Information Retrieval-based Dynamic Time Warping

Xavier Anguera

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

Abstract

In this paper we introduce a novel dynamic programming algorithm called Information Retrieval-based Dynamic Time Warping (IR-DTW) used to find non-linearly matching subsequences between two time series where matching start and end points are not known a priori. In this paper our algorithm is applied for audio matching within the query by example (QbE) spoken term detection (STD) task, although it is applicable to many other problems. The main advantages of the proposed algorithm in comparison to similar approaches are twofold. On the one hand, IR-DTW requires a much smaller memory footprint than standard Dynamic Time Warping (DTW) approaches. On the other hand, it allows for the application of indexing techniques to the search collection for increased matching speed, which makes IR-DTW suitable for application in large scale implementations. We show through preliminary experimentation with a QbE-STD task that the memory footprint is greatly reduced in comparison to a baseline subsequence-DTW (S-DTW) implementation and that its matching accuracy is much better than that of pure diagonal matching and just slightly worse than that of S-DTW.

Index Terms: Information Retrieval, Dynamic Time Warping, Mediaeval, zero resources, dynamic programming

1. Introduction

In this paper we propose an algorithm we call IR-DTW to find non-linearly matching subsequences between two time series of n-dimensional real-valued data, and we test it in a QbE-STD speech search task. Although we apply the algorithm to a speech problem, the IR-DTW algorithm does not impose any speech specific constraint to the data, provided that every element in the time series can be represented as an n-dimensional vector and a metric of distance or similarity is defined between any two elements.

Most prior work in the area of pattern matching between real-valued vectors uses variations of the well known Dynamic Time Warping (DTW) algorithm [1] to find the optimal non-linear alignment between two time series. When dealing with speech data, audio recordings are usually represented as time sequences of MFCC or posteriorgram features [2]. Some recent DTW implementations include [3, 4, 5], which achieve considerable speed improvements in comparison to the standard DTW by imposing some upper bounds and global constraints to the matching sequences, thus allowing the algorithms to discard patterns without fully processing them. The main drawback of standard DTW implementations for the task at hand is that they require a prior knowledge of the start and end points in both time series.

Searching for unknown subsequences between two time series, where start and end points are unknown, poses a much bigger challenge and is a possible application scenario for the IR-DTW algorithm. Some prior work on this topic includes [6, 7, 8, 9], all derived from the DTW algorithm. These algorithms are generally computationally expensive and/or memory-hungry. Probably the most promising proposal is [7] which finds possible matching subsequences through a fast analysis of the similarity matrix of both time series by using a Hough transform image-based algorithm. Although the Hough transform only looks for linear alignments between both time series, it is sufficient to roughly find regions to which a fine-grained DTW is then applied. In [10] this algorithm was modified to use a pre-indexed search collection by using a Locality Sensitive Hashing (LSH) technique [11]. This allowed the authors to efficiently find matching subsequences within large-scale datasets.

Within the information retrieval community, [12, 13] propose an algorithm to find matching subsequences between two time series by using a vector of counts. This algorithm served as an inspiration for the IR-DTW and is thus briefly reviewed in section 2.1. While their implementation is highly memory-efficient, it does not allow for any time warping between both matching subsequences.

The algorithm discussed in this paper is a hybrid between that proposed in [13] and the DTW algorithm, by finding non-linearly matching subsequences while requiring very limited memory. IR-DTW performs subsequence matching in two steps. In a first step it identifies the set of possible matching frames from the search collection for every query frame. In this paper an exhaustive search is used, although we have shown in [14] that indexing techniques can be used to speedup the process. In a second step certain matching points (i.e. pairs of very similar frames in both time series) are connected together to form matching paths between both time series. This step modifies the algorithm in [13] to allow for time-warping between both time series by using a one-dimensional structure to register the growing matching paths. Steps one and two above are executed sequentially for each query frame. During the process, whenever a matching path is considered finished, it is erased from memory, thus dynamically reducing the total amount of memory required.

To show the suitability of the algorithm, preliminary tests have been conducted on a QbE-STD task, comparing results of IR-DTW with those of an optimized implementation of subsequence-DTW (S-DTW) [15] proposed by the authors in [16] and a modification of IR-DTW to allow only for diagonal matches. Results show that IR-DTW is more accurate that diagonal matching and less accurate than S-DTW (which is taken as an upper bound for the algorithm), while its memory requirements are one order of magnitude smaller than S-DTW.

