Model-Based Search in Large Time Series Databases

Alexios Kotsifakos[1], Vassilis Athitsos[2], Panagiotis Papapetrou[3,4], Jaakko Hollmén[3,4], and Dimitrios Gunopulos[1]

[1] *Department of Informatics and Communications, University of Athens, Greece*
[2] *Computer Science and Engineering Department, University of Texas at Arlington, USA*
[3] *Department of Information and Computer Science, Aalto University School of Science and Technology, Finland*
[4] *Helsinki Institute for Information Technology, Finland*

ABSTRACT

An important theoretical topic in assistive environments is reasoning about temporal patterns, that represent the sequential output of various sensors, and that can give us information about the health and activities of humans and the state of the environment. The recent growth in the quantity and quality of sensors for assistive environments has made it possible to create large databases of temporal patterns, that store sequences of observations obtained from such sensors over large time intervals. A topic of significant interest is being able to search such large databases so as to identify content of interest, for example activities of a certain type, or information about a patient's well-being. In this paper, we study two different approaches for conducting such searches: an exemplar-based approach, where we describe what we are looking for by giving an example, and a model-based approach, where we describe what we are looking for via a generative model. In particular, we describe the two different approaches, and we identify some important pros and cons for each approach. We also perform a comparative evaluation of exemplar-based search using dynamic time warping (DTW), and model-based search using Hidden Markov Models (HMMs), on large real datasets. In our experiments, when the number of training objects per model is sufficiently high, model-based search using HMMs produces more accurate search results than exemplar-based search using DTW.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining; H.2.4 [**Systems**]: Multimedia Databases

General Terms

Algorithms

Keywords

Dynamic Time Warping, time series, Hidden Markov Models

1. INTRODUCTION

In recent years, there has been a rapid increase in the quality and quantity of sensors that can be used in assistive environments. Such sensors include, for example, cameras, microphones, accelerometers, implantable sensors measuring vital statistics of a patient, RFID's, and devices measuring carbon dioxide, temperature, and humidity. Typically, each sensor produces measurements at regular or irregular time intervals. Time series are formed as sequences of such measurements. The amount of data that is produced by these sensors can be overwhelming: for example, a simple webcam operating at 30 frames per second, at 320x240 pixel resolution, generates about 600GB of data per day. Very large time series databases can result from storing sensor measurements from multiple sensors over large periods of time.

Such large databases of time series can serve as an important repository of knowledge, that can be used to understand what has happened in the past or to recognize what is happening in the present. For example, given a recently observed sequence of measurements, identifying the most similar sequences in the database can help identify the activity, state of health, or state of the environment that is associated with the recent measurements. Similarly, occurrences of interesting events in the past can be identified by searching the database for time series that match a specific example or model of such an event.

One family of approaches for searching time series databases are exemplar-based methods, where we describe what we are looking for using an example. In that case, the search process identifies time series stored in the database that are the most similar to the given example. The accuracy of the search results depends critically on the choice of similarity measure, as the similarity measure is the module that decides, for every time series in the database, whether that time series is a good match for the query (and thus should be retrieved) or not.

Several algorithms based on dynamic programming have been proposed for measuring similarity between time series. The most common one is Dynamic Time Warping (DTW) [8] and variants (e.g., constrained DTW [14], EDR [3], ERP [4]). These algorithms have the attractive property that they are robust to misalignments along the temporal axis, i.e., to differences in the speed in which observations evolve across time. Such methods have also been shown to achieve high accuracy in applications such as time series mining and classification [4, 6].

Another approach for identifying time series of interest is by describing what we are looking for using a model. *Hidden Markov Models (HMMs)* [1, 13] are widely known and have been used in a variety of domains, such as biology, speech recognition, and music retrieval among others [11, 12, 15]. Given a set of training sequences, an HMM models the underlying structure of these sequences by identifying the relationships between their observations. Conditional Random Fields (CRFs) [9, 16] are an alternative approach for modeling temporal patterns. CRFs focus on describing the differences between different types of patterns, so as to facilitate discrimination among different classes.

In this paper, we compare the accuracy of exemplar-based search methods and model-based search methods. We use Dynamic Time Warping (DTW) and constrained DTW (cDTW) as representative similarity measures for exemplar-based search. We use HMMs as a representative type of model for model-based search. For our experimental evaluation, we use the 20 publicly available datasets from the UCR time series archive [7].

