**TECHNICAL UNIVERSITY OF CRETE**

**Department of Electronics & Computer Engineering**

# SOFT FEATURE DECODING FOR SPEECH RECOGNITION OVER WIRELESS CHANNELS

**a thesis by**

**Dermatas Konstantinos**

**Supervisor Committee**

**Potamianos Alexandros (Supervisor)**
**Digalakis Vassilios**
**Sidiropoulos Nikolaos**

*to my parents*

# Table of contents

# ACKNOWLEDGEMENTS

I would like to thank my advisor, Pr. Alexandros Potamianos, for his invaluable guidance.

I am also very grateful to the other members of the supervisory committee.

I would also like to thank my family for their pure love and support all these years where I studied in Technical University of Crete.

Finally, a big thank to my friends and especially to Georgia.

# Abstract

We begin this thesis with a small introduction to the topic. We discuss the main aims of our work. In the first chapter, we will see the basic ideas and functions which there are in all of the speech recognition systems nowadays. More specific we will talk about continuous speech recognition. We will explain with details the use of Hidden Markov Models (HMMs) and the ideas of initialization and training of HMMs in these systems. The probabilities of the observation sequences will be used by the Viterbi recognition algorithm to produce the final results. The chapter that follows shows the how the tools of HTK toolkit implement the functions we saw in the first chapter.

The third chapter is an introduction to the automatic speech recognition process in mobile networks. In our work, we have used a distributed speech recognition system, so we discuss the main functions of it. We have the signal parameterization, quantization and coding in the user side and in the server side there are the signal decoding and the speech recognition process.

Chapter four explains the theory of Missing Feature. One of the main aims of this thesis is to apply the technique of Soft Feature decoding in the server side to improve the recognition results. We discuss three basic techniques in this area:

- Data Imputation
- Marginalize Unreliable Feature
- Exponential Feature Weights

In chapter five, there is all the information about the implementation we followed. We will see with all the details all the components of the implemented system. Parameterization of the speech signal and its quantization are the first functions occur in the mobile terminal. The transmission system with the coding algorithm and an unequal error protection scheme follows. In the server side there is the most important part of this work. The Soft Feature Decoding algorithm is explained.

We can see the results we obtain with the implementation of Soft Feature Decoding in speech recognition over wireless channel in chapter six. We used two different speech databases to evaluate better Soft Feature algorithm, TIMIT database and AURORA 2. We also present the tests we made to be sure that the system we created works correctly. In the end there are the conclusions and some ideas for future work.

# Introduction

Due to the expected explosion in the use of wireless devices in cellular environments, it is expected that the interest in deployment of mobile speech application will continue, and that the interest will further research in robust mobile ASR considerably. This thesis aims to contribute in the continuous efforts to improve speech recognition in mobile ASR applications. We explicitly focus in the problem of recognition through a wireless digital communication network.

In this thesis, we applied the Missing Feature Theory in speech recognition for applications in mobile network. In a wireless channel the noise is the main factor which degrades the performance of speech recognizer. So, if we cannot avoid the noise why not to try to minimize its effects to the recognition process. The idea of a Soft Feature Decoder is to try to give a confidence level of receive correctly each feature from the decoder in the server side. The outputs of the Soft Feature Decoder they will be used from the speech recognition unit in order to achieve better performance. In other words, the speech recognizer is going to count less on features which are more probably to be noisy.

We use the model of distributed speech recognition in mobile networks. We have the terminal side and the server side. In the first part, we parameterized the signal voice of the user and we produce the MFC coefficients which we will use later for the final recognition. Then we apply a scalar non-uniform quantizer and we code the signal to transmit through the channel applying an unequal error protection scheme. We have to mention that the aim of this work in not to find the best quantization algorithm or coding scheme. So, our work in mobile terminal is based on [6] which have been proved that in general gives very well results in speech recognition over wireless channels.

In the server side the functions of decoding and speech recognition occur. The decoder generates the transmitted sequence of bits. The Soft Feature decoding algorithm which we want to apply in this thesis it is placed in this part. We examine each bits we receive in the decoder and we assign a probability which shown how sure we are that this bit has been received correctly. Then we group the bits and we produce the features, in our case MFC coefficients. Now, we use the probability of each bit to determine the confidence level of each received feature. This confidence score is a continuous value between zero and one. When this value is close to one, it means that the feature is more probably to have been received correctly and when it is close to zero the opposite. Theses values are used later by the speech recognition unit. So for example, a value of 0.55, tells to recognizer to multiply the output probability of this feature with 0.55. In this way we hide from the recognizer the features which we believe that have been changed due to noise of the channel and we minimize the its effects. Using noisy feature gives worst results than delete them. So, this is an easy way to improve the performance in speech recognition over wireless channels.

The technique of Soft Feature decoding is promising a big improvement in recognition performance. This technique has been implemented already in other speech recognition system for mobile networks and it has shown to improve the recognition accuracy. This thesis is based on the work of [2]. We use the same Soft Feature Decoding algorithm but we have many differences in the whole speech recognition system. We can say that two are the main characteristics that distinguish the system proposed in [2] and ours; the speech recognition unit and the speech database. In our work we used the HTK toolkit for speech recognition and two different databases; DARPA TIMIT and AURORA 2.

When we applied the Soft Feature algorithm in TIMIT database we did not see any improvement in the final results. This is in contrary, not only with the theory but also with the results of other researches. So, with the first look we thought that somewhere we had made a mistake and we spent a long time searching for the error. We made a lot of tests and finally we decided that there is no error in the system. Something is going wrong with the characteristic of the speech recognition system. The important difference between this thesis and the work of the other researchers in the recognition system is the level in which we achieve the recognition. In our application using TIMIT database we have a phoneme base recognition instead of word level which there is in [2].

So, we decided to work also with a word based speech database and we chose AURORA 2 which has for vocabulary the ten digits. With this change, the results we achieved have a significant improvement. In general, the improvement is independent of the mobile speed and higher improvement is shown in lower SNR channel. The reduction of word error rate is around 50% for all the occasions, except for channels with very high SNR and speed. The improvement is substantial and is equivalent to enhancing the channel SNR by about 1.5db and even in some occasions 2db.

The conclusions for this thesis agree with the works of other researchers in the same area. The Soft Feature Decoding can give a significant improvement in speech recognition over wireless channel. Of course, more work is needed to improve some parts of the whole process. More research is underway to tune the parameters of Soft Feature algorithm we used in this thesis, in order to further improve performance.

# Chapter 1

# Few words for speech recognition

In this first introductory chapter we discuss the main functions that occur in all of the speech recognition system nowadays. In this thesis, the HTK toolkit was used for the whole process of speech recognition. So, we use the "HTK book" as a guide reference to explain the basic theory of a speech recognition system.

## 1.1 Introduction

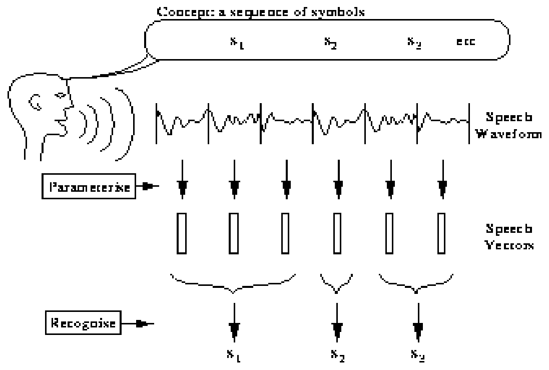

Fig. 1.1 Message Encoding/Decoding

Speech recognition systems generally assume that the speech signal is a realization of some message encoded as a sequence of one or more symbols (see Fig, 1.1). To effect the reverse operation of recognizing the underlying symbol sequence given a spoken utterance, the continuous speech waveform is first converted to a sequence of equally spaced discrete parameter vector. This sequence of parameter vector is assumed to form an exact representation of the speech waveform on the basis that for the duration covered by a single vector (typically 10ms), the speech waveform can be regarded as a stationary. Although this is not strictly true, it is a reasonable approximation. Typical parametric representations in common use are Linear Prediction Coefficients

(LPC) or Mel Frequency Cepstral Coefficients (MFCC) plus various other representations derived from these.

The role of the recognizer is to effect a mapping between sequences of speech vectors and the wanted underlying symbol sequences. Two problems make this very difficult. Firstly, the mapping from symbol to speech is not one-to-one since different underlying symbols can rise to similar speech sounds. Furthermore, there are large variations in the realized speech waveform due to speaker variability, mood, environment, etc. Secondly, the boundaries between symbols cannot be identified explicitly from the speech waveform. Hence, it is not possible to treat the speech waveform as a sequence of concatenate static patterns.

