CONTENT BASED AUDIO RETRIEVAL BASED ON HIDDEN MARKOV MODELS

SPEECH AND AUDIO PROCESSING AND RECOGNITION FINAL PROJECT

TECHNICAL REPORT
SPRING 2001

COLUMBIA UNIVERSITY

PROFESSOR: DAN ELLIS
STUDENT: MANUEL REYES

SPEECH AND AUDIO PROCESSING AND RECOGNITION
FINAL PROJECT
PROFESSOR: DAN ELLIS **STUDENT: MANUEL REYES**

CONTENT BASED AUDIO RETRIEVAL BASED ON HIDDEN MARKOV MODELS

This project consists in the implementation of a system that retrieves the five most similar audio files from an audio database when an audio file is presented as the input.
I concentrated on indoor and outdoor environmental audio files.

MOTIVATION.

Audio is a very important kind of media that includes speech, music and various kinds of environmental noise. With the recent public access to different audio databases, the management of audio databases has become a really interesting topic.

Very little work has been done in content based audio retrieval systems compared to the one done for content based image and video retrieval systems. The ideal content-based audio retrieval system should include all kind of audio files, speech, music and environmental sounds. Given the time for the realization of this project, I only concentrated on environmental audio files.

INTRODUCTION.

Hidden Markov Models (HMM) have been widely and successfully used to model speech for automatic speech recognition (ASR) systems.

The basic structure of a model HMM is depicted in Figure 1.

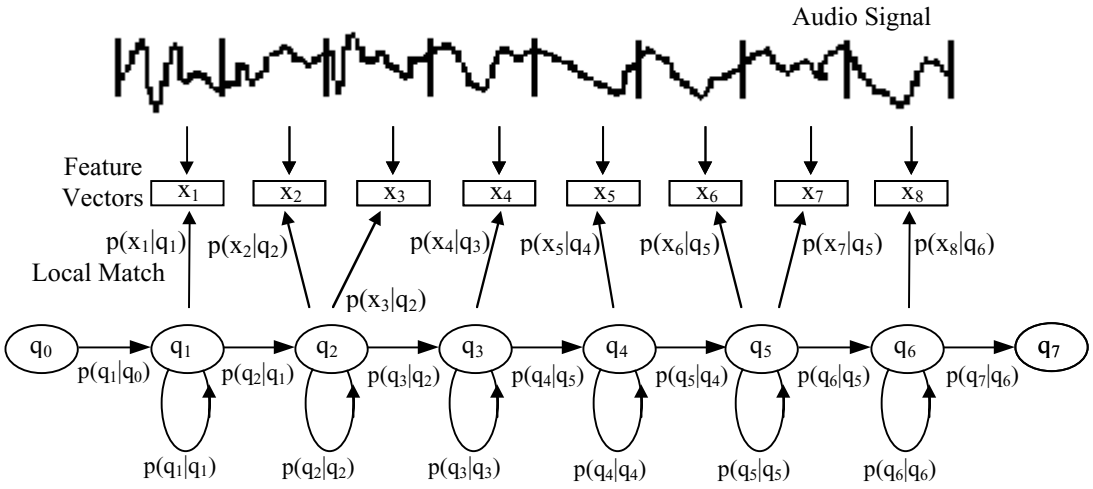

Figure 1

A Hidden Markov Model is a finite-state automaton with stochastic transitions in which the sequence of states is a Markov chain. Each state emits an output, which is non-deterministic, so it corresponds to a probabilistic density function. Then, for any observed output sequence, the generating state sequence of a HMM is hidden.

Each state (q_j) of the model is associated with an audio-symbol ; typically several states are used to model a single symbol. The outputs correspond to the observed feature vectors from the audio signal (Figure 1). Typically the initial and final states are non-emitting states.

The local match stage of the (ASR) system calculates the probabilistic density function (likelihoods) of the output given the state, $p(x_k|q_j)$. The global decoder finds the most likely allowable state sequence from the given models, choosing the model M, for which the joint probability of the sequence of states and the sequence of observations is the maximum. This is done using the Viterbi Algorithm [1].

In an ASR system the fundamental audio-symbols are the phones, so the states of the HMM are associated with a phone class. During the training phase the words and then the sequence of phones contained in a speech signal are known, so the association between the speech signal and the states can be estimated.

Figure 2, consists in the spectrogram of a speech signal, the divisions between the different phones can be observed.

