INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 666-2900 • FAX (510) 666-2956

ACOUSTIC CHANGE DETECTION AND CLUSTERING ON BROADCAST NEWS

Javier Ferreiros[1], Dan Ellis

TR-00-006

March 2000

Abstract

We have developed a system that breaks input speech into speech segments using an acoustic similarity measure between two segments. The aim is to detect the time points where the acoustic characteristics change. These changes are caused mainly by speaker changes but also by acoustic environment changes. We have also developed another system that performs a clustering of the speech chunks generated by the former system and creates clusters containing the segments with homogeneous acoustic conditions. This clustering is fed back to the acoustic change detector to make more robust decisions based on both the acoustic similarity measurement between two consecutive segments and using extra information coming from the distance between the two clusters to which each of them belong. The interaction between the two systems (acoustic change detection and clustering) improves the results obtained for both aims.

[1] Currently with GTH-UPM, Speech Technology Group, Universidad Politecnica de Madrid, Spain.
Research supported by the Spanish Inter-ministerial Commission of Science and Technology (CICYT).

1. Introduction

1.1 Objectives of the task

Suppose that you have some speech material to be processed. In our case, this material comes from recordings from the radio and television as part of the well-known Broadcast News database [BN 97]. In this material one can find multiple speakers with very different speaking styles and different acoustic conditions, music, and many other sounds. Suppose that part of the material has the following aspect:

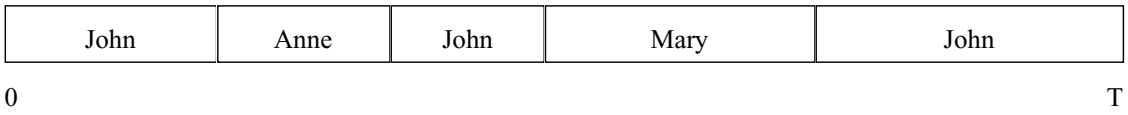

Figure 1: Example of original speech material

Then, the acoustic change detection (ACD) task consists of finding the time points where there is a speaker change (or more generally, an acoustic environment change). So, ACD will just determine the time points t_1, t_2, t_3, t_4 in the following sketch:

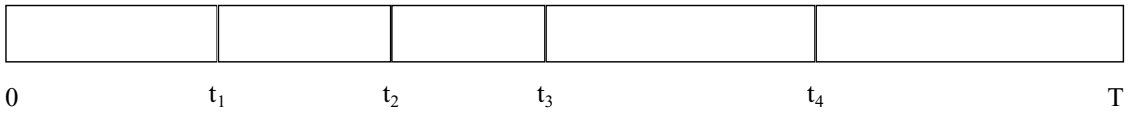

Figure 2: After an ACD task

A clustering task of the acoustic segments consists of generating some clusters containing homogeneous acoustic conditions inside each of them. In our example, this procedure will label each broken segment as belonging to a particular cluster:

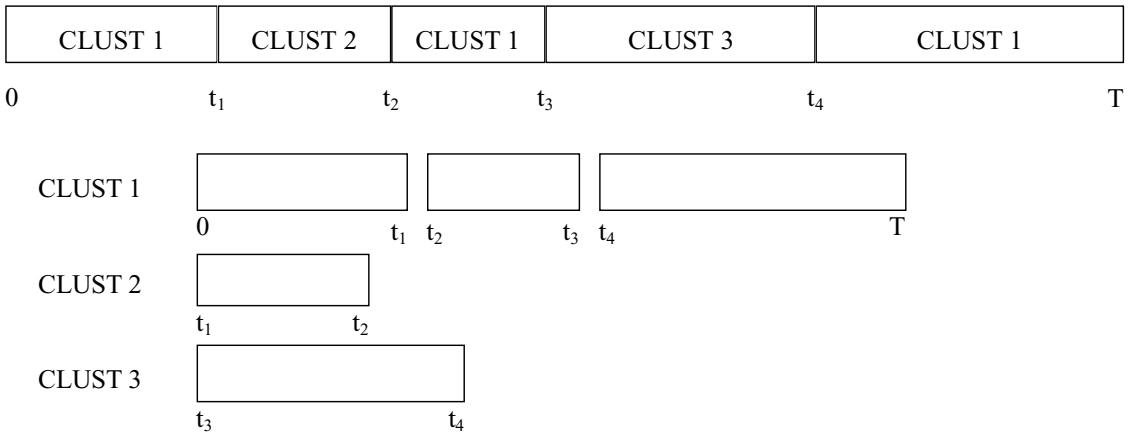

Figure 3: After ACD and clustering tasks

Eventually, if we had some speech material labeled from each of the speakers or even for each acoustic condition, we could properly label the segments with the speaker (acoustic environment condition in general) label. In this way, we obtain both the acoustic change points and the recognition of the acoustic condition in each segment:

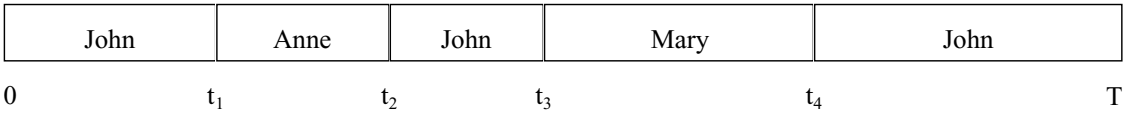

Figure 4: Produced by an acoustic condition labeling task.

The interest of the first task (the basic ACD), besides the pure time marks, can reside in the fact that we are obtaining as a consequence some segments of homogeneous speech. On these segments we can try short-term adaptation techniques when trying to recognize the speech in each segment from break point to break point. It has been also suggested that the language model that could be guiding the speech recognition could be "reset" at these break points to originate a new sentence [Liu 99]. With segments clustering we will of course obtain larger amounts of homogeneous speech material to be used for recognition models adaptation. This clustering information is more robust to make subsequent decisions on acoustic environment identity and can be fed back to the ACD system to perform better segmentation decisions. Completing the tasks with speech recognition and understanding, we can use these labels as indexes for powerful information retrieval.

