar. In order to detect overlaps we apply equation 7 on two reference matching subsequences $Y(a_1^* : b_1^*)$ and $Y(a_2^* : b_2^*)$ to obtain the overlap ratio.

$$ovl(Y(a_1^* : b_1^*), Y(a_2^* : b_2^*)) = \frac{min(b_1^*, b_2^*) - max(a_1^*, a_2^*)}{min((b_1^* - a_1^*), (b_2^* - a_2^*))} \tag{6}$$

Whenever the ratio is greater than 0.5 we consider the two sequences to be in overlap and eliminate the one with highest average distance to the query.

The last step in the process involves inserting back the eliminated non-speech frames in order to get accurate start-end times in the reference dataset.

5. EXPERIMENTAL SECTION

In this section we show the results of comparing the proposed MES-DTW and the QbE system to the S-DTW algorithm and another state of the art algorithm in QbE search. To do so, we used the Mediaeval 2012 SWS evaluation datasets and metrics [15]. The database consists of around 7.5 hours of telephone recordings from 4 different African languages, created as a subset of the Lwazi database [16] and split into a 3.6h development set and 3.8h evaluation set. A set of 100 development and 100 different evaluation queries are used to query the data, where multiple instances of each of the queries might exist, produced by different speakers. As a result, the system needs to return the exact locations of found matches, and their scores. The metrics used to compare results are the MTWV (Minimum Term Weighted Value) and the real-time speedup factor. The MTWV was initially used in the NIST 2006 Spoken Term Detection evaluation [17]. Like in the Mediaeval 2012 evaluation, the scoring parameters of the MTWV metric were modified to reduce the impact of false alarms in the results and give more importance to missed matches. The real-time speedup factor computes the average time speedup obtained when searching for a 1 second query within the reference database compared to manually listenning to the whole database. Experiments were conducted using an Ubuntu virtual machine with 2 assigned cores and 8GB of RAM over a computer with 2xIntel Xeon CPUs 2.5GHz and 96Gb of DDR3 RAM 1333MHz.

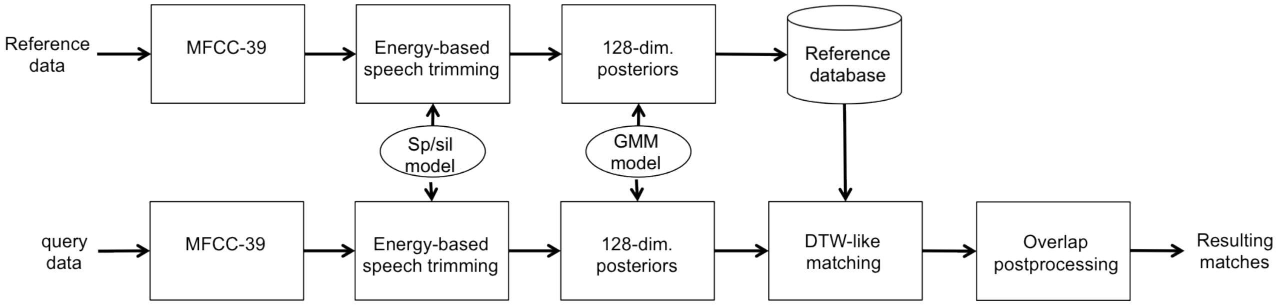

Fig. 1. *General system blocks for SWS task*

Table 1. Evaluation of modifications to S-DTW (development set)

System	MTWV	Speedup
S-DTW [11]	0.169	897
S-DTW + origin matrix	0.169	903
S-DTW + local norm	**0.404**	834
MES-DTW + local norm	**0.404**	1165
S-DTW + global norm	0.393	919
MES-DTW + global norm	0.393	**1287**
RAILS [9]	0.381	800[2]

Table 2. Evaluation of modifications to S-DTW (evaluation set)

System	MTWV	Speedup
S-DTW [11]	0.243	1033
S-DTW + origin matrix	0.243	1045
S-DTW + local norm	0.375	956
MES-DTW + local norm	0.375	1272
S-DTW + global norm	**0.394**	1006
MES-DTW + global norm	**0.394**	**1377**
RAILS [9]	0.384	n/a

5.1. Performance analysis

Tables 1 and 2 show the results obtained with the development and evaluation sets respectively. We compare the original S-DTW as proposed in [11] with the modifications that derive into our proposed MES-DTW. In all cases, the same QbE-STD system was used, with the same parameters. As expected, the inclusion of the S matrix to store the origin of each possible path does not affect the MTWV score. Surprisingly, although on paper the use of matrix $S(n, m)$ should bring about big processing performances, the actual real-time speedup increase is not very significant. Our hypothesis is that the time spent in memory management is quite big, therefore eating up the improvements brought by not performing a trackback.