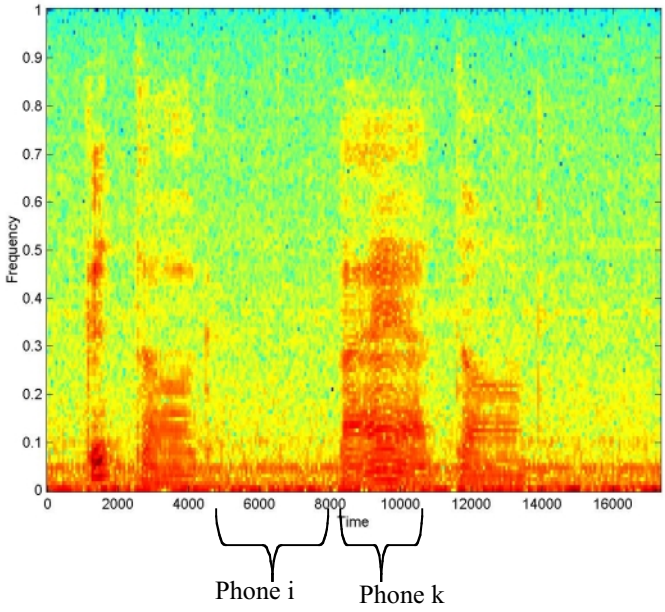

Figure 2

In environmental noise audio signal there are not clear audio-symbols and even if there were, they are not known in advanced.

MODELING ENVIROMENTAL NOISE SIGNALS USING HIDDEN MARKOV MODELS.

A HMM can be totally described by its states, its transition probabilities matrix and the emitting probabilities (likelihoods) that a state would produce a given observed feature vector. The modeling can be divided in four stages: a) Feature calculation b) Clustering Algorithm, c) Counting Transitions and d) Estimation of the likelihoods.

Feature Calculation

The single perceptual linear prediction coefficients and perceptual linear prediction coefficients with single deltas were calculated. Before applying the clustering algorithm, the feature vectors were normalized in order to prevent a bias towards the energy coefficient.
The system was tested using both feature vectors.

Clustering Algorithm

In this stage the states of the HMM are defined. The objective of this stage is to find clusters of audio samples that have common features. Once these clusters are found, a state is associated with each one of the clusters.

The clustering algorithm implemented is the following:

1.- Two thresholds are defined t_2 and t_1, where $t_2 > t_1$.
2.-The partition (of clusters) is initialized by taking the sample with the largest norm (Σx_i^2, where x_i are the coefficients of the X_i sample) as the center Z_1 of the first cluster.
3.- Compute the Euclidian distance d_{ij} between all the not classified samples X_i and the center Z_j of all the existing clusters. Find the minimum of such distances $\min\{d_{ij}\}$.

a) If $\min\{d_{ij}\} \leq t_1$ assign X_i to the jth cluster, and update the center of the cluster.
b) If $\min\{d_{ij}\} > t_2$ create a new cluster with X_i as the center.
c) If $t_1 < \min\{d_{ij}\} \leq t_2$, do not assign X_i to any cluster.

4.- Repeat step 3, until all the samples have been checked once. Calculate the variances of all clusters
5.- Use the summation of the variances of the clusters as a measure of the goodness of the partition. If the total variance is the same as last time. Go to step 6, otherwise go to step 3.
6.- If there are still unassigned samples, assign them to the closest cluster.
7.- If there are clusters with only one element, eliminate the cluster and assign the center to the closest valid cluster. A very similar version of this algorithm has been successfully used in a related work [2].

The number of states (since each cluster correspond to a state) obtained is related directly by the value of the thresholds t_1 and t_2.

If t_1 is large and t_1 and t_2 are closed in value, the number of states is large.
If t_1 is small and t_2 is large the number of states is small and the percentage of unassigned samples is quite large..

The choice of these thresholds is extremely important as would be shown in the results.

Choosing the thresholds

The idea is to find a partition that would represent the different groups enclosed in the signal. I got the spectrograms of all the signals in the database. At the beginning, I adjusted the thresholds in order to get a partition according with the different groups observed in the spectrograms.