1.2 Overview

2. Block diagram of the whole system

The whole system that we have developed can be divided into the following main blocks:

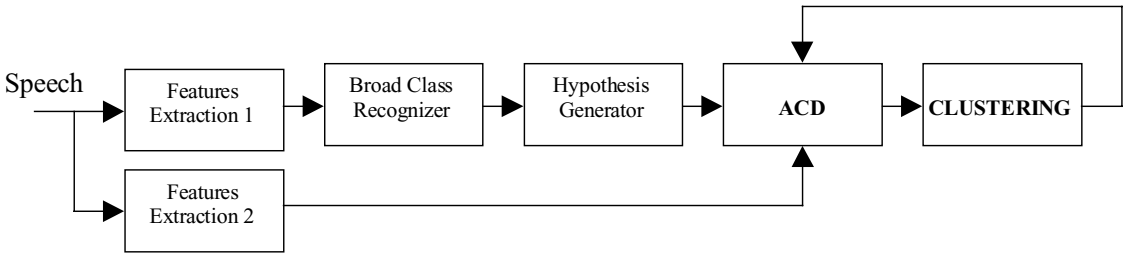

Figure 5: Block diagram of the whole system

2.1 Speech Features Extraction Modules

The first blocks are speech features extraction modules that extract a vector of parameters for each frame defined as a windowed portion of the input waveform. We have depicted two different features extraction modules because it is usually the case that a set of features is more suitable for speech recognition (in our case, for broad class phone recognition) and another

different set is more suitable for speaker recognition. In our case, we have used perceptual filtered cesptrum parameters for our broad class phonetic recognizer and cepstrum parameters without the perceptual filtering for the ACD and clustering systems. Specifically, for ACD (Features Extraction 2 module), we have studied the behavior of the system using different number of cepstral coefficients extracted each 16 msec. We have made experiments (described in a later section) with from as few as 6 to as many as 19 parameter vectors. Some other experiments were also carried out to check the effects of PLP (Perceptual linear predictive [**REFERENCE**]) features, critical band energies, root compression and delta parameters on the performance of the ACD. For the Broad Class Recognizer, we have always used 12^{th} order PLP cepstrum features normalized per sentence that were the features that we found already calculated for other tasks on this part of the database.

2.2 Broad Class Recognizer

We are using this module to generate a rather small number of hypothesized break points (specifically for the experiments presented here, 5180 break points for a total of 686069 frames, i.e. 0.76%, in the current implementation) that the ACD will check. Using the reference hand labeling, we have found that only 2 of the true 617 break points are not in the set of 5180 hypothesized points. This means that the false rejection rate of the ACD that uses this hypothesis system has an absolute minimum of 0.32%, quite a small floor error already introduced by this module. The generation of the hypothesis is based on the initial assumption that the acoustic changes occur always on non-speech zones (i.e. the h# symbol in our usual phonetic labeling scheme) that we can detect making use of a fairly simple neural network. As we have already indicated, this is true for 99.68% of the hand labeled acoustic changes in our target database. The neural network that has been used has been designed to be in fact a broad-class phonetic recognizer for which we have defined the following classes:

$Class_0$	vowels + nasals
$Class_1$	fricatives
$Class_2$	obstruents
$Class_3$	non-speech

Table 1: Broad classes

We have used a 3-layered MLP. The input layer interfaces to 9 consecutive feature input vectors, as is usual at ICSI, to give the NN the opportunity to extract some time evolving characteristics. In the hidden layer we have tried NNs with a range from 200 to 6400 hidden units. We have 4 output units, one for each of the different broad phonetic classes. The NN has been trained using Train-97 Broadcast material, specifically we have used 20 hours of speech for the learning phase, another 2 different hours for cross-validation and still another set of 2 different hours as a development set with which we have obtained the recognition rates presented here. We have evaluated the frame accuracy of the NN when trying to give the proper broad class label to each frame in the development set and we have obtained the following results using 12^{th} order PLP cepstrum features normalized per sentence:

HIDDEN UNITS	% FRAME ACCURACY
200	87.71
400	88.19
800	88.43
1600	88.61
3200	88.62
6400	88.65
Recognition NN	91.09

Table 2: Frame accuracy for broad classes

We have included also the results for a "Recognition NN" that is a neural network trained at ICSI for recognition purposes, specifically the one defined by the weights file:

boot-plp12N-16k-70h-aI5+117i+2000h+54o.wts

That is a NN that uses the same12th order PLP cepstrum + energy input features normalized to zero mean and unit variance within each training utterance with input speech sampled at 16k samples/second. It has been trained with 70 hours of speech against the forced alignment known as aI5 at ICSI. It uses 117 inputs for 13 features with a time span over 9 consecutive frames, 2000 hidden units and 54 phone outputs. We have required that the output probability for non-speech (h#) would be higher than 0.5 to decide whether a frame is not speech. Because the NN uses soft-max output configuration, this will ensure that the h# output will be higher than the sum of the rest of outputs, i.e., the most probable. Of course this bigger and more powerful NN outperform our simpler ones but there are no strong differences in the frame accuracy that would advice to complicate our approach. In the presented results we have found a saturation effect from 1600 hidden units on. This is the reason for us to choose the 1600 hidden units NN as our non-speech zone detector. We can have a view of the confusion matrix for this NN:

	NUMBER OF TIMES LABELED AS				
	vowels + nasals	fricatives	obstruents	non-speech	TOTAL (100%)
vowels + nasals	218000 (92.0%)	7000 (3.0%)	9000 (3.8%)	3000 (1.3%)	237000
fricatives	5000 (7.6%)	54000 (81.8%)	6000 (9.1%)	1000 (1.5%)	66000
obstruents	6000 (8.6%)	6000 (8.6%)	56000 (80.0%)	2000 (2.9%)	70000
non-speech	2000 (3.0%)	2000 (3.0%)	2000 (3.0%)	60000 (90.9%)	66000
					439000

Table 3: Confusion matrix for broad class frame by frame recognition

This net was trained in 5 epochs, three with learning rate equal to 0.008, and two more epochs with 0.004 and 0.002 learning rates using the softmax output type. To our satisfaction, the NN seems to do a pretty good job recognizing non-speech events with low confusion errors with the rest of broad classes. Although in our current implementation we are using only these regions where the neural network detect non-speech zones in the way we will explain later, we have designed a 4 broad classes recognizer because we want also to check at some point the behavior of

the system if we would have hypothesized also as possible break points those where the NN detects a change between two different classes. We devise two possible effects from this alternative approach. On one hand, we expect that we could account for these 2 hand labeled changes in our target database that are not covered with the current NN. Of course, this will mean that many more possible break points will be hypothesized by the NN and this in turn will make harder for the ACD to maintain its performance although the new absolute minimum rejection rate be 0%. But, on the other hand, we expect that we will be able to decrease the probability of missing some high-overlap acoustic change break points (which may be the case for these 2 misses) where two speakers overlap, an effect that could be quite common in other tasks of spontaneous speech such as the speech from a meeting recorder system. Anyway, throughout the work being presented here, we have just used the zones with more than 3 consecutive frames being labeled by the NN as the symbol "h#" (broad class 3, non-speech) as the hypothesis break points to be further analyzed by the ACD system.

