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

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

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 $\mathcal{Q}$ in two steps. In the first step the function "best_points()" searches for the list of search collection frames $\mathcal{R}' \subset \mathcal{R}$ 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 $\mathcal{R}$ 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 frames, tq_i and tr_j, measured from the start of their respective temporal sequences, and their computed distance/similarity $d(q_i, r_j)$. In this work we use the negative logarithm of the normalized dot product as the distance between frames.

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 ΔT 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 ΔT is created. Each position k in ΔT 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 $(q_i, r_j)|q_i \epsilon \mathcal{Q}; r_j \epsilon \mathcal{R}$ found between each query frame and the search collection, a location in ΔT 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 ΔT. 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.

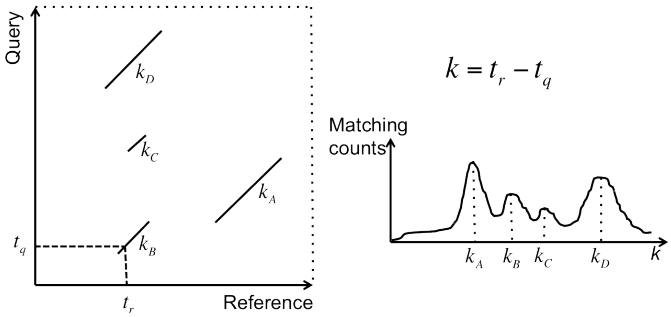

Figure 1: *Example of diagonal matching between time series*

By using the one-dimensional structure ΔT 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 ΔT

Input: match point $m = \{tq_i, tr_j, d(q_i, r_j)\}$, maxQDist
Output: ΔT with inserted match_point

```
k ← tr_j − tq_i
best_path ← m
for k' = k − WRange to k + WRange do
    p ← ΔT[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
ΔT[k] ← best_path
```

