Speech recognition
-
Testing of non-stationary noise suppression method

Pablo Ros Flores
Ending thesis
Hochschule Niederrhein (University of applied sciences)
ERASMUS - Summer semester 2009
**Contents**

1. Introduction and hypothesis ................................................................. 2

2. The speech recognition process .......................................................... 4
   2.1 Front-end feature extraction algorithm ........................................... 4
      2.1.1 Offset compensation ............................................................. 4
      2.1.2 Framing .............................................................................. 5
      2.1.3 Energy measure ................................................................. 6
      2.1.4 Pre-emphasis and Windowing ............................................... 6
      2.1.5 FFT .................................................................................. 7
      2.1.6 Mel Filter bank ................................................................. 7
      2.1.7 Non-linear transformation .................................................... 8
      2.1.8 Cepstral coefficients ......................................................... 9
   2.2 Speech boundaries ......................................................................... 12
   2.3 The statistical approach .................................................................. 14
   2.4 Modeling speech recognition: HMMs ............................................ 16
   2.5 Adaptation of HMMs .................................................................. 20

3. Garbage recognition process ............................................................... 21
   3.1 Database and cepstral analysis .................................................... 22
   3.2 Previous recognition .................................................................. 24
   3.4 Garbage model training ................................................................ 29
   3.5 Evaluation of the system .............................................................. 35

4. Results and conclusions ....................................................................... 38
   4.1 Previous results .......................................................................... 39
   4.2 Synthesis files ............................................................................ 40
   4.3 Experiments ................................................................................ 41

5. Resources ........................................................................................... 51
1. Introduction and hypothesis

The speech recognition it is one of the parts of the artificial intelligence that has become more challenging for the humans. The goal to allow the spoken communication between human beings and computers has not been and it is not an easy task. The problem that arises with a system of automatic speech recognition is to cooperate a set of informations that come from different sources of knowledge (acoustics, phonetics, phonology, lexicon, syntactic, semantic and pragmatic) in presence of ambiguities, uncertainties, and inevitable errors in order to achieve an acceptable interpretation of the acoustic message received.

The purpose of this project is to find an specific improvement inside the wide area of the inductive learning; this means, the techniques that try to obtain the necessary knowledge with real examples above the task that we want to model (hidden markov models).

The experience has demonstrated that one of the main problems when we run some recognition experiments beyond the own nature of the language, it is the noise which adds to the signal (channel, atmosphere, reverberation, etc). As we will see along the development, the main posing nowadays brings us to the modification of some parameters during the training phase. This gives us a wider actuation range compared to the modifications over the the input signal characteristics.

For that reason the actual techniques are focused on the modification of the the patterns of the recognition process according to the acoustic conditions. This could bring us to think as a first approach the development of a system trained with data recorded (or thanks to artificial addition) in adverse conditions. In most of the cases, the best performance will be reached when we apply the reference patterns that have been trained exactly under the same conditions that will be there where we will do our recognition.

This however, is not always the most optimum action. What if we are talking about non-stationary noises. What if I am using my cellular in my car and I have some people talking next to me or in the back seats, the exterior noise, the engine. Some of them are more predictable noises than others, but usually we have an endless amount of different noises that could perturb the communication human-machine.

We could train systems that have samples of all possible conditions (multi-condition). But we will have still a problem; no matter if we train 10, 100 or 1000 different examples as there will keep on...
being interminable atmospheres and noises out of reach of our system – apart from the extraordinary work to carry all this task of recordings and database creations out. As we said then, it seems more clever to attack over the recognition patterns through, for example, the adaptation of this references thanks to the estimation of acoustic conditions.

From Germany in the “Hochsule Niederrhein” (electronics and computers department) - where I have been working on this project – under the leadership of H.G. Hirsch professor, they present an adaptation of the energetic and spectral parameters to noisy and reverberant atmospheres. This has been my reference point all along this project. How could we try to improve the performance of a system which is already very optimum and thanks to recent database creation (Aurora-5)?

Let us try to set an addition by creating thanks to this database a model made for the non-stationary noises recognition. If we have a system which is capable to distinguish until some rates the words even if they are “dirty”, it sounds fine to add one more model to avoid one precise thing:

- Do not let the system think because of noise (no speech signal) that there is some useful information to be recognized.

We will try to find a way to make that a file which originally looks like that:

![original sections](image1)

not to mistake due to the presence of noise:

![recognized sections](image2)
2. The speech recognition process

2.1 Front-end feature extraction algorithm

This is just the way to compute feature vectors from speech waveforms. This is; when we get the file as a whole we must ask ourselves how we are going to work with this. The problem is that it is difficult and senseless trying to characterize all in once as it is a non-stationary signal; we must try to keep the relevant information and eliminate the redundancies.

The sound signal presents a variation along time which is slower than the analysis frequency used in the speech process systems. Therefore we can consider that when we take really small pieces of this stream, these are close to the stationariness. Besides, it is common to work in the frequency domain and the parameters used to describe the signal are the **Cepstrals** which have been proved as the most optimum.

All this process it is what I will try to explain. This diagram gives us a general idea of the process:

![Diagram of the Front-End algorithm from the ETSI ES 201 108 V1.1.3 (2003-09)](image)

2.1.1 Offset compensation

Prior to the framing, a notch filtering operation is applied to the digital samples of the input speech signal (SIN) to remove their DC offset, producing the offset-free input signal.

DC offset is an offsetting of a signal from zero. The term originated in electronics, where it refers to a direct current voltage, but the concept has been extended to any representation of a waveform. DC offset is the mean amplitude of the waveform; if the mean amplitude is zero, there is no DC offset.
DC offset is usually undesirable. In audio processing, a sound that has DC offset will not be at its loudest possible volume when normalized.

Along this document I will add the mathematical expressions used in my matlab implementation. In this case we can eliminate this offset by applying the following expression:

\[ S_{of}(n) = \sin(n) - \sin(n - 1) + 0.999 \times \text{sof}(n - 1) \]

### 2.1.2 Framing

Once we have got rid of the DC Offset we can start by framing the signal; this is we start cutting the signal off in small pieces as we introduced. When we talk about speech or in other words, a signal where the sound in which we are interested will have 4 kHz as the highest frequency, the proper size of the windows is the following:

<table>
<thead>
<tr>
<th>Sampling rate (kHz)</th>
<th>( f_{s3} = 16 )</th>
<th>( f_{s2} = 11 )</th>
<th>( f_{s1} = 8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame length ( N ) (samples)</td>
<td>400</td>
<td>256</td>
<td>200</td>
</tr>
<tr>
<td>Shift interval ( M ) (samples)</td>
<td>160</td>
<td>110</td>
<td>80</td>
</tr>
</tbody>
</table>

*Values of Frame Length and Frame Shift Interval Depending on the Sampling Rate*

This means for examples that for the (most common) case of 8 kHz we take frames of 25 ms in terms of time and we shift the windows 10 ms. This settings give us let us reach the optimum levels of stationariness.

Here we just must be aware that we will probably exceed the number of samples in the last window and we must 'add' some samples to reach the 200 (in my case just with the '0' value). Here is an idea of loop and it is control:

```matlab
out=0;
lim=length(Sof); %Soff is the vector just after the DCOffset compensation
i=1;
while out==0;
    if (i+N-1)>=lim; %We need it to control out of bounds at the end
        R=(i+N-1)-lim
        X=(Sof(i:lim));
        X=[X zeros(1,R)]
        out=1
    end
...
end
```
2.1.3 Energy measure

The logarithmic frame energy measure (logE) is computed after the offset compensation filtering and framing for each frame but in parallel to the rest of the process. We need this value as an extra coefficient which is not really useful till further on the process:

$$\text{logE} = \ln \left( \sum_{i=1}^{N} s_{of}(i)^2 \right)$$

Here \( N \) is the frame length and \( sof \) is the offset-free input signal.

2.1.4 Pre-emphasis and Windowing

The pre-emphasis filter is meant to compensate the lack of energy of the high components in comparison with the low ones; we can achieve that by processing our signal like that:

$$\text{Spe}(n) = \text{Sof}(n) - 0.97 \times \text{Sof}(n - 1)$$

Apart from that we also apply a 'Hamming window'. An amplitude weighting of the time signal used with gated continuous signals to give them a slow onset and cut-off in order to reduce the generation of side lobes in their frequency spectrum.

And in the Matlab implementation it is done following the ETSI standard too:

$$s_w(n) = \left\{ 0.54 - 0.46 \cos \left( \frac{2\pi(n-1)}{N-1} \right) \right\} \times s_{pe}(n), \ 1 \leq n \leq N$$

Here \( N \) is the frame length and \( Spe \) and \( Sw \) are the input and output of the windowing block, respectively.
2.1.5 FFT

Each frame of N samples is zero padded to form an extended frame of 256 samples for 8 kHz sampling rate, and 512 samples for 16 kHz. An FFT of length 256 or 512, respectively, is applied to compute the magnitude spectrum of the signal.

```matlab
if SR==8000
    bin(:,j)=abs((fft(Sw,256)));
end
if SR==16000
    bin(:,j)=abs((fft(Sw,512)));
end
```

Now we have our frame in the spectral domain.