The experimental results show that, on the datasets we have used, there are cases where exemplar-based search is advantageous, and other cases where model-based search is advantageous. In particular, in our results, exemplar-based search has produced better accuracy in cases where we did not have a sufficient number of training examples for the category of interest. On the other hand, when a sufficient number of training examples per class is available, model-based search has produced better accuracy in identifying time series of that class. These findings motivate the need for additional research, to more precisely characterize the advantages and disadvantages of these two alternatives, and they also motivate the need for practitioners to be aware of both alternatives, as opposed to automatically opting for HMMs when the goal is classification and opting for exemplar-based search when the goal is retrieval of multiple time series of the same type.

2. MEASURING SIMILARITY USING DTW

In this section we briefly describe Dynamic Time Warping (DTW), and a popular variant called constrained Dynamic Time Warping (cDTW). These methods are described in more detail in [2, 6, 8].

2.1 Dynamic Time Warping

The main computation of Dynamic Time Warping (DTW) is to identify an optimal alignment between two time series, and compute the matching cost corresponding to that alignment.

Given two N-dimensional time series $Q = (Q_1, \ldots, Q_{|Q|})$ and $X = (X_1, \ldots, X_{|X|})$, the DTW matching cost $D(Q, X)$ is defined recursively using a dynamic programming matrix D of size $(|Q| + 1)$x$(|X| + 1)$. A *null* element is added at the beginning of Q and X and has the property that it matches with another null element with a score of 0 and any other element with a score of ∞. Let D_{ij} denote the element at the i_{th} row and j_{th} column of D. Then, the DTW cost $D(Q, X)$ is defined as follows:

$$D_{0,0}(Q, X) = 0 \; . \tag{1}$$

$$D_{0,j}(Q, X) = \infty \; . \tag{2}$$

$$D_{i,0}(Q, X) = \infty \; . \tag{3}$$

$$D_{i,j}(Q, X) = L_p(Q_i, X_j) + \min \begin{cases} D_{i,j-1}(Q, X) \\ D_{i-1,j}(Q, X) \\ D_{i-1,j-1}(Q, X) \end{cases} \tag{4}$$

$$\forall (i = 1, \ldots, |Q|; j = 1, \ldots, |X|) \; .$$

$$D(Q, X) = D_{|Q|,|X|}(Q, X) \; . \tag{5}$$

Notice that $L_p(Q_i, X_j)$ is the L_p norm of Q_i and X_j.

2.2 Constrained Dynamic Time Warping

To speedup the computation of DTW, an additional constraint is added to the matrix computation that narrows down the number of positions in one sequence that can be aligned with a specific position in the other sequence. This method is called constrained DTW (cDTW). We consider only using the Sakoe-Chiba band [14] constraint where the lengths of the two time series sequences are the same. Given a warping width w, this constraint is defined as follows:

$$D_{i,j}(Q, X) = \infty \text{ if } |i - j| > R \; . \tag{6}$$

Notice that if $R = 0$, cDTW becomes the L_p distance. While a simple modification of DTW, cDTW has been shown to be more efficient than DTW for full sequence matching, and to also produce more meaningful matching scores in many datasets [6].

3. MODELING TEMPORAL PATTERNS USING HMMS

In this section we give a brief overview of the Hidden Markov Models (HMMs) and then describe the method that we followed in this paper to create the HMMs.

3.1 Hidden Markov Models

An HMM [10, 13] is a doubly stochastic process containing a finite set of states. In general, when being in a state a symbol/value can be emitted/observed according to a probability distribution, and *transition probabilities* (as their name implies) define the probabilities of moving from one state to another state (it may be the same). Using a more formal mathematical notation, an HMM is defined by:

- M distinct states,
- L values/symbols that can be observed at each state,
- set $T = \{t_{ij}\}$ of transition probabilities, where $t_{ij} = P[s_t = j|s_{t-1} = i], 1 \leq i, j \leq M$, where s_t is the state at time t. This means that the first order Markov chain is adopted, according to which the current state depends only on the predecessor one,
- set $E = \{e_j(k)\}$ of the probabilities of observation values/symbols at state j, where $e_j(k) = P[o_t = k|s_t = j]$, where o_t is the observed value/symbol at time t, and
- set $\Pi = \{\pi_j\}$ of prior probabilities, where $\pi_j = P[s_1 = j], 1 \leq j \leq M$.