The speech recognizer is designed according to the needs of each application. There are simple applications that utilize Isolated Word recognition and more advanced ones that try to recognize Continuous Speech. The Continuous Speech recognition, which is the most interesting and is used more, can be divided in two categories according to the assumed underlying symbol. These could be either whole words for so-called *connected speech recognition* or sub-words such as phonemes for *continuous speech recognition*.

No matter what kind of application we want to create, there are some basic ideas that are used in all the speech recognizer nowadays. The main tool we use in speech recognition is the Hidden Markov Models (HMMs). Speech recognizer, as all pattern recognizers, has two main parts, *Training* and *Testing*. The main task in first part is the creation and training of HMMs and in the next part we use the trained HMMs to recognize the underlying symbols. We are going to see in more details the HMMs and how they are used by speech recognizers.

## 1.2 HMMs in speech recognition

In continuous speech recognition each phoneme is represented by a sequence of speech vectors or *observations* **O,** defined as

$$\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T \qquad (1.1)$$

where $\mathbf{o}_t$ is the speech vector observed at time t. The continuous speech recognition problem can then be regarded as that of computing

$$\arg\max_i \left\{ P(w_i \mid \mathbf{O}) \right\} \tag{1.2}$$

where $w_i$ is the *i*'th phoneme from the vocabulary. This probability is not computable directly but using Bayes' Rule gives

$$P(w_i|\mathbf{O}) = \frac{P(\mathbf{O}|w_i)P(w_i)}{P(\mathbf{O})} \tag{1.3}$$

Thus, for a given set of prior probabilities $P(w_i)$, the most probable spoken phoneme depends only on the likelihood $P(\mathbf{O}|w_i)$. Given the dimensionality of the observation sequence **O**, the direct estimation of the joint conditional probability $P(\mathbf{o}_1, \mathbf{o}_2, \ldots| w_i)$ from examples of spoken words is not practicable. However, if a parametric model of phoneme production such as Markov model is assumed, then estimation from data is possible since the problem of estimating the class conditional observation densities $P(\mathbf{O}| w_i)$ is replaced by the much simpler problem of estimating the Markov model parameters.

In HMM based speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each phoneme is generated by a Markov model as shown is Fig. 1.2. A Markov model is a finite state machine which changes state once every time unit *t* that a state *j* is entered, a speech vector $\mathbf{o}_T$ is generated from probability density $b_j(\mathbf{o}_T)$. Furthermore, the transition from state *i* to state *j* is also probabilistic and is governed by the discrete probability $a_{ij}$. Fig. 1.2 shows an example of this process where the six state model moves through the state sequence X= 1, 2, 3, 4, 5, 6 in order to generate a sequence $\mathbf{o}_1$ to $\mathbf{o}_6$. Notice that in HTK, the entry and exit states of a HMM are no-emitting. These states provide the "glue" needed to join models together.

The joint probability that **O** is generated by the model M moving through the state sequence X is calculated simply as the product of the transcription probabilities and the output probabilities. So for the state sequence X in Fig. 1.2

$$P(\mathbf{O}, X \mid M) = a_{12} b_2(\mathbf{o}_1) a_{22} b_2(\mathbf{o}_2) a_{23} b(\mathbf{o}_3) \ldots \qquad (1.4)$$

However, in practice, only the observation sequence **O** is known and the underlying state sequence X is hidden. This is why it is called *Hidden Markov Model*.

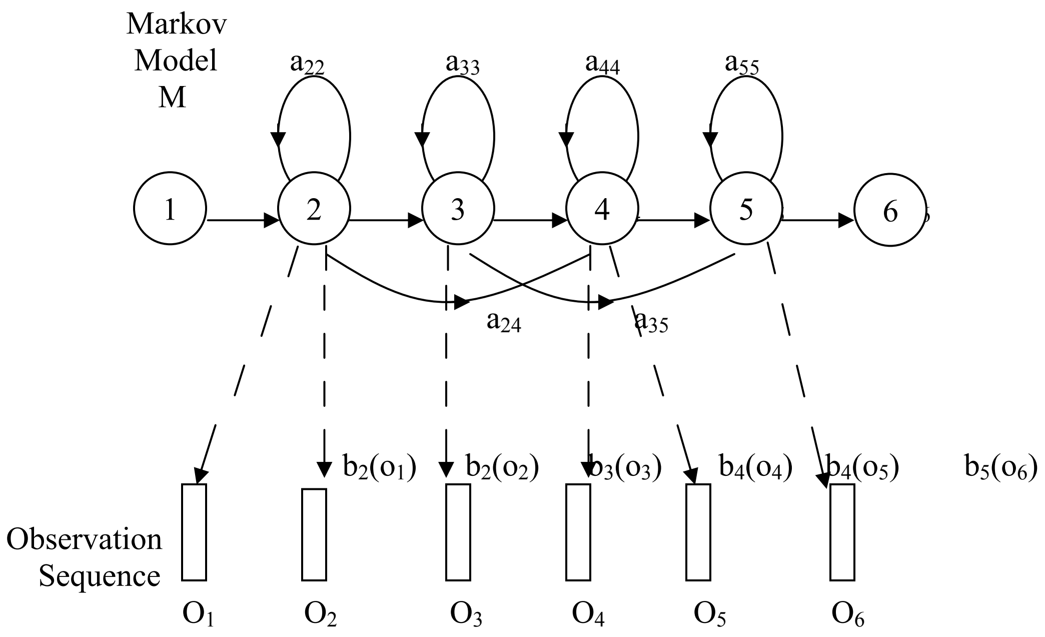

Fig. 1.2 The Markov generation model

Given that X is unknown, the required likelihood is computed by summing over all possible state sequences X = x(1),x(2),...,x(T), that is

$$P(\mathbf{O} \mid M) = \sum_{x} a_{\chi(0)\chi(1)} \prod_{t=1}^{T} b_{x(t)}(\mathbf{o}t) a_{x(t)x(t+1)} \qquad (1.5)$$

where x(0) is constrained to be the model entry state and x(T+1) is constrained to be the model exit state. As an alternative to equation 1.5, the likelihood can be approximated by only considering the most likely state sequence, that is

$$\hat{P}(\mathbf{O}|M) = \max_{x} \left\{ a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(\mathbf{o}t) a_{x(t)x(T+1)} \right\} \quad (1.6)$$

Although the direct computation of equation 1.5 and 1.6 is not tractable, simple recursive procedures exist which allow both quantities to be calculated very efficiently. Notice that if equation 1.2 is computable then recognition problem solved. Given a set of models *Mi* corresponding to phonemes $w_i$, equation 1.2 is solved by using 1.3 and assuming that

$$P(\mathbf{O}|w_i) = P(\mathbf{O}|M_i) \quad (1.7)$$

All this of course, assumes that the parameters $\{a_{ij}\}$ and $\{b_j(\mathbf{o}_t)\}$ are known for each model *Mi*. Herein lies the elegance and the power of the HMM framework. Given a set of training examples corresponding to a particular model, the parameters of that model can be determined automatically by a robust and efficient re-estimation procedure. Thus, provided that a sufficient number of representative examples of each phoneme can be collected then a HMM can be constructed which implicitly models all of the many sources of variability inherent in real speech.

## 1.3 Output Probability Specification

The form of the output distributions $\{b_j(\mathbf{o}_t)\}$ needs to be made explicit. HTK is designed primarily for modeling continuous parameters using continuous density multivariate output distributions. It can also handle observation sequences consisting if discrete symbols in which case, the output distributions are discrete probabilities. For simplicity will assume that continuous density distributions are being used. In common with most other continuous density HMM systems, HTK represents output distributions by Gaussian Mixture Densities. In HTK, however, a further generalization is made. HTK allows each observation vector at time to be split into a number of S independent data streams $\mathbf{o}_{st}$. The formula for computing $b_j(\mathbf{o}_t)$ is then

$$b_j(\mathbf{o}_t) = \prod_{s=1}^{S} \left[ \sum_{m=1}^{Ms} c_{jsm} N(\mathbf{o}_{st}; \boldsymbol{\mu}_{jsm}, \boldsymbol{\Sigma}_{jsm}) \right]^{\gamma_s} \tag{1.8}$$

where $M_s$ is the number of mixture components in stream s, $c_{jsm}$ is the weight of the m'th component and $N(.;\mu,\Sigma)$ is a multivariate Gaussian with mean vector $\mu$ and a covariance matrix $\Sigma$, that is

$$N(\mathbf{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\mathbf{o}-\boldsymbol{\mu})} \tag{1.9}$$

where n is the dimensionality of **o.** The exponent $\gamma_s$ is a stream weight. It can be used to give a particular stream more emphasis.