2.1.6 Mel Filter bank

This filter-bank is based on the knowledge of the human auditory system and its sensibility. The energy at low frequencies is normally higher. By means of this filter we focus mostly on low frequency components, which is besides where more of the noises are and we compact higher frequencies. This concept turns on the Mel scale which results from dividing the spectrum in little banks narrow and linearly spaced at low frequencies and wider and logarithmically spaced at high frequencies. This way we give more importance to the information contained at low frequencies according to human auditory system and recovering better the phonetic characteristics.

We have used the triangular filters which has its highest level at the mid-point and its extremes are in the middle of the adjacents.

![Mel Filter Bank](image)

The calculation of the energy in every band is the sum, for the indexes of the FFT in every triangle, of the products between the FFT values and the value from the triangle corresponding to the position of this index; in other words, the output of the mel filter is the weighted sum of the FFT magnitude spectrum values in each band.
2.1.7 Non-linear transformation

After the mel filter bank, as it has 23 triangular overlaped filters, 23 values to which we apply a logarithm function (natural logarithm).

\[ f_i = \ln(f_{bank_i}), i = 1, \ldots, 23 \]

The same flooring is applied as in the case of energy calculation, that is, the log filter bank outputs cannot be smaller than -50.

```matlab
function fbank=MEL(bin,SR)
    %First we must set the starting and ending point of every channel
    if SR==8000
        mel = mel_filter (SR, 23, 256); %We'll find the matrix for the filterbank
        FFTL=256/2;
    else
        mel = mel_filter (SR, 23, 512);
        FFTL=512/2;
    end
    for i= 1:23
        fbank(i) = mel(i,:)*bin(1:FFTL,1);
    end

function mel = mel_filter (fm, num_chan, nfft)
    df = fm/nfft;
    Nmax = nfft/2;
    fmax = fm/2;
    MELmax = freq2mel(fmax);
    MELinc = MELmax / (num_chan + 1);
    MELcenters = (1:num_chan) .* MELinc;
    Fcenters = mel2freq(MELcenters);
    f1 = [0 , Fcenters(1:(num_chan-1))]; %freq. inicials de cada filtre
    f2 = [Fcenters(2:num_chan), fmax]; %freq. finals de cada filtre
    BW = f2 - f1; %amples de banda de cada filtre
    %quantize into FFT indices
    indexcenter = round(Fcenters ./df);
    %compute resulting frequencies
    FFTfreq = indexcenter.*df;
    istart = [1 , indexcenter(1:num_chan-1)]; %index d'inici de les finestres
    istop = [indexcenter(2:num_chan), Nmax]; %index dels finals de les finestres
    idxbw = (istop - istart)+1; %nombre d'index a cada finestra
    %compute bandwidth [Hz]
    FFTBandwidth = idxbw.*df;
    diff = Fcenters - FFTfreq; %error de quantització
    %generem la matriu de valors i creem les finestres trianulars
    mel = zeros(num_chan,Nmax);
    for j = 1:num_chan
        %trampa esquerra
        increment = 1.0/((indexcenter(j) - istart(j));
        for i = istart(j):indexcenter(j)
            mel(j,i) = (i - istart(j))*increment;
        end
        %trampa dreta
        decrement = 1.0/(istop(j) - indexcenter(j));
        for i = indexcenter(j):istop(j)
            mel(j,i) = 1.0 - ((i - indexcenter(j))*decrement);
        end
    end
end
```
2.1.8 Cepstral coefficients

Finally we use the DCT (Discrete Cosinus Transformation). It is important to know why we make use of this transformation; as we have seen we changed from the time domain to the frequency domain with the FFT. In this case, we make use of this transformation as it helps to get more non-correlated coefficients and a reduced dimensionality (less computational burden)

\[ C_i = \sum_{j=1}^{23} f_j \times \cos \left( \frac{\pi \times i}{23} \left( j - 0,5 \right) \right), \quad 0 \leq i \leq 12 \]

A discrete cosine transform (DCT) expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering as it is the case of lossy compression of audio and images (where small high-frequency components can be discarded). particular, a DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. Like the DFT, a DCT operates on a function at a finite number of discrete data points. It can be thought of as implicitly defining an extension of that function outside the domain. That is, once you write a function \( f(x) \) as a sum of sinusoids, you can evaluate that sum at any \( x \), even for \( x \) where the original \( f(x) \) was not specified. A DCT, like a cosine transform, implies an even extension of the original function.

This gives us 12 cepstral coefficients as the most significant coefficients of the DCT.

However, I must say we have worked with 13 cepstral coefficients; this previous explanation is done according to the ETSI standard.

In the following pages there is a general idea of the main script function to make all this process. Here it is possible to see an overview of the main script function with the call to these other functions we have seen along the breakdown.
%% Introduction to the speech recognition %%
% We read the file with we want to work
[Sin,SR] = wavread('p109.wav');

% Once we have the vector with the samples we start with the Front-End
% Algorithm strictly speaking:
% DC Offset compensation (removal of the DC offset due to A/D conversion)
Sof=DCOff(Sin);

% Now we'll need to cut off the sample to have an optimum (almost
% stationary) stream. Obviously we'll work on the frames in this loop too.
% The "framing" depends on the SF
if SR==8000,
    N=200; %Frame length
    M=80;  %Shift interval
end
if SR==16000,
    N=400;
    M=160;
end
lim=length(Sof);
i=1;
j=1;
out=0;
ilog=1; % This is the index for the energy vector
Y=0;    % I'll use it to store the value of the last previous frame sample
Z=0;

while out==0;
    if (i+N-1)>=lim; %We need it to control out of bounds at the end
        R=(i+N-1)-lim
        X=(Sof(i:lim));
        X=[X zeros(1,R)]
        out=1
    else
        X=Sof(i:i+(N-1));
    end

    %We start by the energy measure as the "0th coefficient"
    logE(log)=energie(X);
    ilog=ilog+1;

    %Pre-emphasis filter to compensate the irregular energy distribution
    % (high-freq have less than low-freq)
    Spe=preemphasis(X,Y);
    Y=X(end);
%Now we will use a Hamming windows which makes a ponderation of each frame; it gives more weight to the central values (as we overlap them).

Sw=windowing(Spe,N);

%Change to the spectral domain

if SR==8000
    bin(:,j)=abs((fft(Sw,256))); end
if SR==16000
    bin(:,j)=abs((fft(Sw,512))); end

%The MEL Filtering will let us treat the spectrum according to the human auditory system
fbank(:,j)=MEL(bin(:,j),SR);
fbank(:,j)=log(fbank(:,j));

j=j+1;
i=i+M; % Window shift of M samples.
end

DCTMat=init_dctmatrix(13,23);
C=fbank'*DCTMat;

Main script for the front-end feature algorithm
2.2 Speech boundaries

Before focusing strictly in the training process there is one task to be done in order to help to reduce unnecessary processing at front end of ASR (Automatic Speech Recognition) systems. This are the words boundaries. This algorithm should let us remove gaps or silences between words in order to reduce part of processing.

Usually with a basic algorithm we make some assumptions:

- In order to compute reference level it is necessary to record piece of speechless signal. This enables algorithm to calculate power reference level.
- In order to sustain reference level the sound pressure should not fluctuate. Changes of sound intensity may influence on accuracy of recognition.

As the purpose by now is just to get rid of the useless speech part (as we can only introduce troubles due to noise) it is just a matter of having a look at the energy content. This can became a tougher task if there is a big presence of noise.

We can graphically see an approach with the signal functions of the spoken word “hallo”:
Then we can have a look at the energy cutting off with windows:

With this function we could already set more or less a preliminary boundaries but we can still make it logarithmic to see it clearer:

And with this one already set an energy boundary which will let us set temporal boundaries.
2.3 The statistical approach

Almost all present-day speech recognition systems are based on hidden Markov models (HMMs). Although the fundamentals of HMM have been understood for several decades, there has been steady progress in refining the technology both in terms of reducing the impact of the inherent assumptions, and in adapting the models for specific applications and environments.

To understand not just specifically what I have been working on but everything that has to do with speech recognition it is inevitable to avoid the concept of HMMs.

A classical problem in physics and engineering is that of getting a model for a given real-world process. Modeling techniques are useful tools in prediction, recognition or identification tasks. The application of models to signals is important for a number of reasons. First, a signal model can help to process that signal (e.g. to clean a noisy signal). Also, a model can help the understanding of the signal source and the signal generation process. In particular, statistical models have been successfully applied to speech recognition because they allow setting the recognition problem in statistical terms. Let us say, as it is our case, that:

\[ W = \{ W_i \} \]

is the set of possible numbers of a given language and we wish to obtain the number 'W(X)' corresponding to an acoustic evidence 'X'. We could obtain the recognized sentence as:

\[ W(X) = \arg \max_j P ( W_j | X ) \]

which can be explained as the maximum probability to recognize a number among all numbers, given the acoustic evidence.

The computation of the probabilities 'P ( W_j | X )' can be decomposed as:

\[ P(W|X) = \frac{P(X|W_j)P(W_j)}{P(X)} \]

And as the 'P(X)' does not depend on 'X' (it is an acoustic parameter actually) we can affirm that:

\[ W(X) = \arg \max_j \left[ P(X | W_j) P(W_j) \right] \]

The number which matches with the acoustic evidence will be the maximum value of the product between the probability of getting an acoustic evidence given a number and the probability or occurrence of this number.

