PARTIAL SEQUENCE MATCHING USING AN UNBOUNDED DYNAMIC TIME WARPING ALGORITHM

Xavier Anguera[1], *Robert Macrae*[2] *and Nuria Oliver*[1]

[1]Telefonica Research
Multimedia research group, Via Augusta 177, Barcelona, Spain
[2]Centre for Digital Music, Queen Mary, University of London
Mile End Road, E1 4NS, London
{xanguera, nuriao}@tid.es, robert.macrae@elec.qmul.ac.uk

ABSTRACT

Before the advent of Hidden Markov Models(HMM)-based speech recognition, many speech applications were built using pattern matching algorithms like the Dynamic Time Warping (DTW) algorithm, which are generally robust to noise and easy to implement. The standard DTW algorithm usually suffers from lack of flexibility on start-end matching points and has high computational costs. Although some DTW-based algorithms have been proposed over the years to solve either one of these problems, none is able to discover multiple alignment paths with low computational costs. In this paper, we present an "unbounded" version on the DTW (U-DTW in short) that is computationally lightweight and allows for total flexibility on where the matching segment occurs. Results on a word matching database show very competitive performances both in accuracy and processing time compared to existing alternatives.

Index Terms— Dynamic time warping, partial sequence match, dynamic programming, similarity matrix, pattern matching

1. INTRODUCTION AND PREVIOUS WORK

Before the use of Hidden Markov Models (HMM) became ubiquitous in speech-based applications, pattern matching algorithms like the well known Dynamic Time Warping (DTW) algorithm [1] were used for applications such as keyword recognition [2]. The constant increase in computing power and the availability of larger labeled datasets pushed pattern matching techniques aside in favor of HMMs, as pattern matching algorithms did not scale well to having an increasing number of matching patterns. Still, HMMs have several well known weaknesses, such as overgeneralization and the need to have labeled training data, limiting their suitability for some speech applications. A few researchers have recently turned their eyes back to pattern matching algorithms for tasks such as template-based speech recognition [3], music synchronization [4] and unsupervised pattern discovery in speech [5] [6], among others.

DTW-based solutions have several drawbacks that have limited the scope of their applicability in real-life problems. The standard DTW algorithm assumes that: (1) the start and end points need to be known *a priori*; (2) the cost matrix between both sequences needs to be fully computed in order to find the optimum warping path; (3) the alignment assumes that both sequences are complete and structurally similar representations of each other.

However, the previous assumptions do not hold in many applications where the word(s) to be matched (in the case of speech) are within two long, and very different, contexts (sentences) or where structural variations such as omissions and insertions of large segments (in the case of music) may exist between two sequences. In [4, 7], Müller *et al.* propose an algorithm for music alignment that iteratively searches for matching subsequences between two music pieces. The Segmental-DTW algorithm is proposed in [4] as a slight modification of the standard DTW to allow searching for a keyword within a long sequence. Interestingly, Park *et al.* [5] give the same name to a very different algorithm that allows finding matching patterns within two long sequences. Similarly, [8] propose a similar approach to [5] for unsupervised word discovery. All these algorithms suffer from the need to compute all frame-pair similarities in the similarity matrix *before* searching for aligned sequences, which becomes intractable for large segments, with quadratic time costs. Therefore, the original segments need to be split into smaller pieces before running any of the previously mentioned algorithms. This limitation is solved by Dixon [9] who proposes an on-line DTW where the end point of the alignment maybe be unknown. However, it needs to know where the two matching sequences start. Very recently [6] proposed an offline DTW-based algorithm that finds repeated sequences with no start-end constraints but restricted to appear at most 1 minute apart and using a quite computational intensive method.

Alternatively, some mechanisms have been proposed in the literature in the text domain for finding text matching sequences when the sequences are composed of a set of discrete symbols. One such example is the Longest Common Subsequence (LCS) algorithm, using the Edit distance. Some works have applied these for music retrieval, such as in [10] where a warped version of LCS is proposed, or in [11] where sub-dimensional matching sequences are found in multi-dimensional data. However, all these solutions have to first convert the data into a set of discrete symbols, suffering from, sometimes strong, quantization errors.