3. Acoustic Similarity Test based on the Bayesian Information Criterion

3.1 The Generalized Likelihood Test with Bayesian Information Criterion

To calculate the similarity between two acoustic segments, i.e., between two sets of feature vectors, we will use a likelihood ratio test based on the Bayesian Information Criterion (BIC). In general, BIC is a likelihood measurement penalized by the complexity of the assumed model [Shaobing 98]. If we have a set of N vectors $X=\{x_i : i=0,...,N-1\}$ that we are trying to represent through a model M:

$$BIC(M) = \log[L(X,M)] - \lambda \cdot \frac{1}{2} \cdot \#(M) \cdot \log(N) \qquad (1)$$

where λ should be 1 according to the theoretical justification of BIC. #(M) is the number of parameters of the model as a measure of its complexity. Bearing this BIC in mind, we formulate our general segmentation problem as a test between two hypothesis: We have a segment of speech and a possible break point and we want to decide if either we should consider the whole segment coming from the same acoustic environment or we should better consider that there are two acoustically different segments we should break apart using the break point. Thus, if we have the following situation:

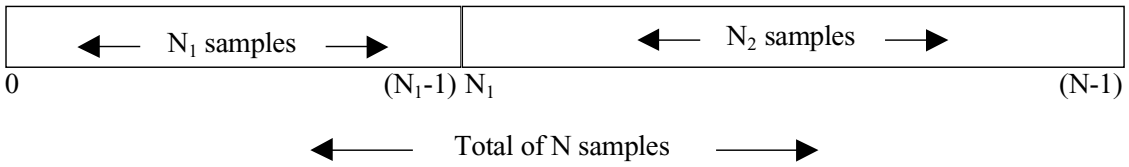

Figure 6: Acoustic similarity test situation

If we choose single gaussian models, we have the two following hypothesis:

H_0: $x_0 \ldots x_{N-1} \sim N(\sigma,\Sigma)$

H_1: $x_0 \ldots x_{N_1-1} \sim N(\sigma_1, \Sigma_1)$
$x_{N_1} \ldots x_{N-1} \sim N(\sigma_2, \Sigma_2)$

Now, we will perform a generalized likelihood ratio test of these two hypotheses that would in general calculate the following ratio [Liu 99]:

$$\frac{L[\{x_0 \ldots x_{N-1}\}/N(\mu, \Sigma)]}{L[\{x_0 \ldots x_{N1-1}\}/N(\mu_1, \Sigma_1)] \cdot L[\{x_{N1} \ldots x_{N-1}\}/N(\mu_2, \Sigma_2)]} \qquad (2)$$

We conclude H0 if the ratio is higher than 1. Using the BIC, and keeping from the likelihood estimation only the part that is dependent on the covariance model [Liu 91] (and consequently independent of the feature means, which can be strongly affected by any filtering difference) we end up with the following hypothesis test:

$$N \cdot \log|\Sigma| - N_1 \cdot \log|\Sigma_1| - N_2 \cdot \log|\Sigma_2| - \lambda \cdot 1/2 \cdot (d + 1/2 \cdot d \cdot (d+1)) \cdot \log(N) \gtrless_{H_0}^{H_1} 0 \qquad (3)$$

If this number is positive, we decide H1. If it is negative, we decide H0. Thus, if this number if positive we break the whole segment into the two sub-segments. In this formula we have introduced the penalty of the complexity model via the factor $d + 1/2 \cdot d \cdot (d+1)$ that is the number of free parameters for a full covariance single gaussian model for d-dimensional feature vectors. Although the parameter λ should be theoretically 1, we have found throughout our experiments that its value had to be tuned in order to obtain the desired performance. We have found a strong dependency of its optimal value with the dimensionality of the input parameter vector (d) that it is not compensated for by the model complexity factor.

3.2 Basic Acoustic Change Detection Procedure

Once we know how to evaluate if we should or we should not break a speech segment into two given a possible break point, we face the problem of dynamically produce the acoustic change detection of a large amount of speech. Instead of hypothesizing a possible break point on each input frame, the non-speech zones detector based on the broad-class NN recognizer will deliver a reduced number of frames to check (5180 hypothesis from 686069 frames in our experiments). Because the outputs of the NN have some "noise", sometimes the non-speech output fires higher than the other three outputs for only one frame or two. We have filtered out these "glitches" and have only allowed registering segments where the non-speech identity is detected for three or more frames. Of course, these non-speech regions are excluded from our acoustic similarity calculations. Then, we hypothesized the center of the non-speech labeled region as the possible break point that will try to hit one of the true target regions. Thus, these initial break points hypothesis define the maximum possible partition of the speech material into small segments. Using these basic segments, we run the following procedure:

1) Add another segment into a buffer and write down a new possible break point.

2) Calculate Likelihood Ratio via the BIC for each hypothesis in the buffer. Find the most likely break point (the one with the highest likelihood ratio).

3) Is the best likelihood ratio test positive?
YES: Produce a break point. Delete segmented zone from buffer.
Go to 1).
NO: Go to 1).