In figure3, the spectrogram of a file containing people clapping is presented

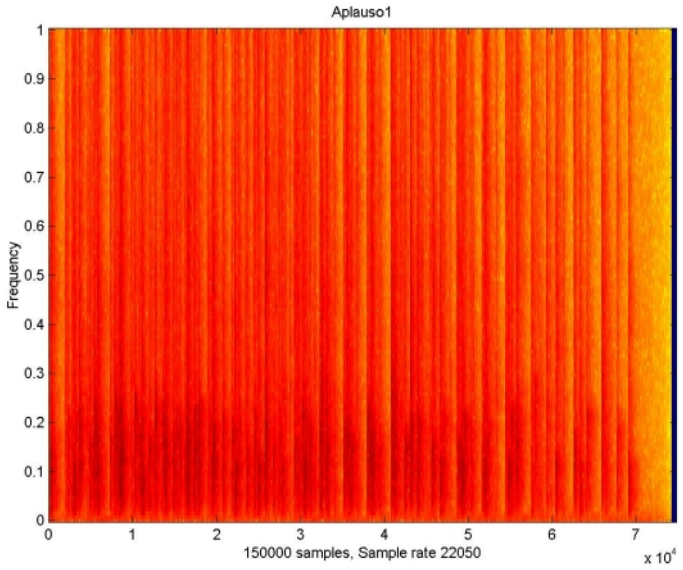

Figure 3

As can be observed some kind of a pattern could be found. So I adjusted the thresholds until I found a partition that would best resemble the groups observed in the spectrogram. For this case the best partition found have four clusters, one to represent the silence period at the end of the utterance and three for the rest of the signal.

Since I wanted to implement an automatic system to model the signal, I tried to find an automatic way to choose the thresholds.

In order to do so, I calculated the histogram of the distribution for the distances between samples. Finding the mean (μ) and the standard deviation (σ) of the distances, I found

that all the thresholds that I calculated manually were pretty similar to one of the following pairs of thresholds.

$(t1,t2) = (\mu-0.666\sigma, \mu+0.666\sigma)$; $(t1,t2) = (\mu-\sigma, \mu+\sigma)$ and $(t1,t2) = (\mu-1.333\sigma, \mu+1.333\sigma)$

I used the above pairs of thresholds to calculate the partition of all the files contained in the database. Including the ones I didn t calculate the partitions by manually choosing the thresholds.

As I will show in the results part of this report, that the system work pretty well to retrieve files containing indoor ambient noise, I didn t calculate manually the thresholds for these kind of signals, as I said the system work pretty well in these kind of signal, even though clear groups of samples can not be easily observed from the spectrograms. An example of such a spectrogram can be observed in figure 4.

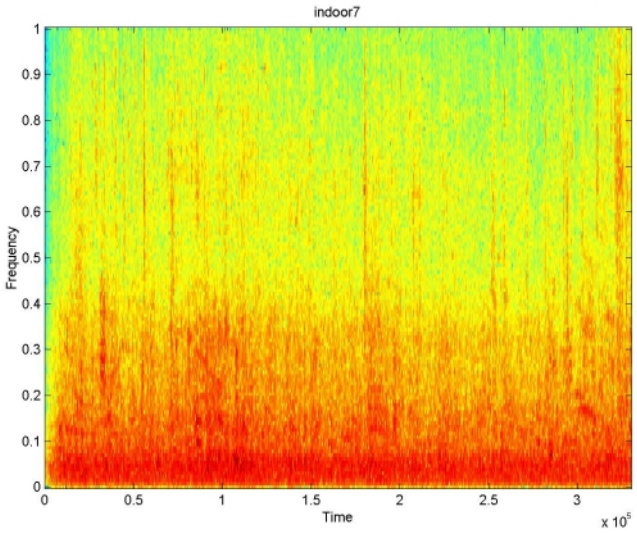

Figure 4. Spectrogram of ambient noise in a restaurant.

The main problem with this approach is that the estimation of the histogram require the calculation of $n(n-1)/2$ distances, where n is the number of samples. A possible solution for this problem could be to downsample the signal by a certain factor k, which would reduce the number of calculations by a factor of k^2, but this approach would change the distribution of the distances since spatial-adjacent samples would tend to be similar. I think that this variation could still be used taking in account that the number of small distances would be reduced.

Counting Transitions

The above clustering algorithm defines the states of the HMM, a classic output of the algorithm would be the following:

11111222222333322222333311111111111111111111111111
11333333333333333333333333331112222333332222333333 3
311111111144444 .

The numbers on the string correspond to the number of the different clusters found, the position of the string corresponds to the sample number. The number of transition from one state to another (including itself) is estimated by simple counting them. The transition probabilities matrix is estimated using the following formula:

$$Pij = \frac{\text{Number of transitions from i to j.}}{\text{Number of transitions from i.}}$$

Estimation of the likelihoods.