Results for the local and global normalizations are also shown, both for the non-efficient and for the memory efficient S-DTW algorithms. Results for the development and evaluation sets are different. While in the development set performing a global normalization achieved best results, it is with the local normalization that the best MTWV values are found for the evaluation set. In average, the local normalization still obtains best results between both sets. In terms of computation, we observe how by introducing an extra normalization matrix C in ”S-DTW+local norm” we obtain slower results than in ”S-DTW+origin matrix”. This could be explained, as before, by the fact that it takes extra computation to allocate and free the necessary memory. All these problems can get resolved by using the MES-DTW algorithm, with which the necessary allocated memory is minimal. For this reason all of the results for MES-DTW have much better real-time speedup factors than their counterparts. Overall, the MES-DTW system with global normalization achieves the best real-time speedup factor of all possibilities, reaching a 30% improvement compared to the S-DTW in the development set and a 25% improvement in the evaluation set.

Finally, we compare our results to those reported in [9] on the same database. The system proposed in [9, 8] follows a pattern-matching approach while using some image processing and information retrieval concepts for fast retrieval of results. Recently, in [18] very good performances have been shown on truly large scale ($>$ 400h) databases. For the database proposed here, their reported MTWV are slightly outperformed by our system’s results while their real-time speedup factor (normalized to 1s queries) is much slower. In their defense we should say that for the databases used in this paper ($\sim$ 4h) their information retrieval setup might not be able to take full advantage of their large-scale speedups.

5.2. Matching length analysis

In the present implementation of S-DTW or MES-DTW there is no explicit enforcement of any global constraints to the aligned paths. This means that no constraints are imposed to the alignment (other than an implicit predilection for diagonal alignments, as explained in Section 3.2) to avoid very short queries from matching very long reference subsequences, and vice versa.

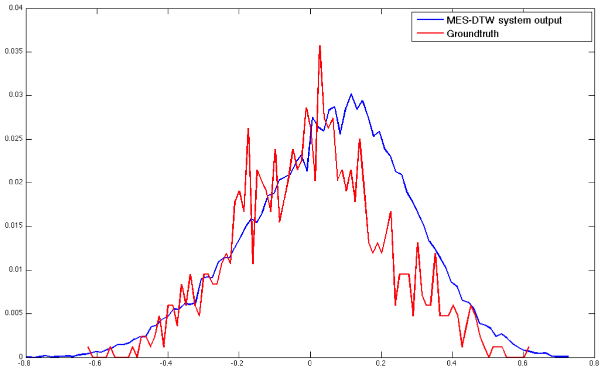

Fig. 2. *Histogram of matching length ratios*

To analyze how this might affect our matching, Figure 2 shows in blue, the normalized histogram of sequence length ratio between

[2] The value reported by the authors was adapted for a 1 second query.

all query sequences and matching reference subsequences returned by the system for the development set. Note that many of these sequences might not correspond to a true match, but do correspond to a match that the MES-DTW finds plausible (we repeated the test for a smaller subset of the top best matches and results were equivalent). The ratio between both sequence lengths has been computed as shown in Equation 8 which is close to 0 when both lengths are equal, positive when the query sequence is longer and negative when the reference subsequence is longer.

$$ratio_i = \frac{M - (b_i^* - a_i^*)}{M + (b_i^* - a_i^*)} \quad (7)$$

For comparison we also plot, in red, the same ratio computed over the ground truth queries and corresponding reference subsequences for the development set. We can see how, in both cases, there is an important portion of the matches in which one of the sequences is more than twice as long ($|ratio| > 0.33$) than the other. This might be due to some of the queries being formed by more than one word, with variable amounts of silence in between or by hesitations. Although a speech/non-speech detector is applied, we find it desirable to allow for any warping between both signals to ensure no matches are missed. Some further work will need to address how to eliminate the extra false alarms that such "free warping" generates. In addition, note how the MES-DTW histogram tends to tilt towards returning short reference subsequences. This might be due to the system's inability to correctly locate the start and end points of reference matching subsequences, mostly when the acoustic context surrounding the matching sequence is different between query and reference. This results in no silence being inserted at the beginning and end of the matching reference subsequence in the last step of the system, which causes a length difference w.r.t. the query.