4. Experimental setup

For the evaluation of our systems we have used the same database than [Liu 99], [Shaobing 98], that is, 3 hours of speech that were defined as the Hub 4 1997 evaluation data. The hand labeled transcriptions define 618 segments in this database (617 breakpoints) that are mostly speaker changes but also include changes to and from music, silence, excluded regions, etc. We will consider them all as true target break points. There are 119 different labels in the hand labeling of these segments, most of them the speaker proper names but also some others as: generic speaker labels (like "CNN_WVW_mAnnouncer1" or "female_nonnative3") and special labels (like "BEGIN", "Inter_segment_gap" or "Excluded_Region"). Following the evaluation directions in [Liu 99] for the ACD task, we have converted the break points into valid break regions that extend with the non-speech region around a certain breakpoint. This extension is produced using the labeling obtained by using the neural network trained at ICSI for recognition purposes that we have already presented when we compared our non-speech NN detector to it, i.e., the one defined by the weights file "boot-plp12N-16k-70h-aI5+117i+2000h+54o.wts". 6 frames of 16 msec. (about 100 msec.) are added to both extremes of these regions as an allowance margin. With these criteria, we have defined the final target regions for our break points evaluation. We present the performance as two figures: the false rejection and false acceptance percentage rates defined as:

$$\%\ \text{False Rejection} = 100 \cdot \frac{\text{TARGETS} - \text{OK}}{\text{TARGETS}} = 100 \cdot \left(1 - \frac{\text{OK}}{\text{TARGETS}}\right)$$

$$\%\ \text{False Acceptance} = 100 \cdot \frac{\text{GENERATED} - \text{OK}}{\text{GENERATED}} = 100 \cdot \left(1 - \frac{\text{OK}}{\text{GENERATED}}\right)$$

where TARGETS are the number of target regions, GENERATED are the number of generated break points and OK are the number of the generated break points that fell within a correct target. Only one generated boundary can be matched to each target region. Different performances can be obtained tuning the value of λ. We can design a large variety of systems from the ones having high false acceptance with low false rejection to those performing low false acceptance with high false rejection. That is why when comparing results we have used the Equal Error Rate (EER) working point where the system has the same false acceptance and rejection rates. At this point, you can check looking at the formulas that we force the system to generate exactly the same number of segments than the one in the reference (617 in our case).

Another task that we had to evaluate was the clustering of the segments produced in a ACD session. Because the ACD is not perfect, each component segment contains speech for different acoustic conditions, not just for one. Eventually, we obtain a set of clusters composed of several segments each and, in turn, with several acoustic conditions inside each of these segments. Using again the hand labeling, we may find something like:

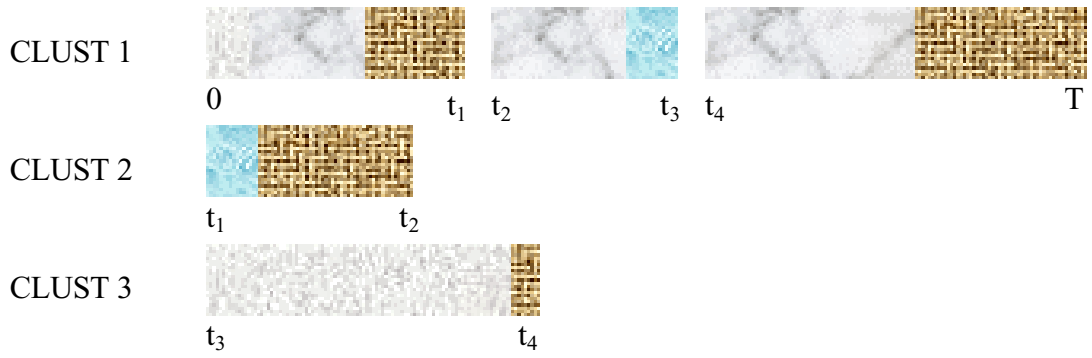

Figure 7: Example of clustering evaluation

where we are representing graphically the material coming from different speakers as different textures. The first measurement that could be done may be called simple average purity and would be calculated as:

$$\%SAP = \frac{\sum_{i=1}^{NCLUSTERS} \max_{j}(C_{ij})}{\sum_{i=1}^{NCLUSTERS} \sum_{j=1}^{NSPEAKERS_i} C_{ij}} \quad (4)$$

where C_{ij} is the number of frames in cluster i that actually belongs to speaker j.

This formula is calculating the proportion of all frames that belong to the majority speaker in their cluster. Thus, this way of calculating the purity of the clustering assumes that the speaker that has the higher number of frames in a cluster is the one that this cluster is trying to represent and the rest of frames are incorrect frames for this cluster. Of course, the same speaker can be represented in different clusters with this approach, but we have decided not to control this effect because it may be the case that different clusters are representing the same speaker but with different acoustic conditions. The problem that we see in this formula is that (following our example) the frames

in cluster 2 will contribute with the same intensity to the numerator of the equation than the frames

in cluster 3, although the latter seem more "pure" in their own cluster. From this thought, we arrived to the following weighted average purity:

$$\%WAP = \frac{\sum_{i=1}^{NCLUSTERS} \max_j(C_{ij}) \cdot \frac{\max_j(C_{ij})}{NSPEAKERS_i}}{\sum_{i=1}^{NCLUSTERS} \sum_{j=1}^{NSPEAKERS_i} C_{ij}} \quad (5)$$

One of the problems of the purity measurement is that as long as we leave the system generate more clusters, they become more and more pure without much effort or relevance. In the limit, each cluster could be just only one frame that has to belong to one speaker and the %WAP is 100% (of course, the actual lower limit for us is in the intrinsic purity of the segments that the ACD system generated). Thus, we also give another measurement that is the weighted average purity per cluster:

$$\%WAP/cluster = \frac{\%WAP}{NCLUSTERS} \quad (6)$$

that somewhat penalizes the generation of too many clusters. We have also used in some experiments an entropy measurement of the information contained in the clustering as:

$$Entropy = -\sum_i \sum_j \frac{C_{ij}}{\underset{j}{sum}(C_{ij})} \cdot \log_2\left(\frac{C_{ij}}{\underset{j}{sum}(C_{ij})}\right) \text{ (bits)} \quad (7)$$

The measurement that we have found more correlated with the %EER ACD performance is the %WAP/cluster. The entropy measurement suffers also from a lowering (improvement) as long as the number of clusters increases. Anyway, neither of these measurements can be used for a convergence criterion, they are just evaluations of the clustering quality against a known hand labeling that we will not have in a real application. This is the reason why we will use a no-changes stopping criterion mixed with a maximum number of iterations in the iterative clustering improvement algorithm that we will introduce later.