The emitting probabilities (likelihoods) of a feature vector given a given state is estimated by a single Gaussian distribution like the following:

$$p(X|q_j) = \eta(X;\mu_j,\Sigma_j) = (1/\text{sqrt}((2*\pi)^n|\Sigma_j|))\ e^{-0.5*(X-\mu j)\ \Sigma j\text{^-}1(X-\mu j)}$$

where μj is the center of the cluster and Σ is a diagonal matrix and the values of the diagonal are given by the variance of the cluster.

SYSTEM IMPLEMENTATION

The database consists in 40 audio files containing environmental sounds of indoor and outdoor ambient noises as: people clapping, people mumbling, people cheering, sounds from a restaurant, sounds from a courtroom, rain, footsteps, etc.
The HMM of these files were calculated using the three pairs of thresholds.

The test set consists on 34 audio files similar in content to the ones in the database. The system calculates and normalizes the feature vectors of these files. Then the system searches for the five most similar files on the database by calculating the five most likely models to produce the given input. This was done using the Viterbi Algorithm, implemented by the HTK framework. In order to do so the models found for the database were written using the HTK format.

The HTK framework [3] by Entropic Ltd. is a powerful software used primarily for speech recognition.

In the recognition phase the HTK framework applies the Viterbi Algorithm to the input using the stored Hidden Markov Models, once it identifies which are the most likely models it consults its dictionary to find the sequence of words associated with those models, then the system produces a file with the written transcription of those words.

For this task, the dictionary consists in a relation between the models stored and the files that produced those models. So the output of my system is a file containing the names of the most similar files on the database.

The following are two examples of such file:

```
.
"htk_dir/applause4.rec"
0 166700000 applause6 -27879.976562
///
0 166700000 applause9 -28299.996094
///
0 166700000 applause5 -28854.785156
///
0 166700000 applause3 -29165.718750
///
0 166700000 court4 -29501.517578
.

"htk_dir/step4.rec"
0 11200000 step1 -1891.097534
///
0 11200000 step3 -1936.566528
///
0 11200000 step9 -1993.590088
///
0 11200000 step7 -1995.009521
///
0 11200000 step11 -2013.184570
```

In the first example the files applause6, applause9, applause5, applause3 and court4 are the five most similar files (in that order) contained in the database to the input file applause4.

Similarly, in the second example the input was the file step4, while the files retrieved are step1, step3, step9, step7 and step11.

RESULTS

The system was tested using eight different conditions.

Two types of features were used: Single perceptual linear prediction coefficients and perceptual linear prediction coefficients with single deltas.

The system was tested using both types of features using four different sets of models. The sets of models used were the ones produced by the following thresholds:

a) $(t1,t2) = (\mu-0.666\sigma, \mu+0.666\sigma)$
b) $(t1,t2) = (\mu-\sigma, \mu+\sigma)$
c) $(t1,t2) = (\mu-1.333\sigma, \mu+1.333\sigma)$
d) All the models used in the first three sets.

Using single perceptual linear prediction coefficients

A really important point in analyzing the results is that the **values of the thresholds** were **found** after that some experiments were manually done **using single perceptual linear prediction coefficients**.

a) $(t1,t2) = (\mu-0.666\sigma, \mu+0.666\sigma)$

7 out of 8 times indoors ambient noise files were retrieved when a indoor file was used as the query input.
5 out of 5 times files containing people clapping were retrieved when a clapping file was used as the query input.
1 out of 2 times files containing people booing were retrieved when a booing file was used as the query input.
0 out of 4 times files containing people laughing were retrieved when a laughing file was used as the query input.
7 out of 7 times files containing footsteps were retrieved when a footsteps file was used as the query input.
6 out of 6 times files containing people cheering were retrieved when a cheering file was used as the query input.
1 out of 1 times files containing sounds of a crowd were retrieved when a crowd file was used as the query input.
1 out of 1 times files containing rain sounds were retrieved when a rain file was used as the query input.

Then **28 out of 34** times a correct file was retrieved.