Multiply data streams are used to enable separate modeling of multiply information sources. In HTK, the processing of streams is completely general. However, the speech input modules assumes that the source data is split into at most 4 streams. The main streams are the basic parameters vector, first (delta) and second (acceleration) difference coefficient and log energy.

## 1.4 Baum-Welch Re-estimation

To determine the parameters of a HMM it is first necessary to make a rough guess at what they might be. Once this is done, more accurate (in the maximum likelihood sense) parameters can be found by applying the so-called Baum-Welch re-estimation formula. The basis of the formula will be represented in a very informal way. Firstly, it should be noted that the inclusion of multiple data streams does not alter matters significantly since each stream is considered to be statistically independent. Furthermore, mixture components can be a special form of sub-state in which the transition probabilities are the mixture weights (see Fig. 1.3).

Thus, the essential problem is to estimate the means and variances of a HMM in which each state output distribution is a single component Gaussian, that is

$$b_j\left(\mathbf{o}_t\right)=\frac{1}{\sqrt{(2\pi)^n\left|\boldsymbol{\Sigma}_j\right|}}e^{-\frac{1}{2}\left(\mathbf{o}_t-\boldsymbol{\mu}_j\right)'\boldsymbol{\Sigma}_j^{-1}\left(\mathbf{o}_t-\boldsymbol{\mu}_j\right)} \qquad (1.10)$$

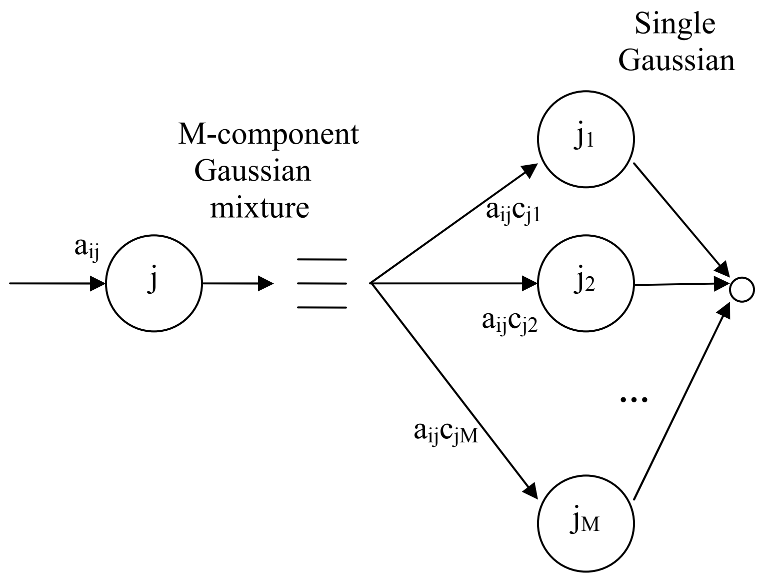

Fig. 1.3 Representing a Mixture

If there was just one state j in the HMM, this parameter estimation would be easy. The maximum likelihood estimates of $\boldsymbol{\mu}_\mathbf{j}$ and $\boldsymbol{\Sigma}_\mathbf{j}$ would be just the simple average, that is

$$\hat{\boldsymbol{\mu}}_j = \frac{1}{T}\sum_{t=1}^{T}\mathbf{o}_t \tag{1.11}$$

and

$$\hat{\boldsymbol{\Sigma}}_j = \frac{1}{T}\sum_{t=1}^{T}\left(\mathbf{o}_t - \boldsymbol{\mu}_j\right)\left(\mathbf{o}_t - \boldsymbol{\mu}_j\right)' \tag{1.12}$$

In practice, of course, there are multiple states and there is no direct assignment of observation vector to individual states because the underlying state sequence is unknown. Note, however, that if some approximate assignment of vectors to states could be made then equations 1.11 and 1.12 could be used to give the required initial values for the parameters. Indeed, this is exactly what is done in the HTK tool called HInit. HInit first divides the training observation vectors equally amongst the model states and then uses equations 1.11 and 1.12 to give the initial values for the mean and variance of each state. If then finds the maximum likelihood state sequence using Viterbi algorithm described below, reassigns the observation vectors to states and then uses equations 1.11 and 1.12 again to get better initial values. This process is repeated until the estimates do not change.

Since the full likelihood of each observation sequence is based on the summation of the possible state sequences, each observation vector $\mathbf{o_t}$ contributes to the computation of the maximum likelihood parameter values for each state j. In other words, instead of assigning each observation vector to a specific state as in the above approximation, each observation is assigned to every state in proportion to the probability of the model being in that state when the vector was observed. Thus, if $L_j(t)$ denotes the probability of being in state j at time t then the equations 1.11 and 1.12 given above become the following weighted averages

$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_{t=1}^{T} L_j(t)\mathbf{o}_t}{\sum_{t=1}^{T} L_j(t)} \tag{1.13}$$

and

$$\hat{\Sigma}_j = \frac{\sum_{t=1}^{T} L_j(t)(\mathbf{o}_t - \boldsymbol{\mu}_j)(\mathbf{o}_t - \boldsymbol{\mu}_j)'}{\sum_{t=1}^{T} L_j(t)} \tag{1.14}$$

where the summations in the denominators are included to give the required normalization. Equations 1.13 and 1.14 are the Baum-Welch re-estimation formula for the means and covariances of a HMM. Of course, to apply the above equations, the probability of state occupation $L_j(t)$ must be calculated. This is done efficiently using the so-called *Forward-Backward* algorithm.

## 1.5 Recognition and Viterbi decoding

The previous section has described the basis ideas underlying HMM parameters re-estimation using the Baum-Welch algorithm. In passing, it was noted the efficient recursive algorithm for computing the forward probability also yielded as a by-product the total likelihood P(**O**|M). Thus, this algorithm could also be used to find the model which yields the maximum value of P(**O**|$M_i$), and hence, it could be used for recognition.

In practice, however, it is preferable to base recognition on the maximum likelihood state sequence since this generalizes easily to the continuous speech case whereas the use of the total probability does not. This likelihood is computed using essentially the same algorithm as the forward probability calculation except that the summation is replaced by a maximum operation. For a given model M, let $\Phi_j(t)$ represent the maximum likelihood of observing speech vector $\mathbf{o_1}$ to $\mathbf{o_t}$ and being in state j at time t. This partial likelihood can be computed efficiently using the following recursion

$$\phi_j(t) = \max_i \{\phi_i(t-1) a_{ij}\} b_j(\mathbf{o}_t) \tag{1.15}$$

where

$$\phi_1(1) = 1 \tag{1.16}$$

$$\phi_j(1) = a_{1j} b_j(\mathbf{o}_1) \tag{1.17}$$

for 1<j<N. The maximum likelihood $\hat{P}(\mathbf{O} \mid M)$ is then given by

$$\phi_N(T) = \max_i \left\{ \phi_i(T) a_{iN} \right\} \tag{1.18}$$

As for the re-estimation case, the direct computation of likelihoods leads to underflow; hence, log likelihoods are used instead. The recursion of equation 1.15 then becomes

$$\psi_j(t) = \max_i \left\{ \psi_i(t-1) + \log(a_{ij}) \right\} + \log(b_j(\mathbf{o}_t)) \tag{1.18}$$

This recursion forms the basis of the so-called Viterbi algorithm and it is used from HTK tool HVite for speech recognition.

## 1.6 Summary

In this first chapter we introduced the basic theory for speech recognition and we discussed the main parts of a speech recognition system. Also we explained the use of Hidden Markov Models in speech recognition, the models that most of speech recognizers are using nowadays. We made clear the separation of all the speech recognition systems in two parts, the Training and the Test. We saw the main techniques that are used in each part, in Training where the main task in the creation of the HMMs and in Test where the decisions for the recognition are taken using the HMMs from the first part of the system. We saw how two basic algorithms for speech recognition work (Baum-Welch, Viterbi).

# Chapter 2

# Speech recognition with HTK

In this chapter we will see how the HTK toolkit implements the main parts of a speech recognition system. As we mentioned in the introduction one of the main aims of this thesis is to apply a new function in the HTK toolkit in the part of the recognition. So, it is necessary to have a clear image, how the HTK implements the main functions of a speech recognition system, we have shown in the first chapter. Once again, we will use the "HTK book" as a reference to present speech recognition with HTK.