When a database consists of sets of homogeneous sequences then HMMs can be highly applicable for retrieval or classification [12]. This lies on the fact that the nature of a trained HMM on homogeneous sequences is to reflect the probabilistic relations of the values within the sequences; training an HMM over sequences presenting heterogeneity cannot provide any implication about any common structure of the sequences. Consequently, having a database consisting of several classes (sets of homogeneous sequences) we can represent each class with an HMM, and then given a query Q look for the model that maximizes the probability of having generated Q. With this direction in mind, the time series matching problem is transformed to probabilistic-based matching.

3.2 Training HMMs

In order to create the HMM models, we broke the training phase into two steps: a) initialization, and b) iterative refinement. Assume that we have a dataset U, which is constituted by C classes, and the size of each class C_i, i.e., the number of time series in the class is denoted as $|C_i|$. Next, we present the two steps that we followed to create an HMM that represents an arbitrary C_i of U, where $|C_i| = n$.

3.2.1 Initialization Step

Initially, for each time series of the class C_i we compute the average distance of all the other time series in the class to this one. The distance measure used for this purpose is the DTW, and we have to mention that during the computation we keep track of the *warping path* of the time series alignment. After such computations we are able to find the *medoid* of C_i, i.e., the time series with the minimum average distance among all time series. Then, we break the medoid to M equal *segments/states* (the last segment may have less elements than the others) and for each of the rest time series of the class we find and store the elements that fall within each segment of the medoid based on the corresponding warping path. For each segment we also store the elements of the medoid that belong to that segment.

Assuming that the probability distribution E of each state is Gaussian, which is a common case in HMMs, it is described by the mean and standard deviation of the elements that comprise it (and have been previously computed and stored). However, apart from the probability distribution based on which an observation is emitted, to define an HMM, we also have to compute the transition matrix T. Considering each segment as an individual state, at each point in time t when an observation is emitted, we can either stay at the same state or move forward to the next segment/state. Thus, if the total number of elements belonging to a state s_t at time t is $|s_t|$, the probability of jumping to the next state is $p = n/|s_t|$, since, according to our methodology, if the number of elements in state s_t for an individual time series X is y, only the last element of the state is intended to lead us to s_{t+1} giving a transition probability $1/y$. Consequently, the probability of staying at the same state is $(1 - p) = (|s_t| - n)/|s_t|$, and when reaching the last state the probability of staying at that state is 1, since there is no other subsequent state.

3.2.2 Iterative Refinement Step

In this step we refine the HMM models created for all classes of the dataset U during the initialization step. First, we initialize a state sequence matrix where we are going to keep for all time series the states by which they pass through. For each time series per class we compute the Viterbi algorithm to find the single best state sequence for this time series [5, 13] as follows.

In the inizialization phase of the Viterbi algorithm we compute the log-likelihood of the first observation of the time series based on the Gaussian distribution of the first state (each time series has to start from state 1), and the rest elements of the first column are set to $-\infty$. In the recursion phase, due to the fact that we know that to get an observation we have either stayed in the same state or have done a transition from the previous state, we take into account only the values of the NxT probability matrix δ, where N is the number of states and T the number of observations, that represent the probabilities of the previous element for the previous and the current state. For example, when we are about to fill in cell (i, j), these values are $\delta(i, j-1)$ and $\delta(i-1, j-1)$. The transition probability values used are the ones computed at the initialization phase of creating the models, and after finding the most probable transition by computing $m = max(\delta(i, j-1) + \log(t_{ii}), \delta(i-1, j-1) + \log(t_{i-1i}))$, we fill in cell (i, j) the value $m + \log(e_i(k))$, where $e_i(k)$ is the probability of emitting value k based on the Gaussian distribution of state i. In the termination phase, we backtrack from the last cell of the probability matrix δ and for each state we store the values of the time series that fell within this state.

At the end of the refinement step the mean, standard deviation, and the transition matrices are updated appropriately, as done in the initialization phase of training the models. We have to mention that the procedure followed in this step can be repeated several times until a satisfiable classification accuracy attained on the training set is achieved.

3.3 Using HMMs

After an HMM has been trained on training examples from a certain class C_i, the HMM defines a probability distribution, that describes the probability of observing any time series, under the assumption that the observation belongs to class C_i. Given a time series X, the probability of X given that X belongs to class C_i (as modeled by our HMM) is computed using the Forward algorithm, which is based on dynamic programming. A full description of the algorithm can be found in [13].