2. Information Retrieval-based Dynamic Time Warping

Algorithm 1 shows the main steps of the proposed Information Retrieval-based DTW (IR-DTW) algorithm. The input to
The algorithm are two time series, composed of n-dimensional feature vectors, and a scalar parameter maxQDist. The output of the algorithm is a set of possible matching paths, where a matching path is defined by two matching subsequences, one in each time series. Within the QBE search task, we call the input time series Q (query) and R (search collection).

The IR-DTW algorithm performs a single forward pass, sequentially evaluating all the query frames $q_i$ in $Q$ in two steps. In the first step the function “best_points(•)” searches for the list of search collection frames $R' \subset R$ that are most similar to each query frame $q_i$, thus obtaining a set of matching frame pairs henceforth referred to as matching points. In this paper, search is performed exhaustively by computing the distance between the query frame $q_i$ and every frame in $R$ and selecting all frames in the search collection below a given threshold. In [14] we proposed the use of a k-means hierarchical-tree structure for enhanced speed and in order to be able to process large-scale databases.

Every identified matching point is uniquely identified by the time offsets of the matching query and the search collection databases.

The second step in Algorithm 1 (function “InputMatch(•)”) processes the set of matching points found in step 1 to identify non-linearly matching paths between both time series. Similarly to [12, 13] we use a one-dimensional structure $\Delta T$ of matching paths where new matching points are added to the most appropriate matching path, or form a new one. Next we briefly review the algorithm in [13] as it is the basis for this step.

### 2.1. Linear/Diagonal Subsequence Matching

In order to find possible diagonal alignments between matching points in two time series, a one-dimensional structure $\Delta T$ is created. Each position $k$ in $\Delta T$ contains information about matching paths, including the start-end offsets of the matching subsequences in both time series, the number of matching points included in the matching path and the overall aggregated score of all matching points assigned to that path.

For every matching point $(q_i, r_j) \in Q \times R$ found between each query frame and the search collection, a location in $\Delta T$ is defined as $k = tr_j - tq_i$. If $\Delta T[k]$ already contains a matching path for which the new matching point is a plausible continuation, it is appended to that path by modifying the path ending points and incrementing the number of matching points and the global score. Instead, if no existing matching path is available at location $k$, a new matching path is created with the information of that matching point.

Figure 1 shows a simple example of diagonal matching between a query and a search collection. On the left in Figure 1 we show the typical representation of matches between two time series. These form the axes represent the time steps in both time series, forming a matrix of all positions where matching points can be found. Any matching path between both time series is seen graphically as a subset of connected points in the matrix. On the right in Fig. 1 we plot the number of matching points for matching paths in every position $k$ in $\Delta T$. Any diagonal matching path of reasonable length will result in a maximum in the curve. Note that although the method is only able to detect diagonal matching paths, we can modify it to find any linear match by first applying a linear transformation to one of the time series and finding diagonal matches on the resulting series.

![Figure 1: Example of diagonal matching between time series](image)

By using the one-dimensional structure $\Delta T$ the algorithm is able to reduce the minimum required memory in comparison to DTW-based implementations, where a two-dimensional structure is required. The obvious drawback of this technique is that it only allows us to find linear matches between time series. In section 2.2 we describe how we modified this algorithm to take advantage of its compact matching strategy while combining it with the capacity of DTW algorithms to match time series with non-linear warping paths.

### 2.2. Non-linear Subsequence Matching Algorithm

**Algorithm 2 InputMatch:** Insertion of matches into $\Delta T$

**Input:** match point $m = \{tq_i, tr_j, d(q_i, r_j)\}$, maxQDist  

**Output:** $\Delta T$ with inserted match point

```plaintext
k ← tr_j − tq_i
best_path ← m
for k' = k - W Range to k + W Range do
    p ← $\Delta T[k']$
    if assert_relevance (maxQDist, m, p) == PASS then
        if assert_warp(m, p) == PASS then
            best_path = choose_best(best_path, (p ∪ m))
        end if
    end if
end for
\Delta T[k] ← best_path
```

Algorithm 2 describes how to register each matching point in $\Delta T$ while accounting for possible time warplings. Its input parameters are a pair of matching points and a system parameter maxQDist. Given that matching points form a sparse representation of the similarity between two time series, param-
eter maxQDist defines the maximum time difference allowed between two consecutive matching points (in either query or search collection) in order to consider that they belong to the same matching path.