## 2.1 An overview of the HTK toolkit

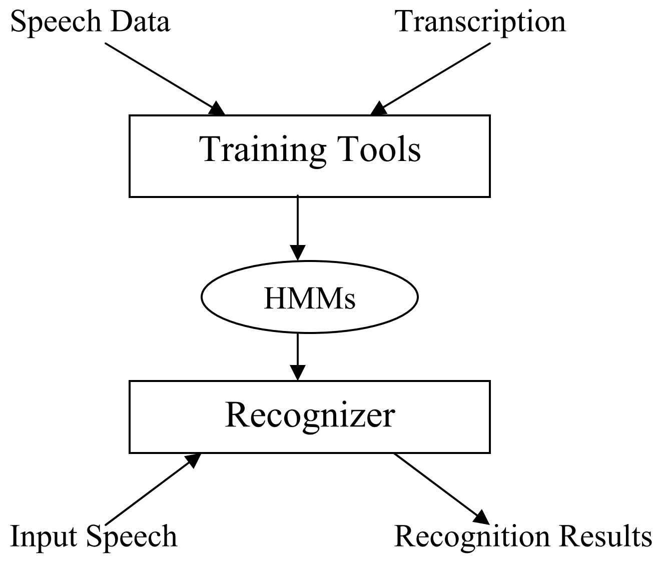

Fig. 2.1 The general image

HTK is a toolkit for building Hidden Markov Models (HMMs). HMM can be used to model any time series and the core of HTK is similarly general-purpose. However, HTK is primarily designed for building HMM-based speech processing tools, in particular recognizers. Thus, much of the infrastructure support in HTK is dedicated to this task. As shown in Fig. 2.1, there are two major processing stages involved. Firstly, the HTK training tools are used to estimate the parameters of a set of HMMs using training utterances and their associated transcriptions. Secondly, unknown utterances are transcribed using the HTK recognition tools.

We will see in more details HTK tools by going through the processing steps involved in building a sub-word based continuous speech recognizer. As shown in Fig. 2.2, there are 4 main phases:

1. data preparation
2. training
3. testing
4. analysis

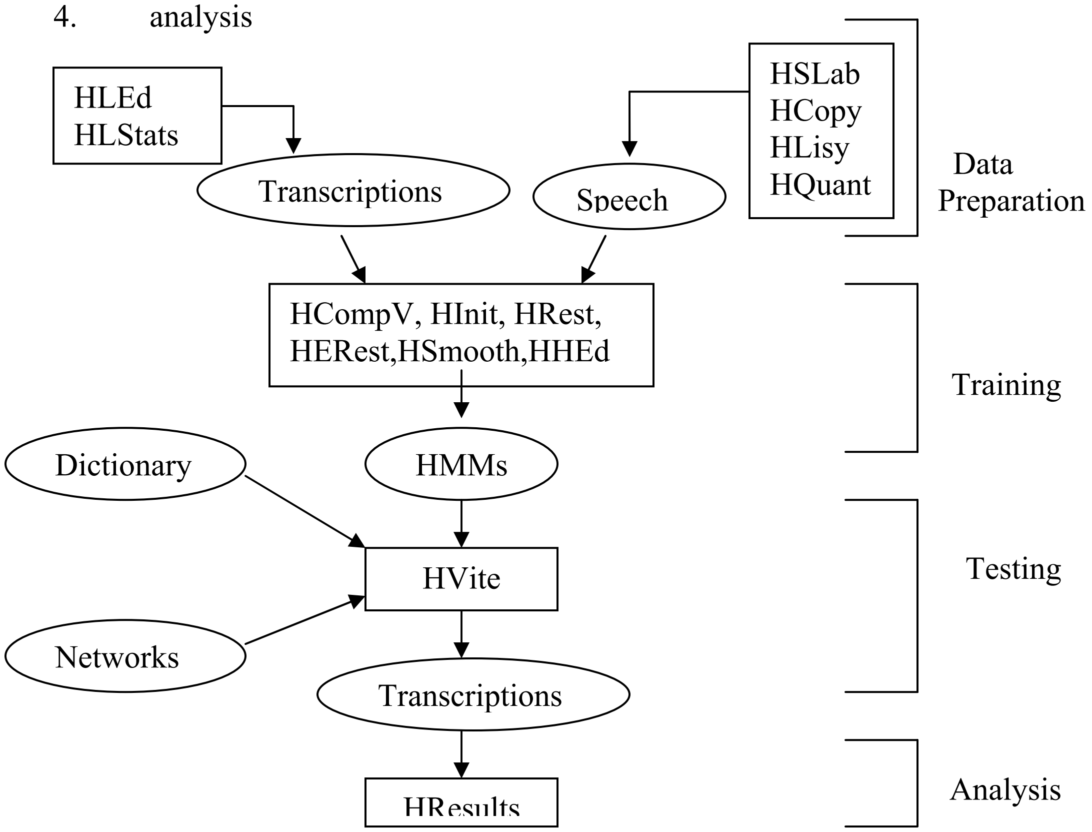

Fig. 2.2 HTK processing stages

## 2.2 Data preparation

In order to build a set of HMMs, a set of speech data files and theirs associated transcriptions are required. Very often speech data will be obtained from databases archives. Before it can be used in training, it must be converted into the appropriate parametric form and any associated transcriptions must be converted to have the correct format and use the required phone or word labels.

Although all HTK tools can parameterise waveforms on-the-fly, in practice it is usually better to parameterise the data just once. The tool HCopy is used for this. As the name suggests, HCopy is used to copy one or more source files to an output file. Normally, HCopy copies the whole file, but a variety of mechanisms are provided for extracting segments of files and concatenating files. By setting the appropriate configuration variables, all input files can be converted to parametric form as they are read-in. Thus, simply copying each file in this manner performs the required encoding. In stage of data preparation there are more available functions but we are not going to reference them in this thesis.

## 2.3 Training process

The second and the most important step of system building is to define the topology required for each HMM by writing a prototype definition. HTK allows HMMs to be built with any desired topology. HMM definitions can be stored externally as simple text files and hence it is possible to edit them with any convenient text editor. Alternatively, the standard HTK distribution includes a number of example HMM prototypes and a script to generate the most common topologies automatically. With the exception of the transition probabilities, all of the HMM parameters given in the prototype definition are ignored. The purpose of the prototype definition is only to specify the overall characteristics and topology of the HMM. The actual parameters will be computed later by the training tools. Sensible values for the transition probabilities must be given but the training process is very insensitive to these. An acceptable and simple strategy for choosing these probabilities is to make all of the transitions out of any state equally likely.

The actual training process takes place in stages and it is illustrated in more detail in Fig. 2.3. Firstly, an initial set of models must be created. If there is some speech data available for which the location of the sub-word (i.e. phone) boundaries has been marked, then this can be used as bootstrap data. In this case, the tools HInit and HRest provide isolated word style training using the fully labelled bootstrap data. Each of the required HMMs is generated individually. HInit reads in all of the bootstrap training data and cuts out all of the examples of the required phone. It then iteratively computes an initial set of parameter values using a segmental k-means procedure. On the first cycle, the training data is uniformly segmented, each model state is matched with the corresponding data segments and then means and variances are estimated. If mixture Gaussian models are being trained, then a modified form of k-means clustering is used. On the second and successive cycles, the uniform segmentation is replaced by Viterbi alignment. The initial parameter values computed by Hinit are then further re-estimated by HRest. Again, the fully labelled bootstrap data is used but this time the segmental k-means procedure is replaced by the Baum-Welch re-estimation procedure described in the previous chapter.

Once an initial set of models has been created, the tool HERest is used to perform embedded training using the entire training set. HERest performs a single Baum-Welch re-estimation of the whole set of HMM phone models simultaneously. For each training utterance, the corresponding phone models are concatenated and then the forward-backward algorithm is used to accumulate the statistics of state occupation, means, variances, etc., for each HMM in the sequence. When all of the training data has been processed, the accumulated statistics are used to compute re-estimates of the HMM parameters. HERest is the core HTK training tool. It is designed to process large databases, it has facilities for pruning to reduce computation and it can be run in parallel across a network of machines.

The philosophy of system construction in HTK is that HMMs should be refined incrementally. Thus, a typical progression is to start with a simple set of single Gaussian context-independent phone models and then iteratively refine them by expanding them to include context-dependency and use multiple mixture component Gaussian distributions. The tool HHed is a HMM definition editor which will clone models into context-dependent sets, apply a variety of parameter tyings and increment the number of mixture components in specified distributions. The usual process is to modify a set of HMMs in stages using HHed and then re-estimate the parameters of the modified set using HERest after each stage.