In this paper, we propose a novel algorithm which we call Unbounded Dynamic Time Warping (U-DTW) which effectively solves the aforementioned limitations. First, possible alignment points (called *synchronization points*) are defined between both segments where time warped matches are searched for, eliminating the exhaustive computation of all the cost matrix at start. Second, a forward-backward dynamic programming algorithm is used to find exact start-end alignment points, unknown *a priori*. These together yield a matches detection accuracy increase of over 9% and a speed increase of up to 10 times over that of the segmental-DTW algorithm

During the development of this work X. Anguera was partially funded by the Torres Quevedo Spanish program

by Park et al. [5].

2. UNBOUNDED DYNAMIC TIME WARPING ALGORITHM

In this section we describe the Ubounded-DTW (U-DTW) algorithm. The algorithm is considered *unbounded* from three perspectives: (1) *start-end*, as U-DTW does not pose any restrictions on the start-end positions of the audio patterns to be matched; (2) *number of matches*, as more than one matching segment may be found with a single pass of the algorithm, returning their start-end locations and matching scores; and (3) *speed*, as it uses a search method that avoids unnecessary comparisons, speeding up the algorithm.

Note that two conditions are imposed on the signal in order to lower false alarm matches. First, a *minimum length* is defined. U-DTW only considers matching sequences that are longer than a minimum length L_{min} (typically set around 500ms such as in [5] and [6]). Second, a *maximum time warping* of $2X$ (and minimum of $\frac{1}{2}X$) is applied by defining proper local constraints, as explained in Section 2.2.

Given two acoustic sequences X_U and X_V their acoustic feature sequences are given by $U := (u_1, u_2, \ldots, u_M)$ and $V := (v_1, v_2, \ldots, v_N)$. We define the *similarity* $\mathcal{S}(m,n)$ between any two feature vectors, u_m and v_n, with $m\epsilon[1:M]$ and $n\epsilon[1:N]$, as their normalized inner product (or cosine of the angle θ between the two vectors): $\mathcal{S}(m,n) = \cos\theta = \frac{<u_m, v_n>}{\|u_m\|\|v_n\|}$.

Unlike other DTW-based algorithms, the similarities are computed only when needed, according to the forward-backward path finding algorithm, which brings significant computational savings. Additionally and in order to further speed up the processing time, two matrices are populated: (1) a global similarity matrix $D_g(m,n)$ that contains the optimum path accumulated similarity at each location (m,n); and (2) a matrix $M(m,n)$ that keeps the length of the optimal paths up to each location (m,n). All these matrices are empty at startup.

The pseudo-code of the U-DTW algorithm is:

Algorithm 1 Unbounded-DTW

```
Define appropriate synchronization points (see Section 2.1) in
  locations $(m,n)$. Compute the similarities for these points,
  populate $\mathcal{S}(m,n)$ and $D_g(m,n)$ with such similarities and set
  $M(m,n) = 1$.
for $m = 1 \ldots M$ do
  for $n = 1 \ldots N$ do
    if $M(m,n) \neq 0$ (a synch point or a possible path) then
      Apply forward path alg. (see Section 2.2).
    end if
  end for
end for
for all paths found in the forward pass do
  Apply backward path alg. (see Section 2.2).
  if Resulting Path length $> L_{min}$ then
    Register found path's start-end points and score
  end if
end for
```

The allowed frame jumps (*i.e.* local constraints) considered in the U-DTW algorithm, for both the forward (a) and backward (b) path algorithms, are shown in Figure 1. Note that these local constraints differ from the standard DTW in that the next points are considered at each step, and not the previous ones. Note also that neither strict insertion nor strict deletion steps are allowed (unlike the classical DTW algorithm), allowing at most $2X$ and $\frac{1}{2}X$ warpings of one signal to the other. This limitation is not unrealistically restrictive in the case of spontaneous speech and is very useful to avoid long consecutive insertions/deletions given that no global constraints are applied.