For us, the conditional probability 'P( X | W_j )' is given by the acoustic model (that we will characterize with the HMMs) and 'P(W)' by the language model.

A very graphical interpretation of the process would be this diagram:
There are different recognition techniques depending on whether we try to recognize isolated words, or continuous speech, if we need the system to be dependent on the speaker or independent, or maybe we need to train large vocabularies or just a few words. What it is for sure is that with the current systems we will make sure of statistical models.

It is important to know before studying thoroughly the investigation how to obtain this acoustic patterns through the hidden Markov models.
2.4 Modeling speech recognition: HMMs

One possible reason why HMMs are used in speech recognition is that a speech signal could be viewed as a piecewise stationary signal or a short-time stationary signal. Speech could thus be thought of as a Markov model for many stochastic processes.

Another reason why HMMs are popular is because they can be trained automatically and are simple and computationally feasible to use.

The hidden Markov model will tend to have in each state a statistical distribution that is a mixture of diagonal covariance Gaussians which will give a likelihood for each observed vector. Each word will have a different output distribution; a hidden Markov model for a sequence of words or phonemes is made by concatenating the individual trained hidden Markov models for the separate words.

Modern speech recognition systems use various combinations of a number of standard techniques in order to improve results over the basic approach described above. A typical large-vocabulary system would need context dependency for the words (so words with different left and right context have different realizations as HMM states); it would use cepstral normalization to normalize for different speaker and recording conditions. The features could have the so-called delta and delta-delta coefficients to capture speech dynamics.

Decoding of the speech would probably use the Viterbi algorithm to find the best path, and here there is a choice between dynamically creating a combination hidden Markov model which includes both the acoustic and language model information, or combining it statically beforehand.

Let us assume a process described by a set of $N$ states $\{s_1, s_2, \ldots, s_N\}$.

Each state represents a certain event or observation. The system changes from one state to another (transition) in each time interval. We will call $q_t$ the state at time $t$.

The Markov processes (or chains) are characterized by the dependence of the current state with respect to previous
states. This process is fully described by the transition probabilities from one state to another:

\[ a_{ij} = P(q_t = s_j | q_{t-1} = s_i) \]

So, the probability of transition from one state to another depends on the probability of changing to this new state \((s_j)\) given that the previous (on time) state has occurred.

Here there is an example to see how the dynamics of such types of Markov processes work.

Supposing a model for weather forecast with only three possible states:

\(\text{(observations): rainy (s1), cloudy (s2) and sunny (s3)}\)

This model allows us to calculate the probability of observing any sequence of weather states. For example, we can calculate the probability of the sequence:

“sunny/sunny/rainy/cloudy”

\((O = (s3, s3, s1, s2))\)

as:

\[ P(O|\text{model}) = P(s_3 s_3 s_1 s_2 |\text{model}) = P(s_3)P(s_3|s_3)P(s_1|s_3)P(s_2|s_1) = P(s_3)a_{33}a_{31}a_{12} \]

To do our task we have been working with utterances of numbers, thus we end up creating individual models for each number that usually will contain 3-4 states per phoneme; to have an idea of how many of them we need for the English utterances we know that:

\[ 1-/\text{w}\text{An}/\, 2-/\text{tu}:/\, 3-/\text{θri:}/\, 4-/\text{f}\text{ær}/\, 5-/\text{f}\text{aɪv}/\, 6-/\text{stɪks}/\, 7-/\text{ˈsevən}/\, 8-/\text{eɪt}/\, 9-/\text{n}\text{ain}/\, 0-/\text{ˈz}\text{ɪərə}/\, 0-/\text{ə}/\]

During my experiments for example I have been working with models of around 18 states per number (with the entering and exit states which doesn't contain acoustic information)

When we say one number after another, the beginning of the new number can vary enough as to not trust a model just based on the individual training.

In the case of the speech recognition we will normally deal with left-to-right transitions between the states as this is the way our speech behaves.

![Left-to-right topology](image-url)
I would also like to give a closer idea of the steps to follow; however I will expose the overview of the methodology enough to understand a development that can be found in the referred resources.

Every state must contain the parameters that should describe the spectral and energy characteristics of the segment we have trained it with. As we have seen initially, we cut off the signal in the way that we obtain new cepstral coefficients every 10 ms of shifting along the signal. This means that we have many vectors with similar information and moreover, from different utterances. For that reason, **we usually assign the mean of some different feature vectors to each state**. Furthermore, if the system must be independent from the speaker we will have to give information to the state of the **distribution of each acoustic parameter**.

Commonly we will train the systems with **several utterances** since in theory it should give us a more representative recognition; this will give us the **information necessary to set the probabilities between the states** and it will also help us to decide how many feature vectors (how long it is the segment) assigned to each state.

When we refer to speaker independent systems this **utterances** will be from different speakers. This plus of information (distribution of the acoustics parameters) comes from the fact that the pronunciation of a speech segment varies with different speakers. In this cases we **model the spectral and energy characteristics with a distribution function**. Each parameter can be described by an individual one dimensional distribution function (usually a Gaussian one) which is estimated during the whole training phase thanks to the statistical information of all the utterances.

As a summary:

- The mean vector calculated over all L feature vectors that are mapped on an individual state
  \[
  \hat{m}(S_i) = \frac{1}{L} \sum_{j=1}^{L} \tilde{X}(j)
  \]

- Occurrence of the acoustic features in each segment (=state) by a multivariate Gaussian distribution
  \[
  p(\tilde{X}[S_i]) = \frac{1}{\sqrt{(2 \cdot \pi)^{M} \cdot |\Sigma|}} e^{-\frac{1}{2}(\tilde{x}-\hat{m})^T \Sigma^{-1}(\tilde{x}-\hat{m})}
  \]

When we talk about independent acoustic features (which is what we try to obtain with the front end algorithm) the occurrence of each acoustic feature can be described by an independent Gaussian
distribution which is defined by a mean and a variance

\[ p(x_{\mu \mid S_i}) = \frac{1}{\sqrt{2\cdot\pi\cdot\sigma_{x_{\mu}}^2}} \cdot e^{-\frac{(x_{\mu}-m_{x_{\mu}})^2}{2\sigma_{x_{\mu}}^2}} \]

- The covariance matrix:

\[ \Sigma(S_i) = \frac{1}{L} \cdot \sum_{j=1}^{L} (\bar{X}(j) - \bar{m}(S_i)) \cdot (\bar{X}(j) - \bar{m}(S_i))^T \]

Once we have the model it is possible to undo the steps and go back to the frequency domain and visualize the spectrum of the HMM (thanks to the energy coefficient we can recover it). This is the most visual way to see which is the shape that should approach the input signal in the recognition to match this model. This the example of the number “seven”
2.5 Adaptation of HMMs

The adaptation of the HMM's as a way to improve the performance of the recognition system has come out to be very powerful. Here the reader will just have some strokes of the topic; it is extensive and conceptually dense as to expose it deeply. Very detailed information about the methods of professor Hirsch can be found in the “Automatic Speech Recognition in Adverse Acoustic Conditions” section of the book “Advances in Digital Speech Transmission” referred.

The idea it is to simulate acoustic characteristics when creating the patterns as to make it more suitable for the environments where we make the recognition process.

It is possible for example to analyse the modifications that causes the reverberation of the room or place where it is being done the recording. Based on the results it is presented a method to adapt the static parameters of HMMs to the reverberant signals of a hands-free speech input.

We can also modify HMMs that have been created from a noiseless database or recording environment and adapt them in a way we could simulate a noisy one.

There are also some procedures capable to deal with the acoustic characteristics of a phoneme in the context of a specified preceding and specified succeeding phoneme.

This would be a graphical idea of where we would attack the conventional system to adapt some distortion effects.

\[ Scheme \text{ for adapting HMMs to distortion effects } \]
3. Garbage recognition process

Overview

The aim of this process is to create a garbage model which should allow us determining when we have an insertion, so to speak, when some parts of the speech that are originally silence become a word for the recognizer due to the noise influence over the speech signal. The scheme of the following figure represents a visual idea of the process to apply:

Scheme of the process for the insertion detection
3.1 Database and cepstral analysis

It all starts with the library Aurora-5 which is meant mainly to investigate the influence on the performance of automatic speech recognition for a hands-free speech input in noisy room environments. The database (which may either be used to measure front-end feature extraction algorithms, using a defined HMM recognition back-end, or complete recognition systems) is created from the source speech Tidigits which consists of connected digits task spoken by American English talkers (downsampled to 8kHz). A selection of 8 different real-world noises have been added to the speech over a range of signal to noise ratios with controlled filtering of the speech and noise.

I have been working with two of these artificially created atmospheres:

- Car Noise
- Interior Noise

Besides, we can find different resultant qualities:

- SNR – 0dB, 5dB, 10dB, 15dB, 20dB

However, the process to follow is the same independently of the case.

We have initially this individual library (one of the exposed options) with 8700 samples. We can find numbers individually spoken by men and women. The files are presented as a plain text with the “written” information (basic PCM to raw) of the sound wave all of them with a “.raw” extension so if we try to open it with any text processor, we won't be able to see anything but unintelligible symbols.