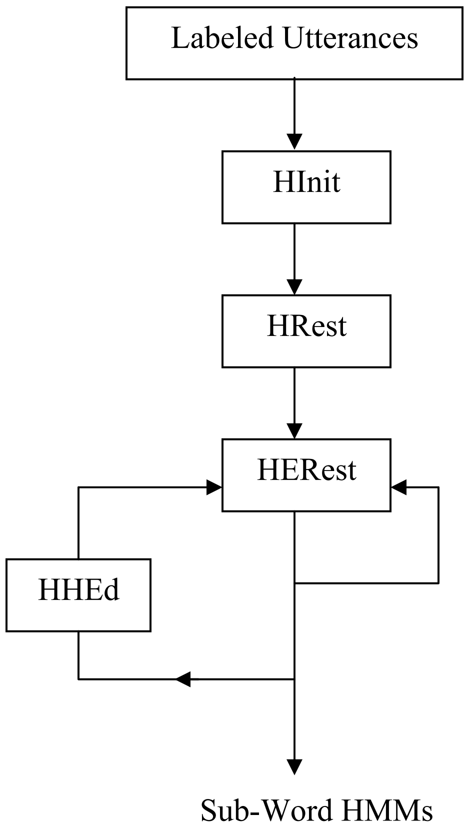

Fig. 2.3 Training Sub-words HMMs

## 2.4 Recognition process

HTK provides a single recognition tool called HVite which performs Viterbi-based recognition, as we described in the previous chapter. HVite takes as input a network described the allowable word sequences, a dictionary defining how each word is pronounce and a set of HMMs. It operates by converting the word network to a phone network and then attaching the appropriate HMM definition to each phone instance. Recognition can then be performed on either a list of stored speech files or on direct audio file. HVite can support cross-word triphones and it can run with multiple tokens to generate lattices containing multiple hypotheses. It can also be configured to rescore lattices and perform forced alignments.

## 2.5 Analysis tool

Once the HMM-based recogniser has been built, it is necessary to evaluate its performance. This is usually done by using it to transcribe some pre-recorded test sentences and match the recogniser output with the correct reference transcriptions. This comparison is performed by a tool called HResults which uses dynamic programming to align the two transcriptions and then count substitution, deletion and insertion errors. Options are provided to ensure that the algorithms and output formats used by HResults are compatible with those used by the US National Institute of Standards and Technology (NIST). As well as global performance measures, HResults can also provide speaker-by-speaker breakdowns, confusion matrices and time-aligned transcriptions.

## 2.6 Summary

In this chapter we saw in more details a complete system of speech recognition. HTK is a specific toolkit which is used widely nowadays for speech recognition. We analysed the main parts of HTK and saw the main functions that are used in each part. HTK is the tool that we are using for speech recognition in this thesis, so this chapter is important for having a complete image of the whole process of speech recognition. In this thesis we made some changes in the Testing part of HTK and more specific in HVite function.

# Chapter 3

# Automatic speech recognition in mobile networks

## 3.1 Introduction

In this chapter we will see the different kind of architectures that are used in mobile networks to perform automatic speech recognition. Also we will see the main techniques for feature extraction and the quantization of them. In the end we will say few words for the coding of speech signal.

Recent progress in Automatic Speech Recognition (ASR) technology has enabled the development and deployment of more sophisticated and more accurate speech recognition applications. This progress, combined with an explosion in the capabilities and use of wireless and mobile communication and computing terminal devices, makes it feasible to become a common feature and services for current and future portable terminal devices and in mobile or wireless networks. This mobile ASR capability can be applied both as a user interface to the terminal device as well as a data input/output modality between the user and the remote application.

In order to achieve more accurate speech recognition applications over wireless channels, it is expected that various modes of operation of speech recognition will exist in mobile networks. ASR in mobiles can be characterized by the location where the recognition takes place; for example, recognition can take place in the terminal device, in a central server, or in a mixed or distributed scenario. The constraints that the network imposes on the bit rate of the transmitted signal, the limitations imposed by the computing capabilities of the device on the complexity of the signal processing front-end and the decoder, compounded with the potential exposure of the user to more intense and challenging environments, make the problem of ASR in mobile environments more susceptible to performance degradation than fixed network speech recognition application.

## 3.2 The idea of a centralized ASR

In this section we enumerate the characteristics and limitations that ASR applications encounter under the different modalities in which they can be deployed employing wireless digital communication links. As we mentioned before in this chapter, mobile speech recognition can be characterized according to the location where recognition takes place. This defines three principal modalities:
a) Network speech recognition
b) Terminal speech recognition
c) Distributed speech recognition

Each of these modalities can have very different and characteristic effects on the performance of ASR systems. We describe these modalities in more details:

### 3.2.1 Network speech recognition (NSR)

In this scenario, the recognizer is implemented in a location remote to the user so the speech signal has to be transmitted from the user's terminal to the recognition server through a wireless channel. Having a recognizer residing in a central server enables more sophisticated and elaborate ASR applications than those possible on terminal devices. In the server side computers usually have sufficient computation and memory resources to support large vocabulary continuous speech recognition tasks requiring complex acoustic and language models. For example, in a server can run more complex applications such as dialog-based systems, speech synthesis and databases queries.

A block diagram of a NSR system is shown in Figure 3.1. It consists of two distinct components, the mobile terminal and the server of the network. The only process occurs to the terminal is the coding of voice signal and the transmission of it over the wireless channel. The network server consists of three parts: the speech decoder, the feature extraction and the pattern recognition.

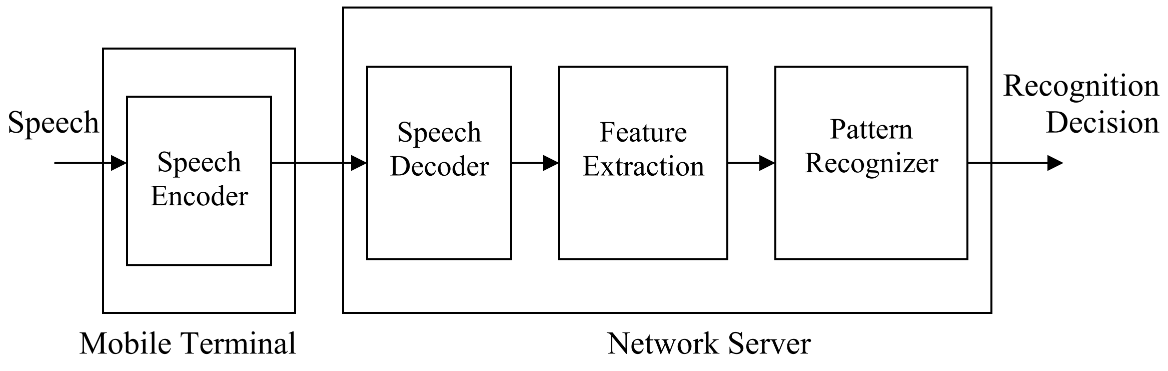

Fig. 3.1 Network Speech Recognition system

### 3.2.2 Mobile terminal speech recognition

In this case, the recognition performed in the user's terminal device. The speech signal does not travel through a wireless communication network, so it is unaffected by the transmission channel or source and compression algorithms. However, the main disadvantage is that there are constrains in computational, memory and power resources due to the cost-sensitive nature of the terminal device. These limitations give the possibility, only relatively simple recognition applications can be implemented on a mobile device. The most popular are hands free voice dialing, basic command and control application.

### 3.2.3 Distributed speech recognition (DSR)

In this client-server architecture the ASR application processing and computation routines are distributed between the terminal and the central ASR server. The speech utterance is acquired at the mobile device (client) and transmitted to a remote recognizer engine (server). With this scenario the client extracts the features from the voice signal. Then, only the extracted features are encoded and transmitted to the server. In the server side the speech recognizer operating on the decode feature data. The above process described in figure 3.2 that follows.