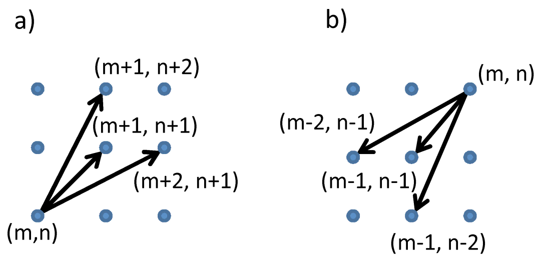

Fig. 1. Allowed frame jumps in U-DTW.

2.1. Synchronization Points

One of the key steps in the U-DTW algorithm is the proper selection of the *synchronization points* (SPs) which the forward path algorithm uses as starting points to look possible matching segments. Given that matching segments can occur anywhere within the two sequences compared, instead of considering all locations as possible start points we take advantage of the minimum length condition by looking at fewer locations (the SP) while ensuring that, if there is a matching segment, it will be found. Both the accuracy and the speed of the algorithm depend on the accurate selection of SP: sparse SPs increase the processing speed at the expense of possibly missing matching segments, whereas dense SPs are computationally more expensive to process.

We have experimented with two approaches to define the SPs by defining horizontal and diagonal synchronization bands.

Horizontal Bands: Given any frame-pair location (m,n) with $m\epsilon 1\ldots M$ and $n\epsilon 1 \ldots N$, we define SPs at positions $m = 1\ldots M$ for $n = \tau_h k$ with $k = 0 \ldots \frac{N}{\tau_h}$ and τ_h being the vertical separation between bands, a design parameter (see Figure 2a).

Diagonal Bands: Similarly, we define SPs at positions where $m + n = k\tau_d$ where $k = 0 \ldots \frac{max(M,N)}{\tau_d}$ for all possible values of (m,n) within the matrix (see Figure 2b).

Other interesting SP approaches such as a checkerboard with horizontal and vertical bands could be considered but will be left for future work.

The values of τ_h and τ_d determine the maximum lengths ($\Delta U_{h,d}^{max}$ and $\Delta V_{h,d}^{max}$) a matching segment can take in either sequence between two SP bands. Figure 2 shows in darker color all possible paths from one SP band to the other, for both considered methods. Taking segment U in the horizontal axis and V in the vertical and given the maximum angle $\frac{\pi}{4}$ the paths can deviate from the diagonal (due to the local constraints used) , for the horizontal bands $\Delta U_h^{max} = 2\tau_h$ and $\Delta V_h^{max} = \tau_h$. Similarly, for the diagonal bands $\Delta U_d^{max} = \Delta V_d^{max} = \frac{2}{\sqrt{(2)}}\tau_d$, where λ is the distance in diagonal between bands.

In order to minimize missed matching patterns, parameters τ_h and τ_d need to be defined according to the desired minimum matching segment length L_{min}. Similarly, they determine the minimum length of a matching segment after the forward path in order to be considered for the backward path.

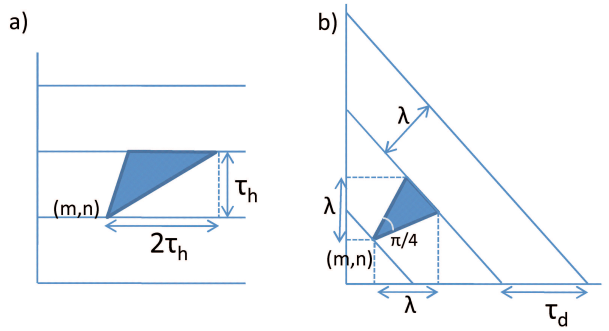

Fig. 2. Horizontal (a) and Diagonal (b) SP bands

2.2. Forward-Backward Paths Algorithm

For any considered frame-pair location (m, n), the forward and backward path algorithms check whether the current path can be extended to any of the surrounding frame-pair locations, conditioned by the local constrains seen in Figure 1.