Algorithm 2 describes how to register each matching point in ΔT while accounting for possible time warpings. 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 ΔT 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 $k' = [k - WRange, k + WRange]$where $WRange = \frac{maxQDist}{2}$ 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. $tq_i < tq_{i+1} \forall i$), paths that do not comply with this constraint are removed from ΔT (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}

```
Δq ← |tq_i − match_path.tq_end|
Δr ← |tr_j − match_path.tr_end|
if Δq < maxQDist & Δr < maxQDist then
    return(PASS)
else if Δq > maxQDist 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.

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

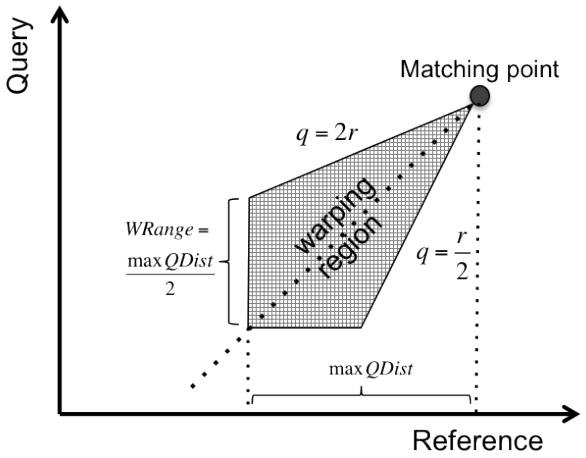

Figure 2: *Warping constraints and matching range applied*

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 $WRange$ 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 ΔT 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 1: Comparison of systems results (MTWV)

System	Dev. set	Eval. set
Diagonal	0.258	0.276
IR-DTW	0.394	0.394
S-DTW	0.442	0.450
RAILS [22]	0.381	0.384

rithm [15] is used instead. S-DTW is a variation of DTW where a given query is searched over the search collection data by considering fixed query start-end points and no constraint in start-end points for the search collection.

First, features are computed on all data by using standard MFCC-39 features and then, by using a background GMM model introduced in [18], 128-dimensional posterior probability vectors are obtained for each feature. Next, a speech/silence detector is used to eliminate low-energy frames, and finally the matching algorithm is used to find possible matching paths. From all returned paths, an overlap detection algorithm merges those paths with an overlap higher than 50% and output the 500 best (at most) non-overlapped matching paths.

Given that IR-DTW usually returns partially matching paths between both time series, an extra step is inserted for this particular task to obtain a more precise score and alignment. This is done so by performing an S-DTW search around the regions where the partial match has been found. This is very similar to what is done in [10].

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

Table 2: Comparison of memory usage

System	Dev. set (mean/std)	Eval. set (mean/std)
S-DTW	506.2Mb / 342.8Mb	568.1Mb / 326.4Mb
IR-DTW	91.7Mb / 15.0Mb	112.3Mb / 21.8Mb

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.

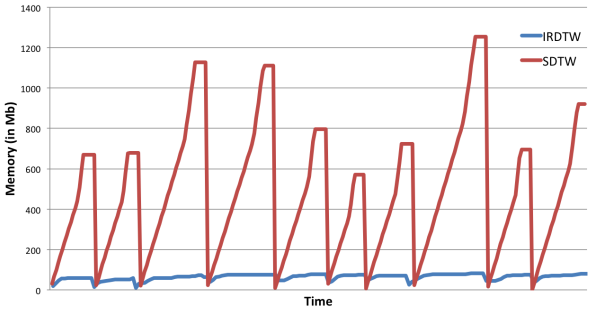

Figure 3: *Memory usage comparison for the first 10 queries in Mediaeval 2012 development data set*

Considering memory usage in more detail, Figure 3 shows the amount of memory allocated by the current C++ implementation of S-DTW and IR-DTW algorithms for the first 10 queries in Mediaeval 2012 development dataset (queries length range from 63 to 139 frames). While the S-DTW algorithm has a quadratic memory usage dependency with the query and search collection lengths, the IR-DTW algorithm has much lower needs, which we estimate are sub-linear with length. For the example in Figure 3 the average peak storage requirements for IR-DTW are 11.66 times smaller than for S-DTW. If we expand the search collection to contain all development and evaluation sets (over 7.5h of data) this ratio increases to 13.24, and expands further with more data.

5. Conclusions and Future Work

In this paper we present an Information Retrieval-based Dynamic Time Warping (IR-DTW) algorithm to find non-linearly matching subsequences between two time series. The algorithm improves upon algorithms used in Information Retrieval for diagonal sequences matching by adding the capability of matching time-warped signals. It also improves upon DTW implementations in terms of memory requirements and its capability of scaling to large amounts of data. We tested the algorithm in a QbE-STD task and compared it with an optimized implementation of a subsequence-DTW algorithm and a diagonal/linear matching implementation. Results show more than one order of magnitude memory savings when IR-DTW is compared to S-DTW and much better matching accuracy than diagonal matching algorithms. Future work includes optimizing the different parts of the algorithm to close the accuracy performance gap with the S-DTW algorithm.

6. Acknowledgements

We would like to thank Gautam Mantena, Jordi Luque and Aren Jansen for their useful insights on the algorithm.

7. References

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

[2] G. Aradilla, "Using Posterior-Based Features in Template Matching for Speech Recognition," in *ICSLP*, 2006.

[3] Y. Zhang, K. Adl, and J. Glass, "Fast Spoken Query Detection using Lower-Bound Dynamic Time Warping on Graphical Processing Units," in *ICASSP*, 2012, pp. 5173–5176.

[4] Y. Zhang and J. Glass, "A Piecewise Aggregate Approximation Lower-Bound Estimate for Posteriorgram-based Dynamic Time Warping," in *Interspeech*, 2011, pp. 1909–1912.

[5] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12*, p. 262, 2012.

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

[7] A. Jansen, K. Church, and H. Hermansky, "Towards Spoken Term Discovery at Scale with Zero Resources," in *Interspeech*, Makuhari, Japan, 2010, pp. 1676–1679.

[8] A. Muscariello, G. Gravier, and F. Bimbot, "Audio Keyword extraction by unsupervised word discovery," in *Proc. Interspeech*, 2009.

[9] X. Anguera, R. Macrae, and N. Oliver, "Partial Sequence Matching using an Unbounded Dynamic Time Warping Algorithm," *ICASSP*, 2010.

[10] A. Jansen and B. V. Durme, "Indexing Raw Acoustic Features for Scalable Zero Resource Search," in *Interspeech*, 2012.

[11] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," in *Proc. International Conference on Very Large Databases*, 1999, pp. 518–529.

[12] A. L.-c. Wang and K. H. Street, "An Industrial-Strength Audio Search Algorithm," in *Proc. ISMIR*, Baltimore, USA, 2003.

[13] A. Saracolu, E. Esen, T. K. Ate, B. O. Acar, U. Zubari, E. C. Ozan, and E. Özalp, "Content Based Copy Detection with Coarse Audio-Visual Fingerprints," in *Seventh International Workshop on Content-Based Multimedia Indexing*, 2009, pp. 213–218.

[14] G. Mantena and X. Anguera, "Speed Improvements to Information Retrieval-Based Dynamic Time Warping using Hierarchical k-Means Clustering," in *ICASSP*, 2013.

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

[16] X. Anguera and M. Ferrarons, "Memory Efficient Subsequence DTW for Query-by-Example Spoken Term Detection," in *International Conference on Multimedia and Expo*, 2013.

[17] X. Anguera, "Telefonica System for the Spoken Web Search Task at Mediaeval 2011," in *Mediaeval Workshop*, 2011.

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

[19] F. Metze, E. Barnard, X. Anguera, and G. Gravier, "The Spoken Web Search Task," in *Proc. Mediaeval Workshop*, 2012.

[20] E. Barnard, M. Davel, and C. V. Heerden, "ASR Corpus Design for Resource-Scarce Languages," in *Interspeech*, 2009, pp. 2847–2850.

[21] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddingtion, "Results of the 2006 Spoken Term Detection Evaluation," in *Interspeech*, 2007.

[22] A. Jansen, B. V. Durme, and P. Clark, "The JHU-HLTCOE Spoken Web Search System for MediaEval 2012," in *Proc. Mediaeval workshop*, 2012.