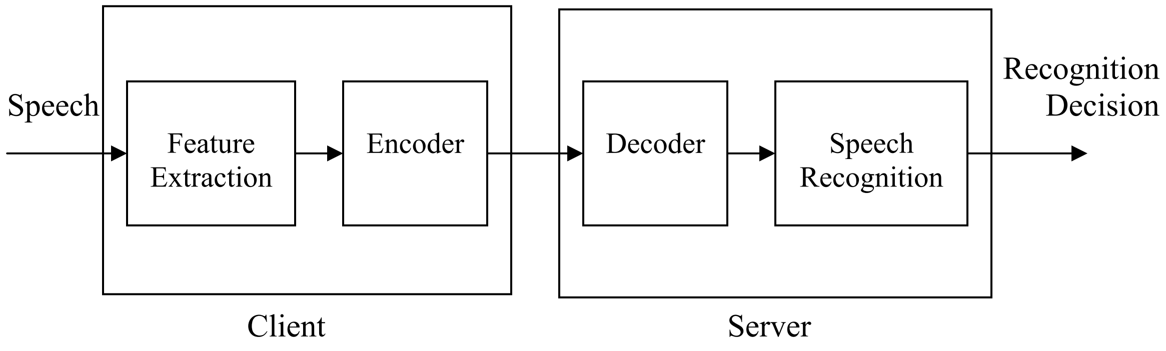

Fig. 3.2 Distributed Speech Recognition system

A system of this type can benefit from the advantages of the two modes of operation we have previously described: sophisticated systems can be implemented (as in NSR), while the features are computed and possibly normalized and compressed at the terminal level (as in mobile speech recognition). In DSR the computation is distributed between the server and the client, most commonly with the client performing the less complex feature extraction and the server hosting the complex pattern recognition. Another advantage of this approach is that updates of speech recognizers need to be done only on the server side, rather than having every device updated. This results in a much faster and cost-effective update procedure, and one that is effectively “invisible” to the client/customer. A major drawback of distributed versus terminal-based speech recognition is the fact that transmission errors can lead to degraded recognition performance. In this thesis we adopt the distributed approach to wireless ASR and we are going to search for a solution to the previous disadvantages.

### 3.2.4 Issues common to the mobile speech recognition modalities

The three mobile speech recognition modalities differ in broad terms of whether or not the transmitted coded speech is used for recognition, and whether the recognizer resides on the terminal device. In spite of these dissimilarities, these modalities share the following issues:

- Potential exposure to intense environmental noise: This problem is compounded by the fact that the additive noise might be highly non-stationary. Speakers may also modify, albeit unintentionally, their speech characteristics when speaking under intense noise conditions. Another problem common to hand-held devices that affects the quality of the speech signal is the physical placement of the terminal device. These types of distortions are the signal typical affects recognition substantially.

- Terminal equipment devices are cost sensitive: This implies that terminal devices will allow only limited computational capabilities and thus allow relatively limited front-end signal processing, feature extraction or recognition algorithms. Thus, the mobile network, terminal, and distributed speech recognition modalities will have to rely, at least in the immediate future on relatively simple signal processing, front-ends, and terminal recognizers, as well as inexpensive microphones.

## 3.3 Speech Signal Parameterization

The first part in the process of speech recognition is the parameterization of the speech signal. This occurs in the client side, inside the mobile phone. The speech waveform, which is the voice of the user, it is transformed into a sequence of parameter vectors. Each of the parameter vectors consist a frame. There are many techniques that can parameterize the speech spectrum quite well, but we are going to discuss only the filterbank analysis.

### 3.3.1 Filter bank analysis

The human ear resolves frequencies non-linearly across the audio spectrum and empirical evidence suggests that designing a front-end to operate in a similar non-linear manner improves recognition performance. Filterbank analysis provides a much more straightforward route to obtaining the desired non-linear frequency resolution. However, filterbank amplitudes are highly correlated and hence, the use of a cepstral transformation in this case is virtually mandatory if the data is to be used in a HMM based recognizer.

A cepstrum is a Fourier analysis of the logarithmic amplitude spectrum of the signal. If the log amplitude spectrum contains many regularly spaced harmonics, then the Fourier analysis of the spectrum will show a peak corresponding to the spacing between the harmonics: i.e the fundamental frequency. Effectively we are treating the signal spectrum as another signal, and then looking for periodicity in the spectrum itself. The cepstrum is so-called because it turns the spectrum inside-out. The x-axis of the spectrum has units of frequency and peaks in the cepstrum are called rahmonics. The cepstrum may be defined verbally:

The cepstrum is the FFT of the log of the FFT

The cepstrum can be seen as information about rate of change in the different spectrum bands. Usually the spectrum is first transformed using the *Mel Frequency* bands. The result is called the *Mel Frequency Cepstral Coefficients* or *MFCC*s. It is used for voice identification, pitch detection and much more. The cepstrum separates the energy resulting from vocal cord vibration from the "distorted" signal formed by the rest of vocal tract.

Mel scale is used to translate regular frequencies to a scale that is more appropriate for speech, since the human ear perceives sound in a nonlinear manner. In the case of speech recognition, a filter bank is applied of which the centre frequency of each bank is scaled according to the Mel scale. This scale takes into account the frequency resolution properties of the human ear. The inverse Fourier transform of the log output of this filter bank yields the MelFrequency Cepstrum Coefficients.

The MFCC parameterization follows common requirements imposed on a speech parameterization for speech recognition purposes. Its main features are aimed above all at:

- To capture an important information presented in a speech signal for recognition purposes
- To handle as little data as necessary
- To use any quick evaluation algorithm

Moreover the benefit of MFCC is also in their perceptually scaled frequency axis. The mel-scale offers higher frequency resolution on the lower frequencies in the same way as a sound is percept by the human auditory organ. In addition, the MFCCs offer through their cepstral nature abilities to model both pole and zeros. MFCCs are the parameterization of choice for many speech recognition applications. They give good discrimination and lend themselves to a number of manipulations.

To augment the spectral parameters derived from mel-filterbank analysis, an energy term can be appended in the parameterized vector. The energy is computed as the log of the signal energy, that is, for speech samples $\{s_n, n = 1, N\}$

$$E = \log \sum s_n^2 \tag{3.1}$$

In addition to, or in place of the log energy, we can use the 0'th cepstral parameter $C_0$. Also, it has been showed that the performance of a speech recognition system can be greatly enhanced by adding time derivatives to the basic static parameters. It is used to append to the parameterized vector, the first order regression coefficients which are also referred as *delta* coefficients, and the second order regression coefficients which are referred as *acceleration* coefficients. When delta and acceleration coefficients are requested, they are computed for all static parameters including energy if presented.

## 3.4 Quantization

The next step to the remote ASR process is the quantization of features vectors. Quantization refers to the process of approximating the continuous set of values in the MFCC files with a finite (preferably small) set of values. The input to a quantizer is the original data, and the output is always one among a finite number of levels. The quantizer is a function whose set of output values are discrete, and usually finite. Obviously, this is a process of approximation, and a good quantizer is one which represents the original signal with minimum loss or distortion.

There are two types of quantization, scalar quantization and vector quantization. In scalar quantization, each input symbol is treated separately in producing the output, while in vector quantization the input symbols are clubbed together in groups called vectors, and processed to give the output. This clubbing of data and treating them as a single unit, increases the optimality of the vector quantizer, but at the cost of increased computational complexity.

A quantizer can be specified by its input partitions and output levels. If the input range is divided into levels of equal spacing, then the quantizer is termed as a uniform quantizer, and if not, it is termed as a non-uniform quantizer. A uniform quantizer can be easily specified by its lower bound and the step size. Also, implementing a uniform quantizer is easier than a non-uniform quantizer.

## 3.5 Coding – Error protection

The coded speech gets organized into packets of bits (frames), modulated and transmitted to the cell/server site. In the server site the packages are demodulated and the speech is decoded. At this point it is possible that errors have been introduced in the bit-stream representing the signal due to interference noise in the transmission. Based on the type of analysis, the most sensitive bits are protected or coded more robustly (e.g., using more bits) than bits carrying less “perceptual important” information. Also, wireless standards permit the regeneration of distorted frames by extrapolation of speech codec parameters from adjacent undistorted frames exploiting the correlation or continuity that exists in the speech signal between adjacent frames. An Unequal Error Protection (UEP) scheme is introduced. The first MFCCs are protected more than the last ones, since they are more important to ASR performance in that they described the overall spectral shape. More details about the UEP scheme we will see in the chapter of implementation, when we describe the transmission system.