4. EXEMPLAR-BASED AND MODEL-BASED SEARCH

In the previous sections we have provided brief descriptions of the DTW similarity measure and HMM models. In this section we describe how to use either DTW or HMMs to search time series databases, so as to identify content of interest.

Suppose, for example, that we want to search the database for time series corresponding to a specific activity, such as "person falling down" (an example that has clear relevance to assistive environments). In that case, the search process must decide, for each time series, whether that time series is a good match for this particular search (of instances of "person falling down") or not. This decision process can be decomposed into three steps:

1. Assign a score to each time series in the database. This score indicates how good a match each time series is for the particular search we are conducting. Whether high scores or low scores indicate better matches is up to convention.

2. Rank database time series according to the score they have received.

3. Return to the user the top K matches, where K is a user-specified parameter.

From the above three steps, the most critical one is the first step, where a score is assigned to each database time series. The system designer is free to choose any mathematically valid scoring function here, and that choice will determine the accuracy of the results. Naturally, we want to use a scoring function that assigns the best scores to the time series that the human user would consider to be "correct matches" for the query.

In exemplar-based search, we specify what we are looking for (e.g., time series corresponding to the "falling down" activity) by submitting, as a query, an example time series Q (in our example, Q would be an example time series labeled as "falling down"). Then, the score that is assigned to each database time series is simply the similarity score that is computed using our similarity measure of choice (which is DTW or cDTW in our experiments). In model-based search, we specify what we are looking for by submitting, as a query, a model of the activity. In our example search for "falling down", the query can be an HMM model trained on examples of the "falling down" activity. Then, the score that is assigned to each database time series is simply the output of the model on that time series. If the model is an HMM, the score is the result of the Forward algorithm [13] with inputs the HMM and the time series.

There is no theoretical guarantee that exemplar-based search will be better or worse than model-based search. Experimental evaluation is necessary in order to establish the performance of each approach in a specific domain. As shown in our experiments, there are specific cases where each approach does better than the alternative.

5. EXPERIMENTAL EVALUATION

5.1 Experimental Setup

5.1.1 Datasets

To compare DTW, cDTW and the HMMs we used the UCR data provided in [7]. In Table 3 we can see the min, max, $mean$, and $standard\ deviation$ of the number of examples in the training and test sets of all datasets, while for more information about the number of classes, size of training and test sets, and lengths of the time series per dataset please refer to the aforementioned web site. The time series of all datasets (both in their training and test sets) were resized to the maximum length of the datasets' time series, i.e., 637, using bilinear interpolation. This was done in order to treat all time series equally when building the models for the classes of different datasets, to avoid any bias against the probability when computing the Forward algorithm for a model and several time series of different lengths, and to perform a fair computation when using DTW and cDTW (which are certainly influenced by the length of the time series under comparison). Then all resized time series of the 20 test sets of the datasets were concatenated forming a test database consisting of 18602 time series.

5.1.2 Training the HMMs

We followed the method described in Section 3 and created an HMM for each class of the datasets, resulting in 178 models/classes. To improve the classification accuracy of the built models, for each dataset a large number of values for the number of states and the number of iterations in the refinement step was evaluated, and the optimal values were chosen based on classification accuracy attained on the training set. In Table 1 and Table 2 we can see the best achieved accuracy for each dataset, along with the corresponding number of states (in % of the length of the time series) and the number of iterations needed to achieve it.

5.1.3 Evaluation

The accuracy measures used to evaluate the exemplar-based methods with the model-based method are *precision* and *recall*. Precision is the percentage of correct answers we get out of the top-K results returned, while recall is the percentage of correct answers out of the total number of correct answers when the top-K results are returned. Thus, it is highly desirable to achieve high precision and recall for small K.

With regard to the exemplar-based methods, we used one time series per class as a query (for a total of 178 queries), and found the DTW and cDTW of all time series of the test database to this query. We note here that, depending on the dataset in which the query time series belongs to, the value of the R constraint used for cDTW was set to the value given for this dataset in [7]. Regarding the HMMs, we used each of the 178 trained models as a query and computed the probability of each time series of the test database being produced by that query model. For both exemplar-based methods and HMMs after computing the 18602 distances/probabilities, respectively, we sorted them in ascending/descending order and identified the ranks of the time series that belong to the same "correct" class with that represented by the query. This was done in order to be able to compute the precision and recall.