Particularly, given $(m', n') = (m, n) + (i, j)$ where $(i, j)\epsilon\{(1, 1), (1, 2), (2, 1)\}$ for the forward path, and $(i, j)\epsilon\{(-1, -1), (-1, -2), (-2, -1)\}$ for the backward path, a new frame-pair position (m', n') is added to the currently considered path if the following conditions are met:

- The normalized global similarity score of the current path is greater than any previous paths (if any) at that location:

$$\frac{D_g(m, n) + \mathcal{S}(m', n')}{M(m, n) + 1} > \frac{G_g(m', n')}{M(m', n')} \quad (1)$$

- The normalized global similarity is greater than a predefined cutoff threshold.

$$\frac{D_g(m, n) + \mathcal{S}(m', n')}{M(m, n) + 1} > Thr \quad (2)$$

If successful, we set: $M(m', n') = M(m, n)+1$ and $D_g(m', n') = D_g(m, n) + \mathcal{S}(m', n')$. Note how Eq. 1 allows us to obtain the optimum DTW path without the need to backtrack, which is the key point to find optimum alignments while avoiding the precomputation of the entire similarity matrix $\mathcal{S}(m, n)$. Also note how at any given frame-pair location, the path can branch out in as many as 3 independent paths.

Any path is terminated at location (m, n) when none of the possible (m', n') meet the conditions above. In such case we proceed as follows: In the forward path step, for all paths sharing the same starting SP we keep only the longest one; in the backward path algorithm we return the total path (backwards + forward) and its average score if it exceeds the minimum length L_{min} in both dimensions.

Figure 3 shows an exemplary similarity matrix $\mathcal{S}(m, n)$ computed with two different sequences containing the word “Barcelona”. Light and dark grey areas indicate locations where the similarities have been computed for the forward and backward step algorithms respectively, in addition to the SP points. The chosen paths (of any length) are shown in black. Finally, white areas show where all computation has been skipped. The matching sequences are located between frames $(57, 38)$ and $(140, 130)$, in this case have been correctly identified by the algorithm.

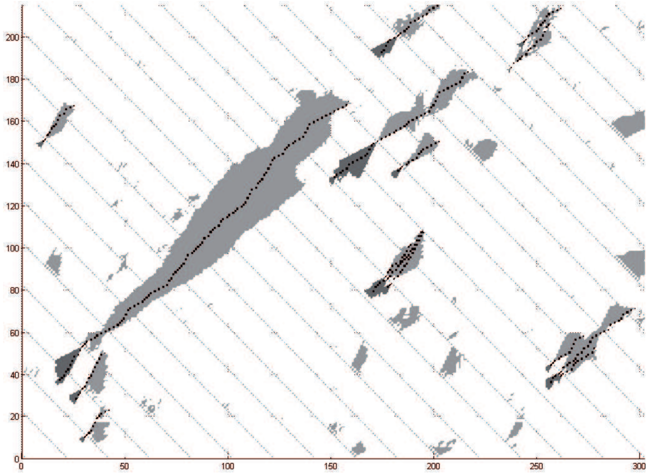

Fig. 3. Similarity matrix for “Barcelona-Barcelona” using U-DTW with diagonal SP bands

3. EXPERIMENTS

3.1. Database and Task

In order to test the proposed algorithm and to compare it with current state-of-the-art, we use a database recorded in-house by 23 people using an HTC-touch cell phone in a variety of office background conditions [12]. Each person recorded a total of 47 isolated words, each one repeated 5 non consecutive times, to a total of 235 recorded words per person. All files were stored at a sampling rate of 11.025Hz with 16bit/sample. Each file was parameterized with 10 MFCC every 10ms and cepstral Mean Substraction (CMS) was applied to the final features. A simple energy-based voice activity detector (VAD) was used to eliminate starting and ending non-speech regions. In order to add context to the words, two different pairs of start-end short sentences of 0.5s to 1.8s were recorded by a single speaker. Test acoustic sequences X_U and X_V were built by appending such segments to each recorded word: $X_U[i] = start_1 + word_i + end_1$ and $X_V[j] = start_2 + word_j + end_2$.