First step then is getting this features from the signal; we have to do it through the cepstral analysis:

```bash
#!/bin/csh
# script to run own feature extraction on training data

set exe = /packages/speech_recognition/simul/rec_sim_hgh_ada_all
set options = "-m analyze -f 8000 -c "
set list_file = /data/home/pablo/Training_Recognition_Scripts/Lists/InteriorNoise00_raw_list
set pat_dir = /data/home/pablo/Training_Recognition_Scripts/anal_pattern_InteriorNoise00

if (-d $pat_dir) then
    echo "WARNING: Directory "$pat_dir" does already exist!"
    #exit
else
    mkdir $pat_dir
endif

$exe $options -i $list_file -o $pat_dir
```

shell script interface to get the feature vectors; “anal_data_extest”
This is kind of “interface” script which lets us do this process. Obviously here there is not the analysis function itself but the call to it at the end; we have already explained what is the point of it in the front-end feature extraction algorithm section (and we can find there the script detailed explained too). We just have to specify where the files to be analysed are with the whole route and this is done through:

```
set list_file  = /data/home/pablo/Training_Recognition_Scripts/Lists/InteriorNoise00_raw_list
```

This is a text file which contains the list of the route of every single speech sample:

```
... 
data/aurora5/speech/test/InteriorNoise00/woman/jd/zz.rjd.raw 
data/aurora5/speech/test/InteriorNoise00/man/jh/zz.rjh.raw 
data/aurora5/speech/test/InteriorNoise00/woman/jw/zz.rjw.raw 
data/aurora5/speech/test/InteriorNoise00/woman/lj/zz.rlj.raw 
data/aurora5/speech/test/InteriorNoise00/man/dj/zz9139.rdj.raw 
data/aurora5/speech/test/InteriorNoise00/woman/ap/zz96.rap.raw 
...
```

Fragment of the file “InteriorNoise00_raw_list”

Now what its new and important to know for us is what we get as a file result. This is a pattern output; another file which we label with a ‘.pat’ extension and which keeps the speech features from each previous ‘.raw’ file. This means that we get 8700 new files with the same name but different extension stored in:

```
set pat_dir    = /data/home/pablo/Training_Recognition_Scripts/anal_pattern_InteriorNoise00
```

In our case, using the HGH analysis scheme, the information that we obtain once we have run this process are both the “static parameters” and the “dynamic parameters” this is, the 12 coefficients from the cepstral analysis, the logarithmic frame energy value and the delta and delta-delta coefficients for the contour information. In the end we have 39 coefficients which give us the necessary information for the training and recognition step. In case of adaptation we need one more coefficient (0th coefficient).

With this files it is posible to make the training. Here, we start making use of the HTK (HMM Tool Kit), a tool for building Hidden Markov Models (HMMs) and which combined with the HGH (Hans Günter Hirsch) can help us to get new patterns/models to set our recognition systems.

As we said our goal is to create a garbage model, to be precise a model capable to detect the appearance of non-stationary noises hidden in our speech. In my case I already had the models of the words to be recognized so it is some processing time I saved. We need obviously a good model to recognize all numbers; our models are trained in base to the database “Tidigits” with no presence
of noise and we had to recognize spoken '1' to '9' plus 'oh' and 'zero'. I had already stored on the server so I just had to link them in my hmm's list file as we will see. Otherwise we should have done the process to train the model for each of these numbers. However, it is quite understandable how to do it once I will have exposed the specific case of the garbage one.

At this point we have the feature vectors (.pat files), the HMMs of the numbers and, very important, the labels of the pattern files. This is just a file text with the time information of the “timing”. We have the time when each number is originally spoken in every sound file; a part from these numbers, there is also the term $w_{sil}$ for the fragments without voice (usually beginning and ending of the number sequence). This files are going to be essential to achieve our goal. They look like that:

```
0 2100000 w_sil
2100000 6500000 w_9
6500000 10300000 w_4
10300000 14900000 w_o
14900000 18700000 w_3
18700000 22400000 w_o
22400000 26400000 w_4
26400000 33000000 w_5
33000000 33100000 w_sil
```

**text file 94o3o45.rml.lab**

This is the file we are going to use as a reference; when we train a system through HTK tool, this needs this files to know where are the bounds of each word/number in order to associate this part of the speech to the training of this word/number models.

From this point and so on starts the modification over the regular process to try to prove our hypothesis. We have to make the regular recognition with the noisy data base (Aurora-5) in order to get new labels. The recognition process gives us exactly the same idea of label with the recognized results; this way we have a clear idea of which are the **insertions** (recognizes a number/word when there is not voice), **substitutions** (confusion of numbers/words) or **deletions** (it does not find a number/word that matches and omits it).

### 3.2 Previous recognition

Here I would like to highlight the difference of processing time depending on which code we use; Matlab does not become a good option as it is a long task itself. If it is not precise to use the matlab software I would recommend running specially this process through shell (C schell – csh) in this
case. This will speed up considerably the job.

We will have a closer look to the recognition interface we have mainly used; we will see later that we will need (in any case we have used it) Matlab tool during this process to make some particular modifications.

```bash
#!/bin/csh
# script to run
# feature extraction & recognition
set exe = /packages/speech_recognition/simul/rec_sim_hgh_ada_all
set options = "-m anal_recog -f 8000"
set list_file = /data/home/pablo/Training_Recognition_Scripts/Lists/InteriorNoise00_raw_list
set hmm_list = /data/home/pablo/Training_Recognition_Scripts/Lists/list_hgh_39coef_ginst_IN10_ending.hmm
set res_dir = /data/home/pablo/Training_Recognition_Scripts/InteriorNoise_sh_rec_39coeff_ginst_ending_synHIRSCH
set syn_file = /data/home/pablo/Training_Recognition_Scripts/ti_gender_garbage_mod.syn
if (-d $res_dir) then
    echo "WARNING: Directory "$res_dir" does already exist!"
    exit
else
    mkdir $res_dir
endif
$exe $options -i $list_file -o $res_dir -h $hmm_list -s $syn_file
```

**shell script recognition interface: “recog_data_sh_IN00_ending”**

We saw in detail how this process works.

As to compare and look for the matches we need to have our input files as feature vectors we need to set some options needed for that:

```bash
set options
We will need to specify a list file which contains the route to all the .raw files that we would like to recognize.

set list_file
```

This is the list we had referred before where we will set the route to the hmm's.

As we can clearly see, the models are for all numbers / both genders and silence.

```bash
set hmm_list
```

We have to tell where to put the output with text files named with the extension .res and which are nothing
else but the label files with the recognition results.

```xml
<NUMNODES> 3
<SUBCLASS> DIGITS_MALE 11 w_1_male w_2_male w_3_male w_4_male w_5_male w_6_male w_7_male w_8_male w_9_male w_o_male w_z_male
<SUBCLASS> DIGITS_FEMALE 11 w_1_female w_2_female w_3_female w_4_female w_5_female w_6_female w_7_female w_8_female w_9_female w_o_female w_z_female

<NODE> 1
  entries: w_sil
  entry-nodes: 1

<NODE> 2
  entries: w_sil DIGITS_FEMALE w_sil
  entry-nodes: 1 2 2

<NODE> 3
  entries: w_sil DIGITS_MALE w_sil
  entry-nodes: 1 3 3

<NUMENDNODES> 2
<ENDNODES> 2 3

<SYNONYMS> 11
  w_1 2 w_1_male w_1_female
  w_2 2 w_2_male w_2_female
  w_3 2 w_3_male w_3_female
  w_4 2 w_4_male w_4_female
  w_5 2 w_5_male w_5_female
  w_6 2 w_6_male w_6_female
  w_7 2 w_7_male w_7_female
  w_8 2 w_8_male w_8_female
  w_9 2 w_9_male w_9_female
  w_z 2 w_z_male w_z_female
  w_o 2 w_o_male w_o_female

<PENALTIES> 30 0 10

<TIMEOUT1> 100
<TIMEOUT2> 125
```

synthesis file for the recognition process
With this file we describe overall the possible paths of the recognition. For examples, if we knew for sure that we are gonna recognize some numbers and after a precise word we will just recognize some letters, we could define something like the following. In this case, as it was referring to the initial recognition trying to recognize number for women and men we include all possibilities (numbers) under the names DIGITS_FEMALE and DIGITS_MALE. The spoken samples are series of numbers that is why we use a synthesis like that:

![Illustración 1: synthesis file 3 states](image)

When we will have the garbage model we will have to make some modifications and try to find the most suitable synthesis file for our purpose.