4.1 First Experiments

Our first experiments aimed the use of simple diagonal covariance matrices for the gaussian models. We began also by using high dimensional feature vectors of 19 cepstrum features. For this case, _=1 nearly produced the EER. Unfortunately, this has been the only case to hold that and appears to be a coincidence. We obtained the following performance:

%FR	%FA	N
30.36	29.67	610

Table 4: ACD performance for 19 cepstrum features and diagonal covariance models

where %FR is the % False Rejection Rate, %FA is the % False Acceptance Rate, and N is the number of GENERATED breakpoints (the hand labeled number of true break points is 617). Thus, our starting point is a system performing EER of about 30%. The behavior of the false rejection and false acceptance rates as we change the value of the tuning parameter λ can be seen in the following illustration:

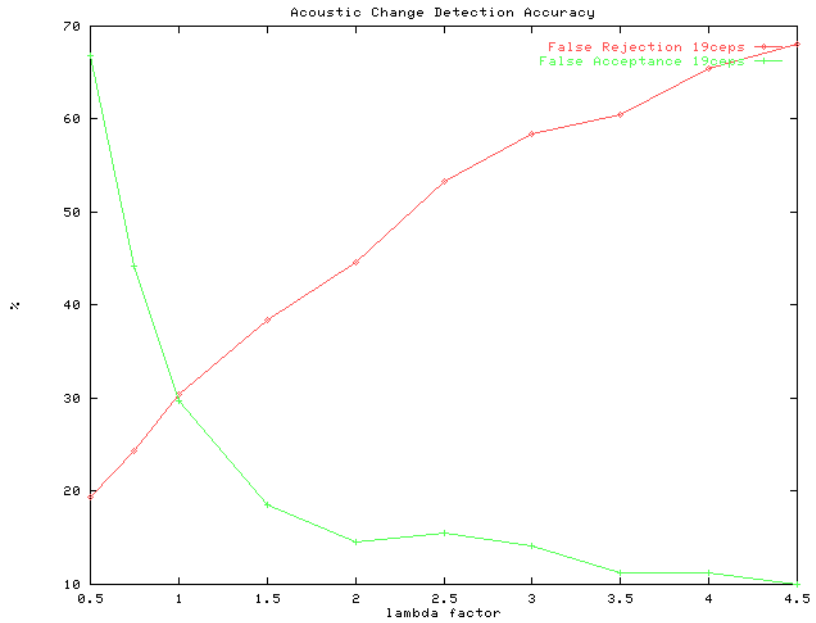

Figure 8: 19ceps

With diagonal covariance models we also evaluated the use of cube root compression of the parameters and this did not alter the results. Also, we used PLP features and the system performed very poorly. This is good news to ICSI because it confirms once more that PLP processing smoothes out acoustic environmental differences while leaving the relevant information for speech recognition. We have also used plain log-energy band features with results in between PLP and plain cepstrum features. Thus, we have selected cesptrum

Next experiments where devoted to the introduction of full covariance matrix for our gaussian models. With the same 19 cepstrum feature vectors we found:

%FR	%FA	N
27.27	27.04	614

Table 5: ACD performance for 19 cepstrum features and full covariance models

This means about 27% EER.

At this point, we made a study of the evolution of the performance of this system with the dimensionality of the vectors being used. We are using all the time cepstrum no-plp feature vectors. The results are the following:

Number of features	%EER
6	32
7	30.5
8	29.5
9	29.5
10	27.5
11	26.5
12	26.5
13	26.5
14	26
15	26.5
19	27

Table 6: ACD performance evolution for different numbers of cepstrum features and full covariance models

These results suggest that about 11 cepstrum no-plp feature vectors are enough to obtain the best performance from the current implementation. It seems also that there are no significant changes in the behavior of the system from 11 to 15 parameters. We think that the slightly lower performance for the 19 features system is due to the increasing number of parameters of the gaussian models that may cause trouble trying to identify some fast changes (few frames to train the model for these cases).

4.2 Clustering of ACD generated segments

Using the same likelihood ratio with the bayesian information criterion, we can check if an acoustic segment belong to anyone of different pools of acoustic segments that we will be calling from now on clusters. To do this, we first calculate:

$$MINRATIO = \min\left(\begin{array}{l} (NC_i + N)\cdot \log\left|\Sigma_{C_i \cup S}\right| - N\cdot \log\left|\Sigma_S\right| - NC_i \cdot \log\left|\Sigma_{C_i}\right| \\ -\ \lambda \cdot 1/2 \cdot (\mathrm{d} + 1/2 \cdot \mathrm{d} \cdot (\mathrm{d}+1)) \cdot \log(\mathrm{NC_i} + \mathrm{N}) \end{array} \right)$$
$$i$$

where

N is the number of samples in segment S
NCi is the number of samples in cluster Ci

and then check if this MINRATIO if greater than zero. If it is positive, this means that we should separate the segment S from the cluster Ci, even though Ci is the cluster with the minimum MINRATIO. In this case, we are generating a new cluster composed of the segment S alone by now. If MINRATIO is negative, this means that we should consider the segment S being the same acoustic condition than the one represented by centroid Ci. For this case, we update Ci data accordingly so as to consider the segment S in it. If we repeat this procedure segment by segment for those generated by the ACD system, we generate a clustering of the speech material in several classes automatically defined by the BIC likelihood ratio. The number of classes (of different clusters) is dependent on the value of λ. If we choose high values of λ, it is easier for a new acoustic segment to fall inside the region of pertinence of a previously existing cluster and we end up with few clusters. On the other hand, if we use low values of λ, it is more likely that the minimum ratio remains still positive and we end up with a higher number of clusters. One problem of this approach to acoustic clustering is that the first segments are classified using far less information than the last ones. Thus, the decision is increasingly more robust, but could be very bad for the first processed segments. To address this problem, we have introduced an iterative improving algorithm that we discuss in the next section.

4.3 Iterative improving algorithm of the clustering

To counteract the effect related in the first approach to acoustic clustering of bad decisions possibly being made on the first segments, We have introduced an iterative improvement procedure that will revisit the decision made on each segment. We will even allow new clusters to be generated.