All experiments were performed on an AMD Opteron 8220 SE processor running as 2.8GHz and implemented in C++.

5.2 Experimental Results

In this section we present the results of our experimental evaluation and discuss on the behaviors observed for the three compared methods, DTW, cDTW and HMMs.

Since we are interested in precision and recall, in order to conduct fair comparisons between the methods, we present the results obtained for different values of recall. As mentioned in Section 5.1 we would like to have high values for precision and recall for small values of K. Ideally, when plotting the precision vs recall we should have a line parallel to the recall axis in the 100% precision, which means that independently of the requested recall to be achieved all returned results are correct. Thus, by setting recall to different values we can compare the methods by looking at the one with the higher precision for each individual recall value.

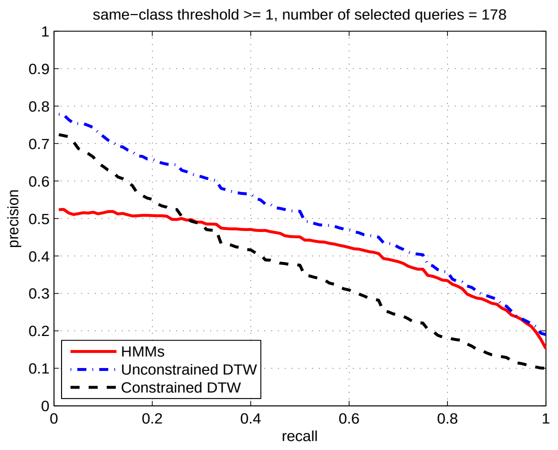

Figure 1: Precision vs Recall for model and exemplar-based queries, when the number of training examples for the models is $>=$ 1.

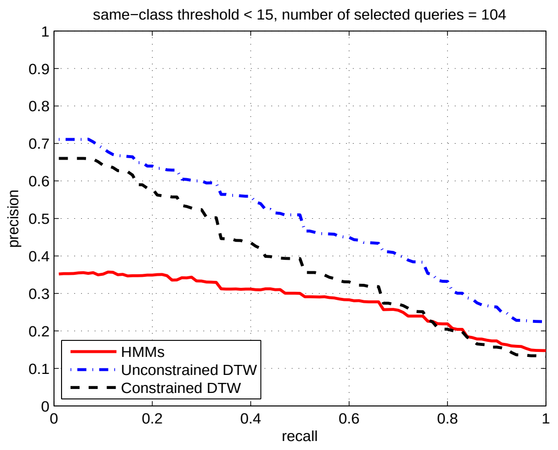

Figure 2: Precision vs Recall for model and exemplar-based queries, when the number of training examples for the models is $<$ 15.

In Tables 4, 5, 6, and 7, we can see the precision for all methods when the requested recall is 20, 40, 60, 80, and 100, for different thresholds on the number of training examples. For example, when the threshold is 10 this means that the models selected as queries had at least 10 examples in their training set, and in these tables the number of queries with such property is also presented; it is reminded that the total number of built models is 178. Referring

Table 1: Best achieved accuracy for the training phase of the first 10 datasets.

	Datasets									
	Synthetic Control	Gun-Point	CBF	Face (All)	OSU Leaf	Swedish Leaf	50 Words	Trace	Two Patterns	Wafer
iterations	2	2	3	3	12	12	9	1	7	6
states (%)	0.5	0.2	0.5	0.5	0.3	0.7	0.1	0.5	0.1	0.2
Accuracy (%)	99.67	90	100	99.11	81.5	84.2	88.44	83	99.8	95.9

Table 2: Best achieved accuracy for the training phase of the last 10 datasets.

	Datasets									
	Face (four)	Lightning-2	Lightning-7	ECG	Adiac	Yoga	Fish	Beef	Coffee	Olive Oil
iterations	1	12	4	4	12	3	11	4	1	7
states (%)	0.1	0.3	0.3	0.4	0.05	0.3	0.1	0.55	0.2	0.01
Accuracy (%)	100	90	94.29	87	65.13	75.33	87.43	43.33	75	70