### 3.3 Labels modification

Now we are ready to get our first results; by now we will not talk about performances as the aim here is to explain the procedure and the files we have to deal with but somehow the results we are obtaining now are the reference to improve. What I will focus on are the new labels that I will have to modify. The best way to see what I am doing is with a particular case:

One of the files – **o384899.rhm.lab** – we have recognized had this original labeling

<table>
<thead>
<tr>
<th>Time (ms)</th>
<th>Value</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2000000</td>
<td>w_sil</td>
</tr>
<tr>
<td>2000000</td>
<td>6700000</td>
<td>w_o</td>
</tr>
<tr>
<td>6700000</td>
<td>7600000</td>
<td>w_sil</td>
</tr>
<tr>
<td>7600000</td>
<td>12300000</td>
<td>w_3</td>
</tr>
<tr>
<td>12300000</td>
<td>12400000</td>
<td>w_sil</td>
</tr>
<tr>
<td>12400000</td>
<td>18700000</td>
<td>w_8</td>
</tr>
<tr>
<td>18700000</td>
<td>18800000</td>
<td>w_sil</td>
</tr>
<tr>
<td>18800000</td>
<td>23000000</td>
<td>w_4</td>
</tr>
</tbody>
</table>
When we make the recognition we get a “rec” file which looks like the following:

```
<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2375000</td>
<td>w_sil</td>
</tr>
<tr>
<td>2375000</td>
<td>5675000</td>
<td>w_4</td>
</tr>
<tr>
<td>5675000</td>
<td>8375000</td>
<td>w_sil</td>
</tr>
<tr>
<td>8375000</td>
<td>1187500</td>
<td>w_3</td>
</tr>
<tr>
<td>1187500</td>
<td>1267500</td>
<td>w_sil</td>
</tr>
<tr>
<td>1267500</td>
<td>1617500</td>
<td>w_8</td>
</tr>
<tr>
<td>1617500</td>
<td>1937500</td>
<td>w_sil</td>
</tr>
<tr>
<td>1937500</td>
<td>2347500</td>
<td>w_4</td>
</tr>
<tr>
<td>2347500</td>
<td>2722500</td>
<td>w_3</td>
</tr>
<tr>
<td>2727500</td>
<td>2927500</td>
<td>w_sil</td>
</tr>
<tr>
<td>2927500</td>
<td>3327500</td>
<td>w_9</td>
</tr>
<tr>
<td>3327500</td>
<td>3417500</td>
<td>w_sil</td>
</tr>
<tr>
<td>3417500</td>
<td>3737500</td>
<td>w_9</td>
</tr>
<tr>
<td>3737500</td>
<td>3927500</td>
<td>w_8</td>
</tr>
<tr>
<td>3977500</td>
<td>4247500</td>
<td>w_sil</td>
</tr>
</tbody>
</table>
```

This output example is the result of a system which makes profit of the adaptation system. The performance rates with the database used are around 90% of accuracy so this one of the few files with insertions but also some substitutions (although is not what we are working with).

Here we can see how there is a first substitution:

```
<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000000</td>
<td>6700000</td>
<td>w_o</td>
</tr>
<tr>
<td>2375000</td>
<td>5675000</td>
<td>w_4</td>
</tr>
</tbody>
</table>
```

This is a substitution since the original value should be 'o' (“oh”) and the recognizer says it is a '4'. We could think it is an insertion (the recognizer says there is a number where originally there was just silence -due to noise-) but if we had a look at the timing of the original labelling we can appreciate that this recognition is made during the time of the 'o' and not the silence.

Secondly there is another substitution caused by the noise addition to the '8':

```
<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>2300000</td>
<td>2790000</td>
<td>w_8</td>
</tr>
</tbody>
</table>
```
And finally we have the case we are really interested in; the insertion of a value which was not
spoken but that due to an unexpected non-stationary noise has made mistaking our recognition
system:

| 33900000 | 40900000 | w_9 |
| 40900000 | 41000000 | w_sil |

| 18075000 | 39775000 | w_o |

As we see, the insertion starts appearing when are already saying the number 9. This was a common
situation so in many cases the insertion was mainly due to noise just over silence but also tanking
some part of voice too.

Now that we know which was the difference, the idea was clear:
We must train a new model which should be able to recognize this insertions and gain some
accuracy in our system performance. We have a long enough database as to trust the following:

3.4 Garbage model training

What if we try to make a model based on all this small periods of time when some non-stationary
noises which make our system confuse the noise, present most of the times while speaker's silence,
with numbers as we will be able to detect as what it is: an insertion. We must be careful to take just
exactly to take the part of speech when this happens; otherwise, we take the risk to increase the
possibilities to over-exploit this model when recognizing (above all with samples with low SNR).
We made a first approach by changing radically the labels in the way it was taking more than just
the exact part where there had been the insertion and the results were not very optimal.

Once we had a clear idea that the patterns had to be trained with the major precision possible we
could develop a “final” script to modify all this labels of this files that had not been recognized
properly and where there were insertions. Otherwise it was not of our interest.

It is possible to see the code attached by the end of the report. The problems came mainly from the
fact I decided to work with matlab software for a task that had to do with reading of files and
precise work with string types; besides I had to make multiple modifications as the length and way
the files were shown were variable.
First I would just like to show, keeping on with the previous example, how looks like the file modified. As I said, the performance of the system's example and the database make possible very optimal results and we will see how we just set the value “g_inst” (our tag for the new garbage training) in the only insertion seen. With worse condition, as it is logical, the insertions were more common.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2375000</td>
<td>w_sil</td>
</tr>
<tr>
<td>2375000</td>
<td>5675000</td>
<td>w_4</td>
</tr>
<tr>
<td>5675000</td>
<td>8375000</td>
<td>w_sil</td>
</tr>
<tr>
<td>8375000</td>
<td>11875000</td>
<td>w_3</td>
</tr>
<tr>
<td>11875000</td>
<td>12675000</td>
<td>w_sil</td>
</tr>
<tr>
<td>12675000</td>
<td>16175000</td>
<td>w_8</td>
</tr>
<tr>
<td>16175000</td>
<td>19375000</td>
<td>w_sil</td>
</tr>
<tr>
<td>19375000</td>
<td>23475000</td>
<td>w_4</td>
</tr>
<tr>
<td>23475000</td>
<td>27275000</td>
<td>w_3</td>
</tr>
<tr>
<td>27275000</td>
<td>29275000</td>
<td>w_sil</td>
</tr>
<tr>
<td>29275000</td>
<td>33275000</td>
<td>w_9</td>
</tr>
<tr>
<td>33275000</td>
<td>34175000</td>
<td>w_sil</td>
</tr>
<tr>
<td>34175000</td>
<td>37375000</td>
<td>w_9</td>
</tr>
<tr>
<td>37375000</td>
<td>38075000</td>
<td>w_sil</td>
</tr>
<tr>
<td>38075000</td>
<td>39775000</td>
<td>g_inst</td>
</tr>
<tr>
<td>39775000</td>
<td>42475000</td>
<td>w_sil</td>
</tr>
</tbody>
</table>

*Modified LAB FILE o384899.rhm.lab*

Fortunately we can use another file (output of an evaluation script) which makes a summary of all the samples where there has been an incongruity between the original file and the recognized.

```
Basic_Results_NoGarbage_NoT60/InteriorNoise10_ML_rec_42coeff//o36o9.rbm.rec
LAB: w_o w_3 w_6 w_o w_9
REC: w_4 w_3 w_6 w_o w_9
Aligned transcription: /data/tidigits/label/o378.rrk.lab vs
Basic_Results_NoGarbage_NoT60/InteriorNoise10_ML_rec_42coeff//o378.rrk.rec
LAB: w_o w_3 w_7 w_8
REC: w_3 w_7
Aligned transcription: /data/tidigits/label/o37o2.rpt.lab vs
Basic_Results_NoGarbage_NoT60/InteriorNoise10_ML_rec_42coeff//o37o2.rpt.rec
LAB: w_o w_3 w_7 w_o w_2
REC: w_o w_3 w_7 w_o w_2 w_o
Aligned transcription: /data/tidigits/label/o384899.rhm.lab vs Basic_Results_NoGarbage_NoT60/InteriorNoise10_ML_rec_42coeff//o384899.rhm.rec
LAB: w_o w_3 w_8 w_4 w_8 w_9 w_9
REC: w_4 w_3 w_8 w_4 w_3 w_9 w_9 w_o
Aligned transcription: /data/tidigits/label/o3o47.rpj.lab vs
Basic_Results_NoGarbage_NoT60/InteriorNoise10_ML_rec_42coeff//o3o47.rpj.rec
LAB: w_o w_3 w_o w_4 w_7
REC: w_o w_3 w_o w_4
```

*Compare file (“eval_results_mod” output)*
With this file it was easier to discriminate this files where there was a insertion from the others. This is the case of the lines where you can find a blank space in the 'LAB' row and some number recognized in the 'REC' row:

LAB: w_o w_3 w_8 w_4 w_8 w_9 w_9
REC: w_4 w_3 w_8 w_4 w_3 w_9 w_9 w_o

First we decided to find all the insertions present along the utterances. This turned out to be not the best option as there were too many samples causing an excess of recognitions of this model above all with databases of low SNR rates. For this reason we decided that, as the most common problem was at the ending of the speech since the speaker stops talking till the sound recording stops, we would focus on a model trained with the ending insertions (after trying the general model and also another one with the insertions at the beginning).

Now it is time to make the new training for the garbage model. Now we will see in detail what we would have done for any of the numbers.