Tests were performed in the following way: for each acoustic sequence $X_U[i]$ of each speaker, the best matching segment score was found with each of the acoustic sequences $X_V[j]$ by the same speaker given that $i \neq j$, totalling $1, 264, 770$ matching runs. Note that both $X_U[i]-X_V[j]$ and $X_U[j]-X_V[i]$ comparisons were computed in order to measure whether any asymmetry in the algorithm could affect the final results.

3.2. Metrics and results

The main metric used is the matching accuracy, computed in the following manner: given a comparison on two acoustic sequences $X_U[i]$ and $X_V[j]$, for all sequences i in $X_U[i]$ ($X_U[i]$ used as query) we compute the percentage of times that the best matching word in $X_V[j]$ corresponds to a different iteration of the same word. The same is done for each sequence j in $X_V[j]$ ($X_V[j]$ used as query) and the average across all words and speakers is computed. Other two metrics considered are the average computing time per word-pair (including parametrization) in milliseconds and the average ratio of computed frame-pair distances in the similarity matrix, measuring algorithmic efficiency.

Algorithm	accuracy	avg. time	ratio
Segmental DTW Eucl.	80.61%	82.7ms	1
Segmental DTW inner prod.	74.62%	86.7ms	1
U-DTW horiz. bands	89.53%	10.6ms	0.51
U-DTW diag. bands	89.34%	9.0ms	0.42
standard DTW	95.42%	0.6ms	-

Table 1. Comparison of algorithms for sequence matching within context

Table 1 compares the proposed U-DTW algorithm (using either SP selection method proposed) with the Segmental-DTW algorithm proposed by Park and Glass in [5]. U-DTW uses a minimum length $L_{min} = 400ms$ and a separation between bands $\tau_h = \frac{L_{min}}{3}$ and $\tau_d = \frac{L_{min}}{2}$, as described in Section 2.1. For the segmental DTW, we used a minimum length of 500ms, 70ms band size and band overlap of 50%. The original segmental-DTW algorithm uses Euclidean distance as metric between frames while as U-DTW uses the normalized inner product. We also computed segmental-DTW using $1-$inner product for comparison purposes.

Columns 2 to 4 of Table 1 indicate the accuracy, average computation time per comparison and the average ratio of computed frame-pairs, respectively. From the results we see that aside from obtaining 9% or more absolute improvement in accuracy using U-DTW versus Segmental-DTW, the computational time is almost 10 times faster. Both U-DTW SP selection methods obtain similar results. However, the diagonal band approach can use slightly larger τ_d separation between bands and so the ratio of computed distances and the computation time are slightly lower.

Finally, we found it useful to compare the proposed algorithm with a standard DTW implementation run only on the words without any context. Such result defines an upper bound on accuracy for the case where exact start-end points are known. By using U-DTW without such knowledge we *only* lose around 6% accuracy versus standard DTW.

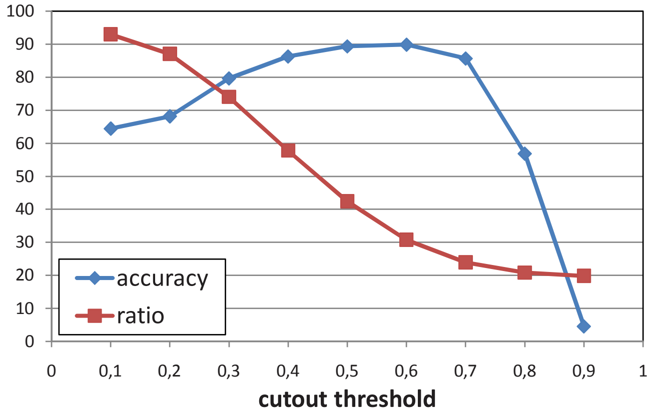

Fig. 4. Accuracy and computed frame ratio evolution as a function of the cutout threshold