As done previously, for every matching point a location in $\Delta T$ is computed as $k = tr_j - tq_i$. Now, instead of just considering the append of the matching point into $\Delta T[k]$, we allow for paths currently finishing at locations near $k$ to continue through the current matching point (i.e. to be registered in $\Delta T[k]$ replacing any previous matching paths). Note that this is equivalent to the selection of the best prior path in DTW. The range of positions around $k$ where we look for possible warping paths is defined as $k' = |k - W Range, k + W Range|$ where $W Range = \frac{\max QDist}{2}$ for the warping constraints applied in the current implementation, as further explained in Section 2.2.2 and illustrated in Figure 2.

For a matching path at position $\Delta T[k']$ to be considered the optimum prior for a given matching point it needs to fulfill three constraints: a) be a relevant path for the matching point (see Section 2.2.1); b) fall within the warping constraints considered (see Section 2.2.2); and c) be the best among all paths according to some selection criteria (see Section 2.2.3).

### 2.2.1. Path Relevance Constraint

The first constraint assesses the relevance of a matching path w.r.t. the current matching point. It is indicated by "assert_relevance()" in Algorithm 2 and described in Algorithm 3. We use this constraint to avoid big non-matching gaps between consecutive matching points. In this work we consider the maxQDist as the maximum elapsed time in either time series.

Moreover, given that the query is processed sequentially in time (i.e. $tq_i < tq_{i+1}$), paths that do not comply with this constraint are removed from $\Delta T$ (function "process&extract()"), as it is ensured that they will no longer comply with the constraint. The removed paths are then evaluated in terms of minimum length, number of matching points and score to determine if they can be considered a good match between both time series. In the current implementation for QbE we limit these paths to a minimum length of 1/4 of the query length (arbitrarily chosen for it not to be too short) and a minimum of 20 matching points to be considered a possible match between both time series. This procedure allows us to dynamically free some memory as the search progresses.

#### Algorithm 3 assert_relevance: asserts whether a given matching path is relevant for a given matching point

**Input:** match_point = $\{tq_i, tr_j, d(q_i, r_j)\}$, match_path, maxQDist

**Output:** bool={PASS, FAIL}

- $\Delta q \leftarrow |tq_i - match.path.tq_{end}|$
- $\Delta r \leftarrow |tr_j - match.path.tr_{end}|$

  if $\Delta q < \max QDist$ 
  & $\Delta r < \max QDist$ 
  then return(PASS)
  else if $\Delta q > \max QDist$ 
  then process&extract(match_path)
  end if

  return(FAIL)

#### 2.2.2. Time Warping Constraints

This constraint filters out those paths not abiding to the time warping constraints, similarly to what is done with local and global constraints in standard DTW algorithms. In addition to the monotonicity constraint (i.e. $q_{i+1} > q_i \forall i$ and $r_{j+1} > r_j \forall j$) used in DTW algorithms, in this paper we force the length difference between any two matching subsequences to be less than double. In speech, this constraint disallows having a speaking rate between two signals of more than the double. This is enforced both at local (for each new matching point) and global level (for the final matching path).

Given a matching path $m$ with its ending time offsets $(m.tq_{end}, m.tr_{end})$ and the time offsets of an input matching point $(tq_i, tr_j)$, the warping condition defined above can be expressed as presented in equation 1.

$$\frac{(tr_j - m.tr_{end})}{2} \leq (tq_i - m.tq_{end}) \leq 2(tr_j - m.tr_{end}) \quad (1)$$

![Warping constraints and matching range applied](image)

Figure 2 illustrates a matching point and the warping region where warping paths are looked for in a matrix representation. The acceptable warping region corresponds to the intersection of both the warping constraints ($q = 2r$ and $q = \frac{r}{2}$ lines) and the path relevance constraint (which indicates how distant from the new matching point a possible matching path can be). From Figure 2 one can derive the range $W Range$ as the maximum deviation from the diagonal in order to accept a path.