When we make the training we need some fields to be specified; we can see this new “interface” code for the training that helps us to interact with the HTK tool. Here we can see the inputs from the script:

```
#!/bin/csh
########## Script for training with HTK ###########
# you have to define the path to your pattern files
set pat_dir = /data/home/pablo/Training_Recognition_Scripts/Anal_Pattern/anal_pattern_InteriorNoise10
set tmp_listfile = /data/home/pablo/Training_Recognition_Scripts/Lists/InteriorNoise10_pat_ending_random
# you have to define the paths where to store the HMMs
set hmm_dir_init = /data/home/pablo/hmm_IN10_ending_random2
set hmm_dir_rest = /data/home/pablo/hmm_IN10_ending_random
# definition of the intial HMM model (42 Mel frequency cepstral coefficients / 5 States)
set prname_garbage = /data/noises/config_files/proto_42coef_5states
# files needed for training
set config_file = /packages/speech_recognition/config_files/config
set label_dir = /data/home/pablo/Training_Recognition_Scripts/Interior_Noise10_g_inst_ending_lab
```

As we see we need to precise the directory with the pattern files:

```
set pat_dir
```

In our case we needed to precise specific files from this folder; we can modify our code and attach a
text file with the routes which is what we do here:

```plaintext
set tmp_listfile =
```

We define the directory where we are going to store the HMM's:

```plaintext
set hmm_dir_init
set hmm_dir_rest
```

We also define the initial model as we need some point where to start (and also where to end); this is done through the prototype variable:

```plaintext
set prname_garbage
```

And also the settings about the architecture (way of reading):

```plaintext
set config_file
```

And finally the text file with the route to the label files of all this ones we want to use to make our training.

```plaintext
set label_dir
```

Thanks to this tool we get a model of this kind:

```
<STREAMINFO> 1 42
<VEGCSIZE> 42</NULLD><MFCC_R_D_A_0><DIAGC>
  ~h "g_inst"
<BEGINHMM>
  <NUMSTATES> 7</NUMSTATES>
  <STATE> 2</STATE>
  <NUMMIXES> 4</NUMMIXES>
  <MIXTURE> 1 2.284423e-01
  <MEAN> 42
  -6.725940e-01 -1.201969e+00 -2.066944e+00 -4.2100703e-01 -7.813931e-02
  5.407984e-01 -1.492669e-01 2.265955e-01 2.095829e-01 2.589585e-01 2.316277e+02
  -2.426690e-01 1.130311e-01 -6.944919e-03 -1.451401e-02 1.631966e-02 2.767651e-02
  5.992732e-03 1.732150e-02 1.934268e-02 8.085116e-02 -4.983671e-03 1.338820e+00
  9.601866e-02 3.525721e-02 1.995616e-02 3.709116e-02 3.614185e-02 -1.033206e-02
  2.466941e-02 -1.628267e-03 -8.890504e-03 -8.095141e-03 -1.443777e-03 9.378366e-03
  -7.005274e-01 -7.685515e-02
  <VARIANCE> 42
  4.956285e+00 1.849604e+01 1.691141e+01 8.162929e+00 7.538730e+00
  5.496907e+00 4.561700e+00 4.387714e+00 2.946002e+00 2.764671e+00
  2.275197e+00 1.844029e+00 1.970661e+00 1.299728e+00
  9.455565e-01 5.813227e-01 4.283608e-01 2.983645e-01 2.737295e-01
  2.068195e-01 1.629491e-01 1.411390e-01 1.158582e-01 1.107423e-01
  8.894034e-02 7.547728e-02 7.088944e+00 7.097957e-02
  1.533085e+00 6.809963e-02 5.214466e-02 3.830293e-02 3.631732e-02
  3.286425e-02 2.502352e-02 2.311916e-02 1.774598e-02
  1.712327e-02 1.493365e-02 1.208429e-02 4.882758e-01 4.947895e-03
  <CONST> 3.775806e+01
  <MIXTURE> 2 2.178412e-01
  <MEAN> 42
  -3.999355e+00 6.095223e-01 -2.261554e-01 -3.436410e-01 -1.292827e+00
  -4.833194e-01 -5.785282e-01 5.369911e-01 -4.548309e-01 8.100171e-02
  -1.403045e-02 5.179238e-01 2.057796e+02 1.530344e+01
  -1.469695e-01 1.882272e-02 2.097053e-02 -1.056855e-01 -4.624204e-02
  -2.466868e-02 3.075590e-02 5.274963e-03 2.773418e-02 3.645448e-02
  -4.066040e-04 -3.485920e-04 1.603179e+00 1.790244e-01
  -1.106396e-01 -5.143663e-02 -2.876591e-02 -3.904942e-02 -1.548666e-02
  7.472967e-03 2.285072e-02 8.619123e-03 4.897626e-03 1.019866e-03
  3.168216e-03 -1.167054e-02 6.645550e-01 6.998351e-02
  <VARIANCE> 42
```
Speech recognition - Testing of non-stationary noise suppression method

Pablo Ros Flores

This model will be a new possibility for the upcoming recognition. We have to add the route to its location in the list of HMMs for the recognition process.

We have seen while introducing the background which information we needed; there is a header in
the HMM which gives an overview:

<STREAMINFO> 1 42
<VECSIZE> 42<NULLD><MFCC_E_D_A_0><DIAGC> → Vector with 13 MFCC +  Energy (14th) + Delta(28) and Delta-Delta(42) coefficients.
~h "g_inst"
<BEGINHMM>
[NUMSTATES> 7 → Number of states for the model; as it is the garbage model we have enough with 5 real states which is quite few less than with the number models.
<STATE> 2
(NUMMIXES> 4 → Each state is described by the mixtures of 4 Gaussian distributions for each of the acoustic features.

At the end we have the covariance matrix which gives the probabilities to remain in one stage or change. This matrix con be useful if we want to change at some particular point the behaviour of our system once we have trained all the different models to use. As we will see, one of the decisions to improve our results was to modify this matrix (the model was very likely to be recognized and by modifying the two first parameters in the matrix we can drop the probability to start recognizing this model)
3.5 Evaluation of the system

Once we have our model it is time to run new recognition experiments to see if it is working. The procedure it is the same previously exposed in the recognition section just that we have this new model that should help us to avoid the insertions. It is time now to evaluate the new behaviour.

```bash
./eval_results
#!/bin/csh -fb
# SHELL Script to evaluate recognition results
#
if ($#argv < 1) goto usage
if ($#argv > 2) goto usage

set label_dir   = /data/tidigits/label
set lab_listfile =
/data/home/pablo/Training_Recognition_Scripts/labels_to_compare/list_tidigits_labels
set tmp_listfile = /tmp/ttt2.list                  # temporary list file
set speech_dir  = /data/aurora5/speech # Just if there's a second argument

#include HTK path
set name = `printenv PATH | grep htk`
if ($name == "") then
   setenv PATH $PATH{:/packages/htk-3.3/bin.linux}
endif

echo "* RECOGNITION RESULTS              *
ls -l $1 | awk '{printf("%s/%s\n", dir, $1)}' dir=$1 > $tmp_listfile
#evaluate recognition results
HResults -e "??" w_sil -e "??" g_inst -e "??" g_instb -e "??" . -e "??" g_breath -p -L 
$label_dir -S $tmp_listfile $lab_listfile
if ($#argv == 2) then
   set list = `HResults  -e "??" . -e "??" g_inst -e "??" g_instb -e "??" w_sil -e "??" 
g_breath -t -L $label_dir -S $tmp_listfile $lab_listfile \ 
   | awk '{if ($1 == "Aligned") {print $3}}'
set nlist = $#list
if (-e $2) then
   rm $2
   echo $2" removed!"
endif
while ($#nlist > 0)
   set name = $list[$nlist]:t:r{.raw}
   find $speech_dir -name $name >> $2
   @ nlist--
end
endif
rm $tmp_listfile
exit
```

Script for the evaluation of the results (eval_results)
To have a compact information of the results I could use a very useful script of the HTK that shows me a summary of what we have obtained. This is going to be the tool to see the accuracy of our experiments.

There is a modification of this same file which is the one we used to obtain the comparison between the labels with the real information and the labels after the recognition system. In this case we are just interested in the matrix that gives as a result with an overall information.