One important parameter in U-DTW is the cutout threshold applied to prune non matching paths. If the threshold is set too high there will be many misses as correct paths will be cut before reaching minimum length, while if the threshold is too low there will be many false alarms, slowing down the system and leading to wrong final paths. Figure 4 shows accuracy and the computed frame-pairs ratio as a function of such threshold. Values are computed with a minimum length set to $L_{min} = 400$ms. Optimum accuracy is found at threshold=0.6 (Note that if two sequences were identical their normalized inner product would be 1).

4. CONCLUSIONS AND FUTURE WORK

The use of Dynamic Time Warping (DTW) for pattern discovery in speech has a wide range of interesting applications. However, the state-of-the-art algorithms in this area suffer from two important limitations: the difficulty in finding matching sequences that arbitrarily occur within other sequences and the high computational cost of computing the matches (exponential with the length of the sequences). In this paper we have proposed a novel algorithm, which we call Unbounded Dynamic Time Warping (U-DTW), which tackles the aforementioned problems. First, it is able to find acoustic matches between two segments wherever these occur in the segments, without any start-end restrictions. Second, it only computes similarities where needed, leading to significant computational savings, mainly when the length of sequences increases. In addition to describing the U-DTW algorithm, we have compared its performance with one state-of-the-art algorithm, Segmental DTW. In a database of 23 speakers each recoding 235 words we have obtained promising results: U-DTW was able to improve the accuracy by over 9% while running almost 10 times faster than a current state-of-the-art algorithm. Future work will focus on applying the U-DTW to finding recurring patterns in speech recordings towards multimedia indexing and classification.

5. REFERENCES

[1] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, pp. 43–49, 1978.

[2] Alan L. Higgins and Robert E. Wohlford, “Keyword recognition using template concatenation,” in *In Proc. ICASSP 1985*, 1985.

[3] Mathias De Wachter, Mike Matton, Kris Demuynck, Patrick Wambacq, Ronald Cools, and Dirk Van Compernolle, “Template-based continuous speech recognition,” *IEEE Transactions TASLP*, vol. 15, no. 4, pp. 1377–1390, May 2007.

[4] Meinard Muller, *Information Retrieval for music and Motion*, Springer, hardcover edition, 2007.

[5] Alex Park and James R. Glass, “Towards unsupervised pattern discovery in speech,” in *In Proc. ASRU05, Puerto Rico*, 2005.

[6] Armando Muscariello, Guillaume Gravier, and Frdric Bimbot, “Audio keyword extraction by unsipervised word discovery,” in *Proc. Interspeech*, 2009.

[7] Meinard Muller and Daniel Appelt, “Path-constrained partial music synchronization,” in *Proc. ICASSP*, 2008.

[8] Louis ten Bosch and Bert Cranen, “A computational model for unsupervised word discovery,” in *Interspeech*, 2007.

[9] Simon Dixon, “Live tracking of musical performances using on-line time warping,” in *Proc. Int. Conf. on Digital Audio Effects*, 2005.

[10] AnYuan Guo and Hava Siegelmann, “Time-warped longest common subsequence algorithm for music retrieval,” in *Proc. ISMIR*, 2004.

[11] David Minnen, Charles Isbell, Irfan Essa, and Thad Starner, “Detecting subdimensional motifs: An efficient algorithm for generalized multivariate pattern discovery,” in *IEEE Int. Conf. on Data Mining (ICDM)*, 2007.

[12] Xavier Anguera and Nuria Oliver, “MAMI: Multimodal annotations on a mobile phone,” in *Proc.. of Intl. Conf. on Mobile HCI (MobileHCI-08)*, 2008.