The basic iteration is:

1) Pick one ACD segment.
2) Remove this segment from its cluster and update cluster data
3) Find the "closest" cluster (if any) to the segment
4) If there is a representative cluster, go to 6)
5) Generate a new cluster with only this segment in it. Goto 7)
6) Update this cluster with the segment information.
7) If last segment then stop, else go to 1)

The algorithm performs several iterations like this one and stops when no change of cluster is produced for any ACD segment. We have also used another limit of 10 iterations as a maximum number of allowed iterations. Most of the experiments used less than 6 iterations before stopping. This algorithm has allowed us to improve significantly the weighted average purity of our

clustering. As a way to compare clustering performance and reveal clearly this improvement, we have realized some experiments tuning the λ factor until we get 119 clusters. We chose 119 clusters because it is the number of true labels in the hand labeling. Under this condition and using 12 cepstrum parameters and the breakpoints generated by the ACD that performed %FR=26.79 and %FA=26.67 (the one listed before as having about 26.5% EER) with 615 cuts, we obtained the following results:

	λ	%WAP	%WAP/cluster
Diagonal covariance	4.1	74.58	0.627
Full Covariance	8.5	81.12	0.682
Full Covariance + Improvement algorithm	9.8	84.86	0.713

Table 7: Clustering performance with diagonal covariance models, full covariance models and full covariance models with the improvement algorithm. We measure the weighted average purity and the weighted average purity per cluster.

In the first line the baseline performance can be seen. This is the performance of a simple diagonal covariance model. The first column is the λ factor that allows 119 clusters to be generated. The other two columns contain the weighted average purity and this purity divided by 119, the number of clusters that we have generated. In the second line we have the results for a full covariance model that represent a relevant improvement in itself. The λ factor has to be tuned to a different value in order to obtain 119 clusters again and allow the comparison. In the third line, we present the new results obtained using our improvement algorithm. Again, the λ factor has to be tuned to a different value to obtain 119 clusters.

4.4 Using clustering information in ACD

Clustering defines a set of classes that represent the separate acoustic conditions in our data. This information can be used to create a new segmentation: rather than comparing adjacent segments based on their own properties alone, we can assign them both to existing clusters and add to the breaking decision the information coming from this classification. Following the BIC criterion and considering the log-likelihoods as "distances", we can think that a segment belongs to the class defined by the cluster calculated as:

$$C(S) = \arg\min_i((NC_i + N)\cdot\log\left|\Sigma_{C_i \cup S}\right| - N\cdot\log\left|\Sigma_S\right| - NC_i\cdot\log\left|\Sigma_{C_i}\right|) \qquad (8)$$

where

C(S) is the cluster to which the segment S belongs
N is the number of samples in S
NCi is the number of samples in cluster Ci

independently now of whether the value inside the minimum argument function is greater or lower that the threshold:

$$\lambda \cdot 1/2 \cdot (\mathrm{d} + 1/2 \cdot \mathrm{d} \cdot (\mathrm{d}+1)) \cdot \log(\mathrm{NC_i} + \mathrm{N})$$

So, as a first approach we could allow a break only when the two segments to be separated belong to different clusters. It is a simple condition but a very powerful one because the clusters have been generated and optimized with the entire speech file. This classification procedure has also being used to evaluate the performance of our clustering improvement algorithm. While in the first produced clustering, there were a lot of classification errors of the components of each cluster, after the improvement algorithm was run, very few and in most of the experiments no classification errors of the cluster components have been found.

Soft integration of clustering information on ACD

Instead of just checking whether the two segments to be separated belong to different clusters, we have also tried a soft integration of the clustering information into the ACD general likelihood ratio test. The idea we were pursuing is to integrate another BIC measurement dependent on the clusters to which the two segments belong. If we think of likelihood values as if they were distances, we could define generically a distance between two acoustic segments as:

$$d(X,Y) = (N_X + N_Y) \cdot \log\left|\Sigma_{X \cup Y}\right| - N_X \cdot \log\left|\Sigma_X\right| - N_Y \cdot \log\left|\Sigma_Y\right|$$

Using this distance, the first idea is to use the function

$$G = d(C_1, C_2) = (N_{C_1} + N_{C_2}) \cdot \log\left|\Sigma_{C_1 \cup C_2}\right| - N_{C_1} \cdot \log\left|\Sigma_{C_1}\right| - N_{C_2} \cdot \log\left|\Sigma_{C_2}\right|$$

where

C1 is the cluster to which the segment S1 belongs
C2 is the cluster to which the segment S2 belongs

as extra information about the suitability of separating S1 from S2. Assuming that C1 and C2 are good representatives of S1 and S2 correspondingly, the "distance" from one to the other will be greater when we should separate S1 from S2 and ideally 0 (because C1=C2) when S1 and S2 come from the same acoustic conditions. We wish to integrate this information with that from the old one we have being using:

$$F = N \cdot \log|\Sigma| - N_1 \cdot \log\left|\Sigma_1\right| - N_2 \cdot \log\left|\Sigma_2\right|$$

To begin with, we give the same relevance to the information conveyed by F than to the one given by G, we decided as a first approach to equalize their dynamic range with the following formula:

$$F + \frac{\sigma_F}{\sigma_G} \cdot G - \lambda \cdot 1/2 \cdot (\mathrm{d} + 1/2 \cdot \mathrm{d} \cdot (\mathrm{d}+1)) \cdot \log(\mathrm{N})$$

where σ_F and σ_G are fixed estimates of the standard deviations of F and G respectively. Basically, we found first an approximate experimental value for this equalizing parameter and then

we realized that it was close to this ratio for the samples we had and then decided to substitute the experimental value by this more justifiable one. We want to stress that this is a fixed value (it is not dynamically changing). We then use the same criterion of separating the two segments if this value is positive. The value λ has to be tuned again to different values to obtain EER. To unbalance the relevance of each sources of information, we used the formula:

$$(1-\alpha)\cdot F + \alpha\cdot\frac{\sigma_F}{\sigma_G}\cdot G - \lambda\cdot 1/2\cdot(\mathrm{d}+1/2\cdot\mathrm{d}\cdot(\mathrm{d}+1))\cdot\log(\mathrm{N})$$

where α can be varied from 0 (only the baseline F function takes control on the decisions) to 1 (only the new G function decides).

finally, we have tried a dynamic α with the expression:

$$\alpha = e^{-\frac{d(S_1,C_1)+d(S_2,C_2)}{factor}}$$

α is 1 if d(S1,C2) and d(S2,C2) are both zero. In this case, we know that C1 and C2 are perfect representatives of the segments S1 and S2 and this is an indication that we can thoroughly rely on the robust information given by the function G of the distance between clusters. On the other hand, α will be close to zero when these distances will be large and C1 and C2 would be in this case bad representatives of the acoustic segments. In this case we prefer not to rely so much on the clustering information, but give more relevance to the information in the function F.