We need to know how to interpret this matrix:

```
------------------------------------ HTK Results Analysis ------------------------------------
Date: Wed Jul 8 16:15:31 2009
Ref: /data/tidigits/label
Rec: InteriorNoise10_sh_rec_39coeff_ginst_ending/1115z.rmi.rec
     InteriorNoise10_sh_rec_39coeff_ginst_ending/1119673.rbw.rec
     InteriorNoise10_sh_rec_39coeff_ginst_ending/111.rah.rec
     InteriorNoise10_sh_rec_39coeff_ginst_ending/111.rpf.rec
     InteriorNoise10_sh_rec_39coeff_ginst_ending/121851.rap.rec

Overall Results
SENT: %Correct=43.72 [H=3804, S=4895, N=8700]
WORD: %Correct=69.67, Acc=69.66 [H=19915, D=8245, S=423, I=4, N=28583]

Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Del</th>
<th>%C</th>
<th>%E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2954</td>
<td>9</td>
<td>5</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>515</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7</td>
<td>1269</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>1331</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1892</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>650</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>399</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>616</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>2378</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>195</td>
<td>98.9/0.1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>7</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>944</td>
<td>4</td>
<td>30</td>
<td>1</td>
<td>1613</td>
<td>94.7/0.2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1891</td>
<td>3</td>
<td>0</td>
<td>695</td>
<td>99.3/0.0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>13</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1664</td>
<td>4</td>
<td>917</td>
<td>98.1/0.1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2138</td>
<td>405</td>
</tr>
</tbody>
</table>

---

**Matrix results (output of Eval_results)**

If we want to see exactly what has happened for example with the recognition of the number 7, we must have a look at the matrix; we find in the first row the list of all the speech possibilities and once we find the column where the w_7 (it appears in vertical, do not confuse) is, we can see the number of substitutions for each word/number and delete deletions and insertions too.
And we find as well precise values of correct recognitions, accuracy, numbers of deletion, substitutions and insertions. Let us remind that:

**Deletion errors**: The system did not find a precise match

**Substitution errors**: The system mistakes and changes the original number for another one

**Insertion errors**: The system recognizes a number when there was originally lack of speech.

Besides, we can separately see statistics of the whole files (SENT) and the whole numbers independently how they were said and in which files (WORD).

The number of substitution errors (S), deletion errors (D) and insertion errors (I) can be calculated as:

\[
\text{Percent Correct} = \frac{N - D - S}{N} \times 100\% \\
\text{Percent Accuracy} = \frac{N - D - S - I}{N} \times 100\%
\]

where N is the total number of labels in the reference transcriptions. Notice that this measure ignores insertion errors. The accuracy is a more representative figure of recogniser performance.
4. Results and conclusions

The purpose here will be to show the most significant results I have obtained during the development as it has been a task of multiples that cannot be omitted if we want to avoid some errors in similar future projects.

However, I would not like to carry on without warning everyone who has to deal with the HTK and both Matlab and shell scripts for the first time, as it was my case, to run test experiments and get to know easily how it is the whole methodology. The amount of data, scripts and times that usually we will have to run the experiments can become itself a muddling task thus, it is dramatically advisable to keep files organized and familiarize with the tools before facing the challenge.

The department of electrical engineering and computer science of the university of applied sciences of Niederrhein as in some other universities and research groups, has been working on the area of speech recognition since long time ago. This has permitted obtaining new results and improvements along the years that become new challenges to beat. As we have seen, there are some standards as well as some particular systems that have reached quite high levels of recognition; for that reason every day is becomes more difficult but also more important to keep on rising this accuracy rates.

With this research, we pretended to find a way to overcome some previous experiments when working with a noisy database.

When we started with the first experiments without using the adaptation, we started by obtaining optimum results that demonstrated that the system was behaving better with the garbage model than without it. Then, when we were trying to make it work with the adaptation recognition, we have not obtained the expected results so we have tried by changing the way to “attack” the problem.

I will expose this transition with the arithmetic results. To know where were we and where we wanted to be, we needed some previous results. We had previous experiments made with the Aurora-5 database but, obviously, without the garbage model.

I have worked mainly with the HGH (firstly) and HGH Adapt (posteriori) as a reference. We can see in the following table the values for the different environments:
4.1 Previous results

<table>
<thead>
<tr>
<th>Frontend</th>
<th>Database</th>
<th>Data</th>
<th>SNR / dB</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>HGH adapt</td>
<td>Aurora 5</td>
<td>Clean</td>
<td></td>
<td>99,49%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise</td>
<td>0</td>
<td>83,28%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise Handsfree</td>
<td>5</td>
<td>95,08%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise</td>
<td>10</td>
<td>98,08%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise Handsfree</td>
<td>15</td>
<td>98,88%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise</td>
<td>5</td>
<td>87,92%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise Handsfree</td>
<td>10</td>
<td>95,60%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise</td>
<td>15</td>
<td>97,95%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise Handsfree</td>
<td>15</td>
<td>94,06%</td>
</tr>
<tr>
<td></td>
<td>Clean</td>
<td></td>
<td></td>
<td>99,54%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise</td>
<td>0</td>
<td>21,37%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise Handsfree</td>
<td>5</td>
<td>46,12%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise</td>
<td>10</td>
<td>75,84%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise Handsfree</td>
<td>15</td>
<td>93,46%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise</td>
<td>5</td>
<td>32,25%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise Handsfree</td>
<td>10</td>
<td>61,87%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise</td>
<td>15</td>
<td>54,99%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise Handsfree</td>
<td>15</td>
<td>85,56%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>80,28%</td>
</tr>
<tr>
<td>ETSI 2</td>
<td>Clean</td>
<td></td>
<td></td>
<td>99,43%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise</td>
<td>0</td>
<td>84,83%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise Handsfree</td>
<td>5</td>
<td>94,63%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise</td>
<td>10</td>
<td>97,87%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G712 CarNoise Handsfree</td>
<td>15</td>
<td>98,76%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise</td>
<td>5</td>
<td>85,78%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise Handsfree</td>
<td>10</td>
<td>93,94%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise</td>
<td>15</td>
<td>84,88%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IntNoise Handsfree</td>
<td>15</td>
<td>97,23%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>90,92%</td>
</tr>
</tbody>
</table>

*Accuracy results without the garbage model*
4.2 Synthesis files

We have mainly worked with 2 kinds of synthesis file; we will refer to them as:

3 Steps:

```
praktiku@master:/data/home/pablo/Training_Recognition_Scripts> cat ti_gender.garbage.syn
<NUMNODES> 3
<SUBCLASS> DIGITS MALE 11 w_1_male w_2_male w_3_male w_4_male w_5_male w_6_male w_7_male w_8_male w_9_male w_r_male
<SUBCLASS> DIGITS FEMALE 11 w_1_female w_2_female w_3_female w_4_female w_5_female w_6_female w_7_female w_8_female w_9_female
w_w_female w_r_female

<NUMCLASS> 1 g_inst
  <NODE> 1
    entries: w_sil
    entry-nodes: 1
  <NODE> 2
    entries: w_sil g_inst DIGITS_FEMALE w_sil
    entry-nodes: 1 2 2 2
  <NODE> 3
    entries: w_sil g_inst DIGITS MALE w_sil
    entry-nodes: 1 3 3 3
  <NUMENODES> 2
  <ENDNODES> 2 3

<SYNONYMS> 11
w_1 2 w_1_male w_1_female
w_2 2 w_2_male w_2_female
w_3 2 w_3_male w_3_female
w_4 2 w_4_male w_4_female
w_5 2 w_5_male w_5_female
w_6 2 w_6_male w_6_female
w_7 2 w_7_male w_7_female
w_8 2 w_8_male w_8_female
w_9 2 w_9_male w_9_female
w_w 2 w_w_male w_w_female
w_o 2 w_o_male w_o_female

<PENALTIES> 30 0 10
<TIMEOUT1> 100
<TIMEOUT2> 125
```

4 steps

```
praktiku@master:/data/home/pablo/Training_Recognition_Scripts> cat ti_gender.garbage_mod.syn
<NUMNODES> 4
<SUBCLASS> DIGITS MALE 11 w_1_male w_2_male w_3_male w_4_male w_5_male w_6_male w_7_male w_8_male w_9_male w_r_male
<SUBCLASS> DIGITS FEMALE 11 w_1_female w_2_female w_3_female w_4_female w_5_female w_6_female w_7_female w_8_female w_9_female
w_w_female w_r_female

<NUMCLASS> 1 g_inst
  <NODE> 1
    entries: w_sil
    entry-nodes: 1
  <NODE> 2
    entries: w_sil DIGITS_FEMALE
    entry-nodes: 1 2
  <NODE> 3
    entries: w_sil DIGITS MALE
    entry-nodes: 1 3
  <NODE> 4
    entries: w_sil g_inst w_sil w_sil
    entry-nodes: 2 4 4 3
  <NUMENODES> 1
  <ENDNODES> 4

<SYNONYMS> 11
w_1 2 w_1_male w_1_female
w_2 2 w_2_male w_2_female
w_3 2 w_3_male w_3_female
w_4 2 w_4_male w_4_female
w_5 2 w_5_male w_5_female
w_6 2 w_6_male w_6_female
w_7 2 w_7_male w_7_female
w_8 2 w_8_male w_8_female
w_9 2 w_9_male w_9_female
w_w 2 w_w_male w_w_female
w_o 2 w_o_male w_o_female