As can be observed the main source of errors were the files containing people laughing.

b) (t1,t2) = (μ-σ, μ+σ)

8 out of 8 times indoors ambient noise files were retrieved when a indoor file was used as the query input.
5 out of 5 times files containing people clapping were retrieved when a clapping file was used as the query input.
1 out of 2 times files containing people booing were retrieved when a booing file was used as the query input.
0 out of 4 times files containing people laughing were retrieved when a laughing file was used as the query input.
7 out of 7 times files containing footsteps were retrieved when a footsteps file was used as the query input.
6 out of 6 times files containing people cheering were retrieved when a cheering file was used as the query input.
1 out of 1 times files containing sounds of a crowd were retrieved when a crowd file was used as the query input.
1 out of 1 times files containing rain sounds were retrieved when a rain file was used as the query input.

Then **29 out of 34** times a correct file was retrieved.

As can be observed the main source of errors again were the files containing people laughing.

c) (t1,t2) = (μ-1.3333σ, μ+1.3333σ)

7 out of 8 times indoors ambient noise files were retrieved when a indoor file was used as the query input.
5 out of 5 times files containing people clapping were retrieved when a clapping file was used as the query input.
1 out of 2 times files containing people booing were retrieved when a booing file was used as the query input.
1 out of 4 times files containing people laughing were retrieved when a laughing file was used as the query input.
7 out of 7 times files containing footsteps were retrieved when a footsteps file was used as the query input.
6 out of 6 times files containing people cheering were retrieved when a cheering file was used as the query input.
1 out of 1 times files containing sounds of a crowd were retrieved when a crowd file was used as the query input.
1 out of 1 times files containing rain sounds were retrieved when a rain file was used as the query input.

Then **29 out of 34** times a correct file was retrieved.

d) **Using all the models.**

8 out of 8 times indoors ambient noise files were retrieved when a indoor file was used as the query input.
5 out of 5 times files containing people clapping were retrieved when a clapping file was used as the query input.
1 out of 2 times files containing people booing were retrieved when a booing file was used as the query input.
1 out of 4 times files containing people laughing were retrieved when a laughing file was used as the query input.
7 out of 7 times files containing footsteps were retrieved when a footsteps file was used as the query input.
6 out of 6 times files containing people cheering were retrieved when a cheering file was used as the query input.
1 out of 1 times files containing sounds of a crowd were retrieved when a crowd file was used as the query input.
1 out of 1 times files containing rain sounds were retrieved when a rain file was used as the query input.

Then **30 out of 34** times a correct file was retrieved.

Perceptual linear prediction coefficients with single deltas.

a) **(t1,t2) = (μ-0.666σ, μ+0.666σ)**

5 out of 8 times indoors ambient noise files were retrieved when a indoor file was used as the query input.
2 out of 5 times files containing people clapping were retrieved when a clapping file was used as the query input.
0 out of 2 times files containing people booing were retrieved when a booing file was used as the query input.
0 out of 4 times files containing people laughing were retrieved when a laughing file was used as the query input.
5 out of 7 times files containing footsteps were retrieved when a footsteps file was used as the query input.
5 out of 6 times files containing people cheering were retrieved when a cheering file was used as the query input.
0 out of 1 times files containing sounds of a crowd were retrieved when a crowd file was used as the query input.
0 out of 1 times files containing rain sounds were retrieved when a rain file was used as the query input.

Then **17 out of 34** times a correct file was retrieved.

b) (t1,t2) = (μ-σ, μ+σ)

6 out of 8 times indoors ambient noise files were retrieved when a indoor file was used as the query input.
5 out of 5 times files containing people clapping were retrieved when a clapping file was used as the query input.
0 out of 2 times files containing people booing were retrieved when a booing file was used as the query input.
0 out of 4 times files containing people laughing were retrieved when a laughing file was used as the query input.
6 out of 7 times files containing footsteps were retrieved when a footsteps file was used as the query input.
5 out of 6 times files containing people cheering were retrieved when a cheering file was used as the query input.
0 out of 1 times files containing sounds of a crowd were retrieved when a crowd file was used as the query input.
0 out of 1 times files containing rain sounds were retrieved when a rain file was used as the query input.

Then **22 out of 34** times a correct file was retrieved.

As can be observed the main source of errors again were the files containing people laughing.

c) (t1,t2) = (μ-1.3333σ, μ+1.3333σ)

7 out of 8 times indoors ambient noise files were retrieved when a indoor file was used as the query input.
5 out of 5 times files containing people clapping were retrieved when a clapping file was used as the query input.
1 out of 2 times files containing people booing were retrieved when a booing file was used as the query input.
0 out of 4 times files containing people laughing were retrieved when a laughing file was used as the query input.
6 out of 7 times files containing footsteps were retrieved when a footsteps file was used as the query input.
5 out of 6 times files containing people cheering were retrieved when a cheering file was used as the query input.
0 out of 1 times files containing sounds of a crowd were retrieved when a crowd file was used as the query input.
1 out of 1 times files containing rain sounds were retrieved when a rain file was used as the query input.