#### 2.2.3. Best Warping Path Selection

Among all matching paths abiding to the previous constraints only the best one is inserted in $\Delta T[k]$. Function “choose_best()” in Algorithm 2 selects the best matching path to be continued in the current matching point. In this work we select the matching path with the highest number of matching points assigned to it, in order to favor longer (and denser) paths versus shorter paths.

Finally, all remaining paths in $\Delta T$ are post-processed and considered as plausible final matches. Next section describes how the set of finally selected paths are further processed for the QbE-STD search task.

### 3. Global System for the QbE-STD Task

The QbE-STD system used to test the IR-DTW algorithm is similar to that proposed in [17, 16] where an S-DTW algo-
<table>
<thead>
<tr>
<th>System</th>
<th>Dev. set (mean/std)</th>
<th>Eval. set (mean/std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagonal</td>
<td>0.258 / 0.276</td>
<td>0.381 / 0.384</td>
</tr>
<tr>
<td>IR-DTW</td>
<td>0.394 / 0.394</td>
<td>0.394 / 0.394</td>
</tr>
<tr>
<td>S-DTW</td>
<td>0.442 / 0.450</td>
<td>0.442 / 0.442</td>
</tr>
<tr>
<td>RAILS [22]</td>
<td>0.381 / 0.384</td>
<td>0.381 / 0.384</td>
</tr>
</tbody>
</table>

Table 1: Comparison of systems results (MTWV)

<table>
<thead>
<tr>
<th>System</th>
<th>Dev. set (mean/std)</th>
<th>Eval. set (mean/std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-DTW</td>
<td>91.7Mb / 15.0Mb</td>
<td>112.3Mb / 21.8Mb</td>
</tr>
<tr>
<td>IR-DTW</td>
<td>506.2Mb / 342.8Mb</td>
<td>568.1Mb / 326.4Mb</td>
</tr>
</tbody>
</table>

Table 2: Comparison of memory usage

required to run the IR-DTW and the S-DTW algorithms on the development and evaluation datasets (excluding the memory required for storing the search collection in memory, which is equal in both cases). In each cell of Table 2 the first number presented is the mean of memory usage while running the algorithm, and the second number is the standard deviation. We see how IR-DTW requires significantly less memory in average than S-DTW and how this value does not vary much across queries.

4. Experimental Section

We tested the algorithm proposed in a QBE-STD task by using the Mediaeval 2012 SWS evaluation datasets and metrics [19]. The database consists of around 7.5 hours of telephone recordings in 4 African languages, which is a subset of the Lwazi database [20], and is split into a 3.6h development set and 3.8h evaluation set. A set of 100 development and 100 evaluation queries (in the form of audio snippets) are used to find where they appear in the data. The metric used to compare results is the MTWV (Minimum Term Weighted Value), as used in the NIST 2006 Spoken Term Detection evaluation [21]. Like in the Mediaeval 2012 evaluation, the scoring parameters used for the MTWV metric have been modified to better balance the impact of false alarms in the results.

4.1. Experiments and Results

The experiments performed compare the proposed IR-DTW algorithm with a linear-alignment version of IR-DTW where only a diagonal alignment is allowed, in order to simulate the linear/diagonal alignment introduced in [12, 13] and reviewed in section 2.1. Note that in this implementation the final S-DTW alignment step is still performed, thus making this implementation comparable to the RAILS system proposed in [7, 22], with which we also compare. In addition, we also show results for an implementation of S-DTW as described in [17].

Table 1 compares the MTWV for the development and evaluation sets. The proposed IR-DTW algorithm clearly outperforms the diagonal matching implementation. This proves the usefulness of including time warping constraints in the connection of matching points into matching paths. Direct comparison of IR-DTW with S-DTW is not fair as S-DTW takes a decision on matching paths by using all the information in the similarity matrix, thus setting an upper bound for the IR-DTW scores. Comparison with the RAILS system applied to this task [22] proves the competitiveness of the proposed solution. Note that this system is using a different set of acoustic features than ours.

In terms of memory usage, Table 2 compares the memory...
7. References