<PENALTIES> 30 0 10
<TIMEOUT1> 100
<TIMEOUT2> 125
```
4.3 Experiments

➔ HGH with 3 states synthesis file – Interior Noise 10 dB SNR - 1st steps

Initially I created a script that was labeling absolutely all the insertions appeared. Normally this should appear at the beginning and ending of the speech as it is the time when we can find silence.

In this case I committed an important error when training the model assuming that it would be good to analyse the whole part of speech where there was silence (just the noise) but also stepping a bit over part of the speech because of the timing of the recognition labeling.

This was not giving very good results. Assuming we were working with non-stationary noise, so many utterances was making the model very likely to happen and although it was managing to avoid insertions, this was not useful as we were obtaining several deletions. As we see here, the accuracy of the system was dropping its efficiency around 8% less respect the HGH without the garbage model.

I wrote this new script restricting just exactly to this segments of, originally, silence but still keeping the labeling of all this files that had insertions without minding where they were.
This brought an important improvement of the results:

➔ HGH with 3 states synthesis file – Interior Noise 10 dB SNR – 1st modification

With this change we made the previous error an evidence and thanks to it we raised the efficiency; however there was still some concept mistake; the results were still around 7% under the previous rates. I show the results obtained with the same library I have used to make the training (Interior Noise with SNR of 10 dB). The rates with other databases (the same samples with different SNR – Clean, 0dB, 5 dB, 15 dB, 20 dB) did not have remarkable different results.

At this point, looking for higher results still using the HGH without adaptation, we opted for restricting even more when training, and decided just to train a model based on the front and end insertions (this which appear when the speaker has not started or has ended her/his speech and we are recording already).
This seemed to be a right way to follow; we are already over the initial results (overcoming them in 6% more or less).

Furthermore, we noticed that with fewer samples we were obtaining better results. For that reason we decided to train the model just with the ending insertions (this was the most common and usually the most bothering).

Besides, we could test a synthesis file that should help to the task when recognizing with this model.
The 4 states synthesis file was assuming that the garbage model had been training with this ending fragments; the idea of it was to drive the system in a way it was recognizing first all the numbers and then just after the series of numbers, see whether there is an insertions or just silence.

⇒ HGH with 3 states synthesis file – Interior Noise 10 dB SNR

⇒ HGH with 4 states synthesis file – Interior Noise 10 dB SNR
Speech recognition - Testing of non-stationary noise suppression method

Pablo Ros Flores

The new synthesis file has not worked as we expected; as before, we guess there is a problem by trying to characterize the garbage; the noise affects the signal in a way it becomes in many cases a lot more likely to match the combination number+noise with the garbage model than the number model. As we can see, the system does not make any difference changing the synthesis file; the garbage model keeps on appearing as a highly more probable model compare to the numbers.

However we had obtain good results (in this case we were 4 % over the original HGH accuracy) so at this point we decided to start working with the HGH Adapt; here we had to made some modifications in the script to let the system work with the garbage model. It is possible to find extended information about the adaptation methods in the referred book “automatic speech recognition in adverse acoustic conditions” by Hans-Günter Hirsch; basically what we try to do is to modify our models which have been trained with a noiseless database (Tidigits) in order to simulate how they’d look like with the addition of noise; this way, as we see in the previous results, we can obtain better results and in theory they should normalize a bit the differences between the garbage model and the others.

Apart from this noise adaptation, there was the possibility to adapt the models to the possible effects of reverberation. This was not a data what to worry much about it thus we decided to avoid this section.

First thing we needed was to have an idea of which results would we get without the garbage model. The results in the table applied both adaptations and there were not exactly the ones to take as a reference. Here we started to work with Matlab (as it is used for the implementation of the adaptation) and I initially run some experiments to know the “basic” results:

Interior Noise SNR 5dB

<table>
<thead>
<tr>
<th>Overall Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENT: %Correct=64.11 [H=5578, S=3122, N=8700]</td>
</tr>
<tr>
<td>WORD: %Corr=87.37, Acc=84.56 [H=24974, D=1924, S=1685, I=805, N=28583]</td>
</tr>
</tbody>
</table>

Interior Noise SNR 10dB

<table>
<thead>
<tr>
<th>Overall Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENT: %Correct=84.77 [H=7375, S=1325, N=8708]</td>
</tr>
<tr>
<td>WORD: %Corr=95.70, Acc=94.31 [H=27355, D=490, S=738, I=399, N=28583]</td>
</tr>
</tbody>
</table>

Interior Noise SNR 15dB

<table>
<thead>
<tr>
<th>Overall Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENT: %Correct=92.90 [H=8082, S=618, N=8700]</td>
</tr>
<tr>
<td>WORD: %Corr=98.14, Acc=97.58 [H=28050, D=157, S=376, I=182, N=28583]</td>
</tr>
</tbody>
</table>

Interior Noise Clean

<table>
<thead>
<tr>
<th>Overall Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENT: %Correct=98.32 [H=8553, S=146, N=8699]</td>
</tr>
<tr>
<td>WORD: %Corr=99.50, Acc=99.43 [H=28439, D=58, S=84, I=21, N=28581]</td>
</tr>
</tbody>
</table>
With the garbage model we obtained the following results:

➔ HGH ADAPT with 3 states synthesis file – Interior Noise 5 dB SNR

➔ HGH ADAPT with 3 states synthesis file – Interior Noise 10 dB SNR
Speech recognition - Testing of non-stationary noise suppression method

➔ HGH ADAPT with 3 states synthesis file – Interior Noise 15 dB SNR

➔ HGH ADAPT with 3 states synthesis file – Clean
The results were far to overcome the basic results. Trying to solve the problem we tried different alternatives focusing on the interior noise with SNR of 10 dB:

➔ **HGH ADAPT with 4 states synthesis file – Interior Noise 10 dB SNR**

```
praktiku@master:/data/home/pablo/Training_Recognition_Scripts>
./eval_results/InteriorNoise10_ML_rec_4coeff_ginst_ending_4statesSyn

* RECOGNITION RESULTS
  * Frontend: HGH
  * Date: mié jul 15 11:00:59 CEST 2009

 recognition of Aurora

Mtk Results Analysis
Date: Wed Jul 15 11:01:08 2009
Ref : /data/tidigits/label
Rec : InteriorNoise10_ML_rec_4coeff_ginst_ending_4statesSyn/11152.rmi.rec
      InteriorNoise10_ML_rec_4coeff_ginst_ending_4statesSyn/1119673.rbw.rec
      InteriorNoise10_ML_rec_4coeff_ginst_ending_4statesSyn/1119673.rpf.rec
      InteriorNoise10_ML_rec_4coeff_ginst_ending_4statesSyn/1121851.rap.rec

* Overall results

  SENT: %Correct=72.03 [H=6336, S=2364, N=8790]
  WORD: %Correct=88.99, Acc=87.84 [H=25436, D=2583, S=564, I=328, N=28583]

Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Del</th>
<th>%c / %w</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>185</td>
<td>99.1/0.1</td>
</tr>
<tr>
<td>w_1</td>
<td>0</td>
<td>2475</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>114</td>
</tr>
<tr>
<td>w_6</td>
<td>6</td>
<td>3</td>
<td>2093</td>
<td>11</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>68</td>
<td>3</td>
<td>4</td>
<td>425</td>
</tr>
<tr>
<td>w_0</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2199</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>291</td>
<td>57.3/0.8</td>
<td></td>
</tr>
<tr>
<td>w_3</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>2485</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>177</td>
</tr>
<tr>
<td>w_4</td>
<td>4</td>
<td>1</td>
<td>54</td>
<td>0</td>
<td>17</td>
<td>2318</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>206</td>
<td>99.0/0.0</td>
</tr>
<tr>
<td>w_5</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>6247</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>153</td>
<td>98.0/0.1</td>
<td></td>
</tr>
<tr>
<td>w_0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2214</td>
<td>35</td>
<td>2</td>
<td>2</td>
<td>347</td>
<td>97.9/0.2</td>
</tr>
<tr>
<td>w_7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4340</td>
<td>1</td>
<td>2</td>
<td>161</td>
<td>99.0/0.0</td>
<td></td>
</tr>
<tr>
<td>w_8</td>
<td>3</td>
<td>4</td>
<td>48</td>
<td>3</td>
<td>5</td>
<td>2153</td>
<td>7</td>
<td>334</td>
<td>94.4/0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>w_9</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>11</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>2300</td>
<td>98.0/0.1</td>
</tr>
<tr>
<td>Ins</td>
<td>13</td>
<td>38</td>
<td>22</td>
<td>18</td>
<td>5</td>
<td>216</td>
<td>42</td>
<td>44</td>
<td>8</td>
<td>36</td>
<td></td>
</tr>
</tbody>
</table>
```

Definitely this synthesis file was not giving the expected results.

Then we tried to take an smaller number of samples to train the garbage model; as it had become a too likely model, a lower amount of samples could give us a preciser recognition:

➔ **HGH ADAPT with 3 states synthesis file**

- Interior Noise 10 dB SNR
- Random
It was an evidence that the model was not working as we expected. The last I have tried is to “attack” directly over the garbage HMM by changing the probability to “start” recognizing this model. This can be done by means of changing the first to values of the convolution matrix diagonal.

The original values of our model were:

\[5.264785e-01 \& 4.735215e-01\]

Then we have been trying to drop it:

➔ HGH ADAPT with 3 states synthesis file
- Interior Noise 10 dB SNR

\[-0.5264785e-01\]
\[-0.4735215e-01\]

➔ HGH ADAPT with 3 states synthesis file
- Interior Noise 10 dB SNR

\[-0.000000000000000000164785e-01\]
\[-0.000000000000000000135215e-01\]
There has been enough improvement with the modifications done as to believe there was some possibility to overcome the previous results.

The way we have tried to characterize the non-stationary noise has not been successful probably because of the own nature of this noises.

The system was always finding several coincidences between the input signal and the garbage model. We have managed to avoid insertions but this was not optimum when in the other hand the number of deletions was rising so dramatically.

I can ensure thus, with the experiments made, that the process followed and explained along the project does not perform as expected and that for this precise purpose of finding some patterns for the noise is not advisable.
5. Resources

- Advances in digital speech transmission - Rainer Martin, Ulrich Heute, Christiane Antweiler
  2008 John Wiley & Sons Ltd.
- Handbook of speech processing - Jacob Benesty, M. Mohan Sondhi, Yiteng Huang
  2008 Springer-Verlag Berlin Heidelberg
- Sprachverarbeitung - Beat Pfister and Tobias Kaufmann
  2008 Springer-Verlag Berlin Heidelberg
- The HTK Book (for HTK Version 3.4)
- Speech recognition over digital channels - Antonio M. Peinado & C. Segura
  2006 John Wiley & Sons Ltd.