6. CONCLUSIONS AND FUTURE WORK

In this paper we propose a novel pattern-matching based algorithm we call MES-DTW which derives from the subsequence-DTW proposed by [11], and initially applied to music matching. The MES-DTW algorithm improves the S-DTW algorithm in three ways. First, we eliminate the trackback across the accumulated cost matrix to find the reference subsequence starting point once the ending point has been found. Second, we experiment with two different score normalization approaches, one at local level and one at global level, that are able to boost matching performance, mostly when applying decisions to multiple query matches. Both these improvements include the need for extra support matrices that might become a memory burden in certain situations, that are solved by a memory efficient implementation that allows for the matching to be performed with very little memory allocations. We then propose a QbE-STD system that is used to compare the proposed MES-DTW with the S-DTW in a common framework. Results not only confirm that the system is more effective than S-DTW in finding matching subsequences but also much faster by reducing the amount of memory being allocated and freed. Future work includes improving the selection of start and end locations in the reference in order not to cut the beginning and end of found words, and testing the algorithm with very large databases.

7. REFERENCES

[1] Florian Metze, Nitendra Rajput, Xavier Anguera, Marelie Davel, Guillaume Gravier, Charl Van Heerden, Gautam V Mantena, Armando Muscariello, Kishore Prahallad, Igor Szöke, and Javier Tejedor, "The Spoken Web Search Task at MEDIAEVAL 2011," in *ICASSP*, 2012.

[2] Hiroaki Sakoe and Seibi Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, , no. 1, 1978.

[3] Alex S Park and James R Glass, "Unsupervised Pattern Discovery in Speech," *IEEE Transactions on audio, Speech and Language Processing*, vol. 16, no. 1, pp. 186–197, 2008.

[4] Yaodong Zhang and James R Glass, "Unsupervised Spoken Keyword Spotting via Segmental DTW on Gaussian Posteriorgrams," in *Proc. ASRU*, Merano, Italy, 2009, pp. 398–403.

[5] Xavier Anguera, "Speaker Independent Discriminant Feature Extraction for Acoustic Pattern-Matching," in *Proc. ICASSP*, 2012.

[6] Timothy J. Hazen, Wade Shen, Christopher M White, and A Phonetic Posteriorgram Representation, "Query-By-Example Spoken Term Detection Using Phonetic Posteriorgram Templates," in *ASRU*, 2009, pp. 421–426.

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

[8] Aren Jansen and Benjamin Van Durme, "Efficient Spoken Term Discovery Using Randomized Algorithms," in *ASRU*, 2011.

[9] Aren Jansen, Benjamin Van Durme, and Pascal Clark, "The JHU-HLTCOE Spoken Web Search System for MediaEval 2012," in *Proc. Mediaeval workshop*, 2012.

[10] Chun-an Chan and Lin-shan Lee, "Unsupervised Spoken-Term Detection with Spoken Queries Using Segment-based Dynamic Time Warping," in *Proc. Interspeech*, 2010.

[11] Meinard Müller, "Dynamic Time Warping, chapter 4," in *Information Retrieval for Music and Motion*, pp. 69–84. Springer-Verlag, Berlin, Germany, 2007.

[12] Cory Myers, Lawrence R Rabiner, and Aaron E Rosenberg, "Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition," *IEEE Transactions on Acoustics Speech and Signal PRocessing*, , no. 6, pp. 623–635, 1980.

[13] Armando Muscariello and Guillaume Gravier, "Variability tolerant audio motif discovery," in *Proc. International Conference on Multimedia Modeling*, 2009.

[14] Haipeng Wang and Tan Lee, "CUHK System for the Spoken Web Search task at Mediaeval 2012," in *in Proc. Mediaeval workshop*, 2012.

[15] Florian Metze, Etienne Barnard, Xavier Anguera, and Guillaume Gravier, "The Spoken Web Search Task," in *Proc. Mediaeval Workshop*, 2012.

[16] Etienne Barnard, Marelie Davel, and Charl Van Heerden, "ASR Corpus Design for Resource-Scarce Languages," in *Interspeech*, 2009, pp. 2847–2850.

[17] Jonathan G Fiscus, Jerome Ajot, John S Garofolo, and George Doddingtion, "Results of the 2006 Spoken Term Detection Evaluation," in *Interspeech*, 2007.

[18] Aren Jansen and Benjamin Van Durme, "Indexing Raw Acoustic Features for Scalable Zero Resource Search," in *Interspeech*, 2012.