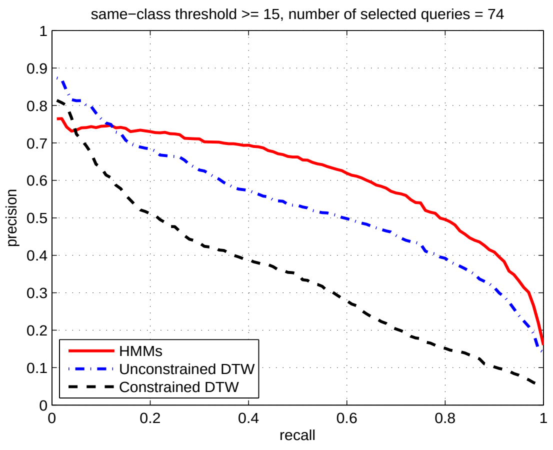

Figure 3: Precision vs Recall for model and exemplar-based queries, when the number of training examples for the models is $>= 15$.

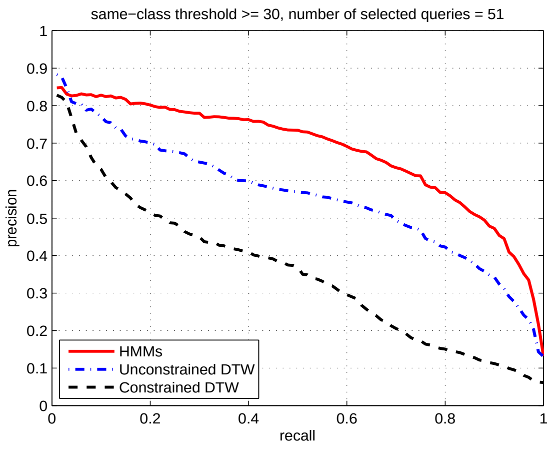

Figure 4: Precision vs Recall for model and exemplar-based queries, when the number of training examples for the models is $>= 30$.

to the HMMs, the values presented for precision are the averages we got for all models that satisfied the threshold condition, and for DTW and cDTW we have taken all time series in the train set of these models and also present the average values. In Figures 1, 3, 4, and 5 we have plotted the precision vs recall for all methods when the threshold is 1, 15, 30, and 60, respectively. In Figure 2 we show the precision vs recall when the threshold is at most 15.

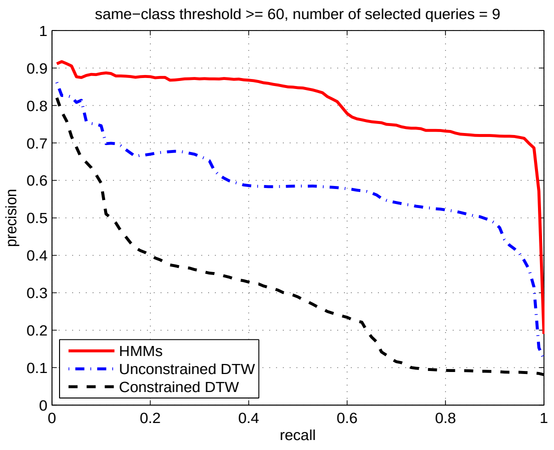

Figure 5: Precision vs Recall for model and exemplar-based queries, when the number of training examples for the models is $>= 60$.

Observing the tables and the figures comparing the precisions of the three methods, it can be clearly seen that the HMMs tend to be more accurate than the exemplar-based methods, for models where at least 15 training examples were used. One interesting exception is the low-recall setting, in which we are only interested in retrieving the top 2%-10% of the correct matches. We see that, for this setting, the advantages of HMMs over DTW become smaller, and in some cases DTW even outperfors HMMs (e.g., Figure 3). Moreover, it is of particular interest the fact that the more the trained examples used to create the HMMs, the higher the difference in precision is compared to DTW and cDTW, for all values of recall. This observation can be easily seen by looking at Figures 3 and 5.

It is also an interesting result that, overall, unconstrained DTW tends to give more accurate results than constrained DTW.

Table 3: Training and Test Examples vs statistic measures.

	Statistic Measures for Examples			
Examples	Min	Max	Mean	StdDev
Trained	1	903	30.32	77.037
Test	1	5499	104.506	462.744

6. CONCLUSIONS

In this paper we performed an evaluation of the accuracy of two approaches for searching time series databases. The first approach is search-by-example, and for that approach we used two

Table 4: Precision vs Recall for cDTW, DTW, and HMMs, for threshold = 10 (number of queries = 111).