4.5 Iterating ACD with clustering improvement

Another strategy we have tested is to run several iterations between the ACD using clustering information and the clustering generation and improvement algorithm. We begin by using the ACD that does not need clustering information to produce the first segmentation of the acoustic material. With this segmentation we produce and improve a first clustering. The clustering information is fed back to an ACD that uses this information and so on. Several combinations of the systems presented have been tried. The first one was using the clustering information using a "hard decision": only allowing a break when the two hypothesized segments belong to different classes. We have used a different λ for the ACD procedures than for the clustering generation and improvement ones. In fact, we have found the best results when the tuning parameter λ for the clustering is higher than the one used in ACD. A high value in the clustering procedures tends to generate fewer clusters and a low value in the ACD tends to generate more break points. Thus, our guess is that combination of higher values for clustering than for ACD produces initially a high number of segments that are pooled in few clusters by the clustering generation and improvement procedure. This clustering will in turn be a higher demanding constraint for the following ACD that begins to generate fewer segments. This dynamics tend to converge to an intermediate number of segments and clusters through the iterations. This combination yielded:

%FR	%FA	N
20.94	21.32	619

Thus, about 21% EER, a great leap forward from the systems that do not use clustering information. We force the system to perform 10 iterations of the loop ACD-clustering, although these results were obtained already in the fifth iteration and then the results stabilized. The second combination we tested is the use of soft integration of the clustering information into ACD using the formula:

$$F + \frac{\sigma_F}{\sigma_G} \cdot G - \lambda \cdot 1/2 \cdot (\mathrm{d} + 1/2 \cdot \mathrm{d} \cdot (\mathrm{d}+1)) \cdot \log(\mathrm{N})$$

that is, giving the same relevance to the information coming from both the functions F and G. No significant differences were found and on the other hand, the system took more CPU time to calculate the function G. The system performed:

%FR	%FA	N
20.29	20.93	618

The last combination tested was the use of the unbalancing function α that we talked about:

$$(1-\alpha) \cdot F + \alpha \cdot \frac{\sigma_F}{\sigma_G} \cdot G - \lambda \cdot 1/2 \cdot (\mathrm{d} + 1/2 \cdot \mathrm{d} \cdot (\mathrm{d}+1)) \cdot \log(\mathrm{N})$$

with

$$\alpha = e^{-\frac{d(S_1,C_1)+d(S_2,C_2)}{factor}}$$

but, unfortunately, only an insignificant improvement have been found again. The best we got is:

%FR	%FA	N
20.29	20.29	617

Thus, we are close to 20% EER. We have to keep in mind that, with a confidence of 95% and for 617 true targets, the actual rates may be between 17.1% and 23.5%. So, all this last performances are surely the same. To know how far we are from the practical limit for this idea of feeding back clustering information to the ACD system, we have used the true hand labels to generate the acoustic segments that the human expert indicated. With these segments we passed the clustering generation and improvement system and the clustering obtained was used for ACD with soft integration. The result obtained was:

%FR	%FA	N
19.16	18.49	611

This means that we were close to the practical limit for this idea even with the hard-decision integration of the clustering information and this may be the cause for the lack of success in the rest of combinations. The rest of the remaining errors must be related to other effects not solved with this idea.

5. Other issues

5.1 CPU time issues

Mainly inspired by the need of more speed to allow the iterative experiments we have presented between the ACD with clustering information and the clustering improvement algorithm, we have tried to optimize the CPU time used. The underlying idea is well known but we will repeat it here. When estimating a gaussian model, we need to estimate both a mean vector and a triangular-superior covariance matrix. The mean vector is calculated as the expectation of the data vectors, i.e., for each vector component we divide the sum of the value of this component in each data vector by the number of vectors.

$$sum\_old_j = \sum_{i=1}^{N} x\_orig_{ij}$$

$$\mu\_old_j = \frac{sum\_old_j}{N}$$

If we need to re-compute this mean vector after adding some new data vectors to the same pool, it would be wise to have saved the old values of each component sum (sum_old_j) and the number of vectors used to estimate it (N). For the new set of N_2 vectors to be pooled with the old ones, we also calculate the sum of components:

$$sum\_new_j = \sum_{i=1}^{N_2} x\_new_{ij}$$

thus, the new mean vector is:

$$\mu\_new_j = \frac{(sum\_old_j + sum\_new_j)}{N + N_2}$$

Something very close happens with the estimation of the covariance matrix. In this case, we need also to save the sum of all possible products of components:

$$prod\_sum\_old_{jk} = \sum_{i=1}^{N} x\_orig_{ij} \cdot x\_orig_{ik}$$

The covariance matrix will then be:

$$co\_old_{jk} = \frac{prod\_sum\_old_{jk}}{N} - mean\_old_j \cdot mean\_old_k$$

And when we need to correct the covariance matrix with new data, we calculate the sum of products for the new data:

$$prod\_sum\_new_{jk} = \sum_{i=1}^{N_2} x\_new_{ij} \cdot x\_new_{ik}$$

and then update the covariance matrix to:

$$co\_new_{jk} = \frac{prod\_sum\_new_{jk}}{(N + N_2)} - mean\_new_j \cdot mean\_new_k$$

With this organization of the calculations, we can rapidly compute new combinations of pooled data just saving the sum of the components and of the squared components of the segments of speech without the need of using the actual values of the data vectors. For our ACD problem we will need to test several hypothesis until one of them is satisfied and a break is produced. In this moment, all the data until the breakpoint can be deleted. The part from the actual break to the last considered segment and other new segments will be involved in the new calculations to find the following break point. This procedure suggest the use of a circular memory buffer that will hold the data of each acoustic segment between two consecutive break point hypothesis in a different place. The perimeter of this circular buffer should be long enough to hold the worst case, i.e., the maximum number of hypothesized break points without an actual break. In our experiments we have found a buffer length of 200 hypothesis to work properly. With this organization of the calculations, we have gone from the first versions that processed 3 hours of speech in 57 minutes Ultra-Sparc CPU time to the improved version that does the same job in less than 2 minutes on the same processor.