Then **25 out of 34** times a correct file was retrieved.

d) **Using all the models.**

7 out of 8 times indoors ambient noise files were retrieved when a indoor file was used as the query input.
5 out of 5 times files containing people clapping were retrieved when a clapping file was used as the query input.
1 out of 2 times files containing people booing were retrieved when a booing file was used as the query input.
0 out of 4 times files containing people laughing were retrieved when a laughing file was used as the query input.
6 out of 7 times files containing footsteps were retrieved when a footsteps file was used as the query input.
5 out of 6 times files containing people cheering were retrieved when a cheering file was used as the query input.
0 out of 1 times files containing sounds of a crowd were retrieved when a crowd file was used as the query input.
1 out of 1 times files containing rain sounds were retrieved when a rain file was used as the query input.

Then **25 out of 34** times a correct file was retrieved.

FINAL COMMENTS

As can be observed in the results the system worked pretty well for almost all the files tested when the simple perceptual linear predictions were used.

The models obtained using these features have between 3 and 9 states. The models produced by the first pair of threshold produce in average models with more states than the other two pairs of thresholds.

Even though the laughing files seemed to have a pattern, as can be observed in figure 5. The system failed on modeling those kinds of files.

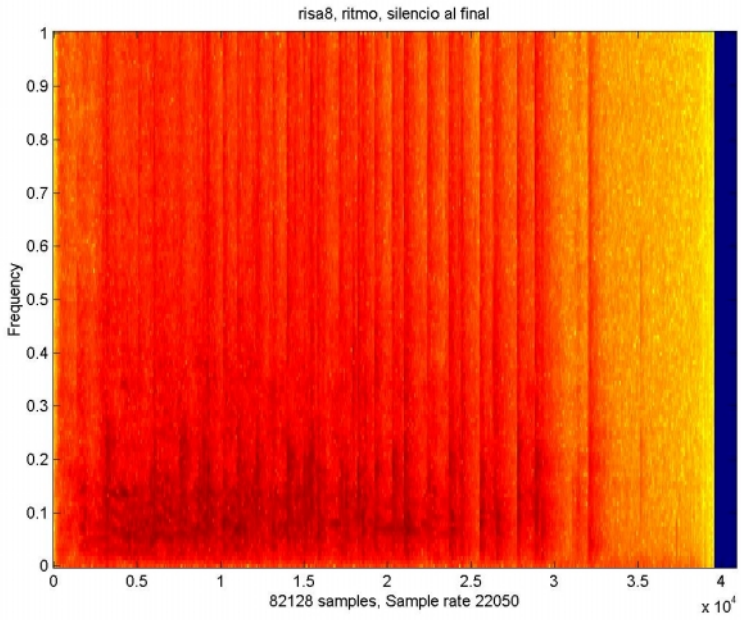

Figure 5

I think that maybe the problem was that the simple PLP coefficients were not capable to represent this kinds of files accurately and that maybe adding delta features may help, but as can be observed from the results that is not the case.

I think that the system perform worst using the PLP features plus deltas because the thresholds used were the optimal for the simple PLP coefficients, I think that the fact that the models obtained using these features have between 8 and 25 states have an impact in the performance of the system. I think that when to many states are used to model the signal, the model become to specialize to that particular file and it would be difficult to fit an incoming file.

I think that the key part of this project is the selection of the thresholds to be used in the clustering algorithm. Machine learning techniques like genetic algorithms or neural networks may be used to estimate these thresholds.

An important feature of this approach is that concatenated files can be retrieved (actually, I tried but I have troubles with the HTK framework). For instance supposing that a file containing cheering at the beginning and applauses at the end is not modeled in the database, if a input query containing a file with this characteristics would have as output the concatenation of a cheering file with a clapping file. This is an interesting feature that systems like the muscle fish system do not have.

REFERENCES

[1] Gold, Ben. Morgan, Nelson.
Speech and Audio Signal Processing
John Wiley and Sons.

[2] Zhang, Tong. Kuo, Jay
Hierarchical System for Content-based Audio Classification and Retrieval

[3] HTKBook
1999 Entropic Ltd.