	Precision				
Recall	20	40	60	80	100
cDTW	54.82	41.33	28.63	16.66	7.38
DTW	67.49	57.75	47.75	36.96	16.63
HMMs	64.10	58.88	52.17	41.17	15.97

Table 5: Precision vs Recall for cDTW, DTW, and HMMs, for threshold = 20 (number of queries = 69).

	Precision				
Recall	20	40	60	80	100
cDTW	50.19	39.09	27.61	14.17	5.49
DTW	69.67	58.24	50.50	39.53	14.49
HMMs	73.65	70.64	63.24	50.94	15.56

Table 6: Precision vs Recall for cDTW, DTW, and HMMs, for threshold = 40 (number of queries = 34).

	Precision				
Recall	20	40	60	80	100
cDTW	53.65	43.82	31.18	13.94	6.46
DTW	74.69	68.82	65.8	54.92	17.65
HMMs	84.75	84.71	79.8	69.44	17.16

Table 7: Precision vs Recall for cDTW, DTW, and HMMs, for threshold = 50 (number of queries = 17).

	Precision				
Recall	20	40	60	80	100
cDTW	53.71	42.48	30.88	18.87	11.52
DTW	74.67	70.50	68.94	61.73	23.58
HMMs	86.66	86.62	80.1	72.91	26.73

representative methods, that evaluated the matching score using unconstrained DTW or constrained DTW. The second approach is search-by-model, and the representative method that we used for that approach was search using HMM models. In the experimental evaluation we conducted on 20 datasets from the UCR time series archive, search-by-model yielded more competitivel results, with respect to search-by-example, for cases where the training examples per HMM were 15 or more, and also for recall values that were above 10%-20%. For search-by-example, its performance relative to search-by-model was more competitive for cases where there were few training examples per class, and also for low recall settings (where the user only wants to find a few matches, as opposed to a large number of correct matches). Moreover, the difference in accuracy when comparing HMMs with DTW and cDTW is increased in favor of HMMs when the number of training examples used to train them is getting larger. As future work, we are interested in finding ways to better characterize the settings where DTW or HMMs methods tend to be advantageous. Another interesting topic is finding ways to improve the efficiency of the search-by-model paradigm. While several highly efficient indexing methods exist for speeding up the search-by-example approach (e.g., [6]), the problem of speeding up search-by-model queries remains relatively unexplored.

Acknowledgments

This work was partially supported by the SensorGrid4Env and the MODAP European Commission projects. This work has also been partially supported by National Science Foundation grants IIS-0705749, IIS-0812601, CNS-0923494, and CNS-1035913, by a UTA startup grant to Professor Athitsos, and UTA STARS awards to Professors Chris Ding and Fillia Makedon. Panagiotis Papapetrou and Jaakko Hollmén were supported in part by the Algorithmic Centre of Excellence (ALGODAN).

7. REFERENCES

[1] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[2] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAI-94 workshop on knowledge discovery in databases*, pages 229–248, 1994.

[3] L. Chen and R. Ng. On the marriage of l_p-norms and edit distance. In *VLDB*, pages 792–803, 2004.

[4] L. Chen and M. T. Özsu. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502, 2005.

[5] G. Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[6] E. Keogh. Exact indexing of dynamic time warping. In *VLDB*, pages 406–417, 2002.

[7] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR time series classification/clustering homepage: http://www.cs.ucr.edu/~eamonn/time_series_data/, 2006.

[8] J. B. Kruskall and M. Liberman. The symmetric time warping algorithm: From continuous to discrete. In *Time Warps*. Addison-Wesley, 1983.

[9] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[10] S. Levinson, L. Rabiner, and M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *The Bell System Technical Journal*, 62(4):1035–1074, 1983.

[11] B. Pardo, J. Shifrin, and W. Birmingham. Name that tune: A pilot study in finding a melody from a sung query. *Journal of the American Society for Information Science and Technology*, 55(4):283–300, 2004.

[12] A. Pikrakis, S. Theodoridis, and D. Kamarotos. Classification of musical patterns using variable duration hidden Markov models. *Transactions on Audio, Speech, and Language Processing*, 14(5):1795–1807, 2006.

[13] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[14] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Transactions on ASSP*, 26:43–49, 1978.

[15] E. Unal, E. Chew, P. Georgiou, and S. Narayanan. Challenging uncertainty in query by humming systems: a fingerprinting approach. *Transactions on Audio Speech and Language Processing*, 16(2):359–371, 2008.

[16] S. B. Wang, A. Quattoni, L.-P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1521–1527, 2006.