5.2 Improving the capabilities to detect fast changes

It is a problem to generate a d-dimensional gaussian model from few frames for common values of d, particularly when trying to estimate the $d(d+1)/2$ covariance parameters. In our first implementations we did not allow the algorithm to produce a break for short segments because the system could not reliably decide with so few information. This minimum was set to 30 frames (480 msec. of speech). Another limit that we also had in the first was that we only processed speech when we had more than 2 seconds in the analysis buffer. When we plotted the histogram of the lengths of the segments produced after ACD and compared them to the hand labeling ones, we realized that there was indeed an effect that our system was producing many fewer short duration segments than the hand labeling. This histogram is depicted in the following illustration:

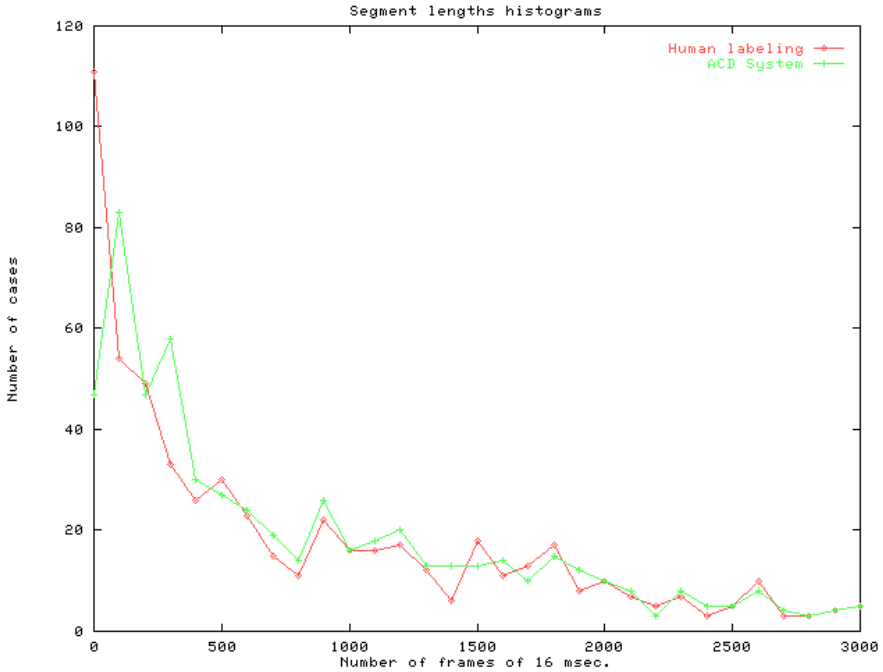

Figure 9: histoold

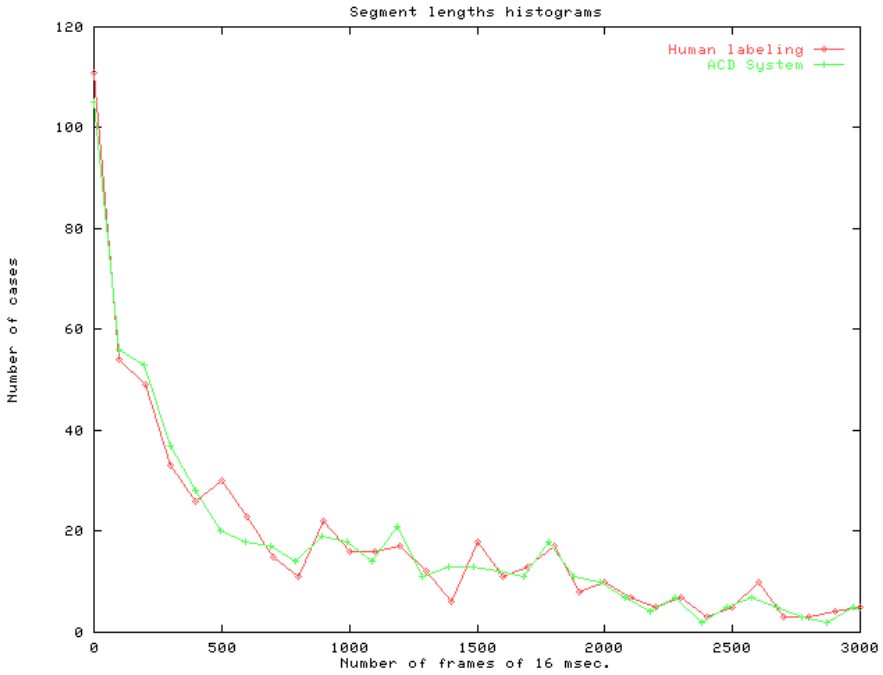

Figure 10: histonew

With the guess that this effect was producing a lack of better performance and the flatness of the ROC curve for low values of λ, we decided to be less demanding in the decision of when a gaussian could be estimated. Basically, now we only check for the determinant of the covariance matrix not to be singular. In the rest of cases, we will rely on the result although it may come from very little data. We also removed the limit of the analysis buffer and begun the process as soon as we have two segments in it. This strategy produced a small improvement of the ACD accuracy allowing us to reach less than 18% EER. Also, the histograms were now closer to the hand labeling for short segments. We have still a flat region in the ROC that may be produced by the new limitation, but now we can force the system to lower false rejection rates in this region. These can be seen in the following illustrations for the new systems. The segment lengths are distributed as following:

Anyway, we have to say again that the differences are not significant with the systems we had before for this testing database. We consider this a small improvement just in this region of fast acoustic changes.

6. CONCLUSIONS

With the experiments that we have made on this problem so far, our final recommendation would be to use a full covariance gaussian model for not many more than 12 cepstrum parameters + energy features, controlling that the determinant of the covariance matrix is not singular and running a few iterations of the loop composed of our clustering generation and improvement algorithm and the ACD that uses the hard-decision based on allowing a break only when the two segments belong to different clusters.

REFERENCES

[Her 90] Hynek Hermansky, Perceptual linear predictive (PLP) analysis of speech, J. Acoust. Soc. Am. 87 (4), April 1990, pp. 1738-1752

[BN97]
NIST, 1997 Broadcast News Speech Corpus,
CSR-V, Hub 4, Produced by the Linguistic Data Consortium, Catalog No. LDC98S71 1997.
http://morph.ldc.upenn.edu/Catalog/LDC98S71.html

[Shaobing 98]
Scott Shaobing Chen, P.S. Gopalakrishnan, "Speaker, environment and channel change detection and clustering via the bayesian information criterion", 1998 DARPA Broadcast News Transcription & Understanding Workshop.

[Liu 99]
Daben Liu, Francis Kubala, "Fast speaker change detection for broadcast news transcription and indexing", Eurospeech 99.