Whatever error protection scheme we use, it is certain that the signal in the receiver will not be the same, due to the noise which will be added. To overcome the detrimental effects of transmission errors, common error concealment strategies are used but the results are not satisfactory. Almost in all the wireless communications there are some basic error mitigation algorithms, such as CRC, consistency check, parameters interpolation and repetition of previous received frames.  These techniques may help to repair random bit errors but may fail for errors occurring in bursts, which are very likely in fading channels. Specially, for the speech recognition systems the techniques above cannot be used. So, we have to find another way to deal with these errors which can reduce a lot the performance of an ASR system. In the next chapter we will discuss a solution to this problem, which is the main aim of this thesis

### 3.5.1 Channel coding considerations

The emphasis in remote ASR is recognition accuracy and not play back. Recognition is made by accumulating feature vectors over time and by selecting the element in the dictionary that is most likely to have produced that sequence of observations. The nature of this task implies different criteria for design encoders than those used in speech coding applications.

For speech coding, frequent frame erasures due to poor channel conditions result in interruptions, buzzing and muting. For speech recognition where we accumulate observations over time, the situation can be different. Frame erasures reduce the number of observation vectors for all models, which may have little effect on recognition performance. Channel decoding errors, however, result in incorrect observation estimates, which in turn affect all state metrics, accumulated in the Viterbi recognizer and can be degrade significantly recognition performance. Speech recognition, as opposed to speech coding, can be more sensitive to channel errors than channel erasures.

In remote speech recognition, especially in wireless communication where fadings occur, the decoded feature is a function of the transmission channel characteristics. When channel characteristics degrade, one can no longer guarantee the reliability of the decoded feature. If the Viterbi algorithm operates without taking into account the decreased confidence in the feature, this can have a dramatic effect on speech recognition accuracy since maximum likelihood trellis searches accumulate metrics over time and errors in decoding a feature will propagate in the path metrics.

## 3.6 Summary

This chapter is an introduction to the automatic speech recognition process in mobile networks. We started by explain the three different modalities which there are nowadays and we continue giving more information about the

Distributed Speech Recognition, which is the modality we will use in this thesis. We saw in more details the speech parameterization which is one of the most important parts in the whole process. We said few words for the quantization and the coding of the speech signal and about error protection for automatic speech recognition over wireless channels. In the end of the chapter we talked a little for the channel coding considerations.

# Chapter 4

# Missing Feature Theory

## 4.1 Introduction

Speech recognition, in contrast with speech coding, can be more sensitive to channel errors than channel erasures. This leads us to the idea of Missing Feature theory which is a category of the Missing Data theory. The concept is that if we cannot correct the errors of transmission in the features, we can hide the wrong features from the speech recognition process.

There are two main problems that we have to solve if we want to implement the techniques of missing data theory to robust ASR:

a) To find an algorithm which can identify the reliable parts of the spectrum in the speech data.

b) To change the recognition process, so it will use only the reliable parts of the spectrum

Many researchers until now have been involved in the problem of identification of reliable data. For this task, we can use only the properties of the speech signal, the physics of sound and human voice. The main idea is to divide the feature vector $\boldsymbol{x}$ of speech data into a reliable (present) and unreliable (missing/noisy) components $\boldsymbol{x} = ( \boldsymbol{x}_{\mathrm{rel}} , \boldsymbol{x}_{\mathrm{unrel}} )$. The process can be visualized as if a "mask" $\boldsymbol{x}_{\mathrm{unrel}}$ has been placed over the feature vector $\boldsymbol{x}$ allowing us to see only the present component $\boldsymbol{x}_{\mathrm{rel}}$.

Later, the idea that dominates the researcher area was to try to deliver assessment of how probable is the proposed mask. It would be even better if it is possible to assess the probability of every possible mask. That means that there will be a probability $P(\mathbf{m})$ associated with each mask $\mathbf{m}$. In other words, every feature vector $\boldsymbol{x}$ will be correlated with another vector $\boldsymbol{w}$, with the same length, which will have the probabilities that show how "clean" each feature is. This technique was named as **Soft Feature Decoding** and it is the one we have used in this thesis.

## 4.2 Recognition techniques with Missing Features

The speech models needs to have their probability with only parts of the observation vector from the source they are modeling. Since there is no prior probability which features are going to missing, it is reasonable to assume that the models inferred during the training will have complete feature vectors, and will be adapted to handle the occlusion occurring during the recognition gracefully and in a principled manner.

They have been proposed three main speech recognition techniques which take into account the Missing Feature theory:

1. Data Imputation
2. Marginalize Unreliable Features
3. Exponential Feature Weights

We are going to see these techniques in more details in the rest of the chapter.

### 4.2.1 Data Imputation

The missing data is "filled in" (imputed) using the available knowledge in the form of the model $P(\boldsymbol{x}|\boldsymbol{s})$ where $\boldsymbol{s}$ is the state of the model and the present data $\boldsymbol{x}_{\mathrm{rel}}$. In order to do that, the conditional distribution of the missing data $\boldsymbol{x}_{\mathrm{unrel}}$ is needed first:

$$p(x_{unrel} \mid x_{rel}, s) = \frac{p(x_{rel}, x_{unrel} \mid s)}{p(x_{rel} \mid s)} = \frac{p(x_{rel}, x_{unrel} \mid s)}{\int p(x_{rel}, x_{unrel} \mid s) dx_{unrel}} \tag{4.1}$$

Then, a single value $\hat{x}_{unrel}$ from the conditional distribution has to be chosen according to some criterion and used as a "plug in" replacement for the missing values. The choice can be made by minimizing an error criterion. One of the most commonly used error measures is the mean square error (MSE).

Depending on the circumstances of application of the technique, other criteria may be used as well. For example, unreliable values could be replaced by the means for those components. A better approach is to use knowledge of reliable components in conjunction with the covariance of the distribution function that follows each value. Once $\hat{x}_{unrel}$ is computed, it is used as a plug-in value instead of $\boldsymbol{x}_{\mathrm{unrel}}$ and the "filled in" feature vector ( $\boldsymbol{x}_{\mathrm{rel}}$, $\hat{x}_{unrel}$ ) is used for further processing.

This technique has some important disadvantages. The idea to use the reliable data for computing the data that are missing, it does not always give good results. In wireless communications it is very common to have a whole frame with noise, so we cannot rely on this data for reproducing any data of this frame. Also, the processing time for this technique it is normally long enough for real time voice applications, so it is very difficult to be used in this kind of applications. The main advantage of this technique is that we do not need to make any change in the speech recognizer because the observation vector has not any changes.

### 4.2.2 Marginalize Unreliable Features

This error concealment strategy discards the transmitted features which are most probably erroneous and uses only the reliable ones for likelihood computations at the speech recognizer. A reduced feature vector is used based only on the components that have a high confidence level. In a hidden Markov model (HMM) based speech recognition system, the observed feature vectors are modeled by state-specific probability distributions $p(\boldsymbol{x}|\boldsymbol{s})$, where $\boldsymbol{x}$ is the feature vector and $\boldsymbol{s}$ is the state of the model. Usually a mixture of Gaussian densities is used for each state of the phoneme specific HMM. In this case, the reduced distribution for the reliable part of the feature vector is the marginal determined by integrating over all the unreliable components:

$$p(x_{rel} \mid s) = \int p(x \mid s) dx_{unrel} \tag{4.3}$$

where $\boldsymbol{x}_{\text{rel}}$, $\boldsymbol{x}_{\text{unrel}}$ are the reliable and unreliable components of the feature vector.

Using the distribution of only the reliable components for HMM likelihood computation is one of the first techniques for improving robustness of speech recognizers in noisy conditions, and it was labeled as the "Missing Feature theory". For speech recognition in noise, labeling unreliable spectrum features can be a challenging task, while in our application the reliability of each feature is provided by the channel decoder. With diagonal covariance Gaussian mixture model, which is the model we are using in this thesis, the reduced likelihood function can be easily calculated by dropping unreliable components from the full likelihood computation. This approach requires little modification in existing speech recognition systems.

The soft-feature decoding algorithm that computes likelihood using the marginal distribution over unreliable features is implemented as follows:

i. for energy and cepstrum features, if the first or the second bit of the decoder symbol has absolute likelihood ratio $|\Lambda(n)|$ below threshold $\Lambda_T$ it is labeled as "unreliable" and not used in the likelihood computation. $\Lambda(n)$ is computed from equation

$$\Lambda(n) = \ln \frac{prob(a(n)=1)}{prob(a(n)=0)} \tag{4.4}$$

ii. for "delta" and "delta-delta" features, if the first or the second bit of any of the symbols in the window used for the delta computation has $|\Lambda(n)| < \Lambda_T$, then, do not use the delta feature in the likelihood computation. Five and seven frame windows are used for delta and delta-delta computation, respectively.

The likelihood ratio threshold that minimizes recognition error can be computed from held out data.

This soft-feature decoding algorithm, which is labeled as **SoftFeatI,** has made an improvement to the results of speech recognition for wireless channels, but still, we can receive better results with a little change. The decision for reliable and unreliable data is so "hard". What if a more "soft" decision was made about the data that are affected by noise in the channel? The idea of continuous confidence values was born. In this case, continuous confidence values between 0 and 1 would be used and the distribution of each feature to the likelihood computation would be scaled by its confidence. In the following paragraph we discuss in more details this idea.

### 4.2.3 Exponential Feature Weights

An alternative soft feature decoding algorithm, labeled **SoftFeatII** applies exponential weights to each feature in the probability computation in decoder. Specially, assuming that the state observation probability density function (pdf) is a mixture of Gaussian pdfs with diagonal covariance the observation probability computation formula is modified as follows:

$$p(x|s) = \sum_{m=1}^{M} w_m \prod_{n=1}^{N} \left[ \frac{1}{\sqrt{2\pi}\sigma_{nm}} \exp\left( -\frac{(x_n - \mu_{nm})^2}{2\sigma_{nm}^2} \right) \right]^{f(C_n)} \qquad (4.5)$$

Where x is the feature vector, *N* is the size of the feature vector, *M* is the number of Gaussian mixtures per states and $w_m$,$\mu_m$,$\sigma_m$, are the mixture weight, mean and standard deviation, respectively, of the *m*th Gaussian for HMM state ***s***.

$\mathbf{C_n}$ is the confidence associated with the *n*th feature and $f(C_n)$ is a function of the confidence $\mathbf{C_n}$. Note that **C** is a function of time and is updated at the frame rate, as often as x is updated. Assuming that the confidence is normalized to a number between 0 and 1, then one possible form of the function

$$f(C) = \frac{a + C}{a + 1} \qquad (4.6)$$

where *a* is a smoothing constant that is experimentally determined so that error is minimized on a held-out data set. For very large values of *a*, all features are more or less weighted equally, confidence **C** is practically ignored, while for very small of *a*, only features with high confidence $(C_n \approx 1)$

are considered in the observation probability computation. All other aspects of the decoding process, apart from the feature weighting in the state observation probability computation, remain unchanged.

To compute the feature confidence the symbol bit probabilities computed at the channel decoder are translated into feature confidence scores. In our case, the mapping from feature value to sequence of bits (quantization) is nonlinear. The feature confidence score is thus computed numerically as follows:

- Assuming that the most probable symbol obtained at the channel decoder is $\hat{S}$ , and P($S_k$) is the probability that the decoder produces the *k*th symbol and also assuming independence among bits, P(S) is simply computed as the product of the decoder probabilities for each of the bits in the symbol. The expected mean square error *E* for that features is computed as:

$$E = \sum_k P(S_k)\left[Q^{-1}(\hat{S}) - Q^{-1}(S_k)\right]^2 \qquad (4.7)$$

  where is the inverse of the quantization mapping. The expected mean square error is normalized by the feature variance and subtracted from 1 to produce the feature confidence C.

## 4.3 Summary

In this chapter we introduced the Missing Feature Theory. We discussed how this idea was born and the two major problems we have to solve. The first one is to find an algorithm which can identify the reliable parts of the spectrum in the speech data. After, we have to change the recognition process, so it will use only the reliable parts of the spectrum. Then we saw three basic techniques for soft feature decoding with their advantages and disadvantages.

# Chapter 5

# Implementation

In this chapter we describe the whole process of speech recognition over wireless channel. We start from what happens into the mobile terminal and we reach until speech recognition results which come out to the server side. Our work is based on the procedure which is followed in [2].

## 5.1 Mobile terminal

In a DSR system, three are the main functions which are executed in the mobile terminal:

1. Parameterization of the speech signal
2. Quantization of the parameterized signal
3. Transmission

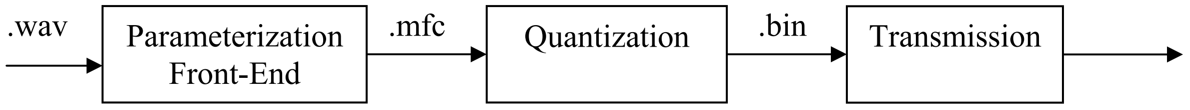

Fig. 5.1: Functions occur in mobile terminal

### 5.1.1 Parameterization of the speech signal

In the mobile, the voice of the user is recorded and is produced a WAV file. In order to receive the cepstral features, we pass the WAV file from a Front-End feature extraction algorithm. Using cepstral features for signal parameterization is a compact and robust speech representation, well suited for distance based classifiers. In this thesis, we calculate cepstral features using a Mel-filter bank analysis. The acoustic features for speech recognition used in this thesis are the 12 cepstral coefficients, $C_1$, $C_2$,…,$C_{12}$, calculated every 10ms based on MFC analysis, and in place of the signal energy, we use the 0'th cepstral parameter $C_0$. For the above we used the HCopy tool of HTK toolkit.

Until now we have a 13-dimensional feature vector every 10ms. The acoustic input to the ASR system will be 39-dimensional feature vector, because as we had said in paragraph 3.3.1 the performance of a speech recognition system can be greatly enhanced by adding time derivatives to the basic static parameters. So, *delta* and *delta-delta* coefficients will be generated in server side, just before enter the speech recognition system. The next step is the quantization of the feature vectors.

### 5.1.2 Quantization of the parameterized signal

In order to transmit from wireless handset to network based recognition server, all 13 features are scalar quantized. A simple non-uniform quantizer is used in this thesis, to determine the quantization cells. The quantizer uses the empirical distribution function as the companding function, so that samples are uniformly distributed in quantization cells. The algorithm is a simple non-iterative approximation to Lloyd's algorithm, which does not necessarily minimize quantization noise. Better performance may be achieved using a k-means type of algorithm applied to the entire feature vector (vector quantization). The coding algorithm that we use is valid for different quantization scheme, so quantization scheme and coding can be implemented separately.

The bit allocation scheme used in all experiments in this thesis is shown in Table I. Six bits were allocated for each of zero cepstrum coefficient ($c_0$) and the most significant cepstrum ($c_1,\ldots,c_5$) features, while four bits were assigned to each if $c_6,\ldots,c_{11}$. Empirical tests showed no significant performance degradation for the evaluated task by replacing the last (12th) cepstral coefficient $c_{12}$ with its fixed precalculated mean. This means that there is not much information relevant to the speech recognition process in $c_{12}$ and, thus, no bits were allocated to $c_{12}$. At the receiver, $c_{12}$ is simply restored to its fixed precalculated average value, and the standard 13-dimensional feature vector is used for the next steps of recognition process. Note that our

ASR unit employs $c_{12}$, therefore, the value of $c_{12}$ is restored at the receiver. The total number of bits allocation scheme is 60bits/10-ms frame. This requires an uncoded data rate of 6kb/s to be transmitted over the wireless channel.

| Feature Component | $c_0$, $c_1$,$c_2$,$c_3$ $c_4$,$c_5$ | $c_6$,$c_7$,$c_8$ $c_9$,$c_{10}$,$c_{11}$ | $c_{12}$ |
|---|---|---|---|
| Bits per Feature | 6 | 4 | 0 |

Table 1: Bits allocation for different feature components

### 5.1.3 Transmission system

The data rate for the quantized speech parameters is 6kb/s. With the addition of error protection bits the coded data rate is 9.6kb/s. This is one of the data rates used in North American cellular standard IS-95. The channel overhead introduced at 9.6kb/s-data rate is reasonable and if lower coded bit rates are required trellis coded modulation schemes with higher order modulation may be considered. The modulation format of the coded signal does not have an important effect to the whole process of remote ASR, so we chose the Binary Phase Shift Keying (BPSK) to simplify the demodulation process.

In slow fading channels, it is useful to have a large interleaver to improve the system performance. However, large interleavers introduce delays and this may not be desirable in some real-time applications. To provide better time diversity and improve performance in slow fading channels coded data is interleaved over 8 speech frames or 80ms. The total interleaving and deinterleaving delay associated with this is 160ms and this time can be tolerated in wireless speech recognition applications.

The 12 parameters that have to be protected in a 10-ms speech frame are the $c_0(n)$ in place of energy and the 11 cepstral coefficients $c_1(n)$, $c_2(n)$,**...**, $c_{11}(n)$ where n denotes the speech frame index. Obviously, the most significant bits of the above parameters should have better channel error protection. In addition, it was determined experimentally that the zero cepstral coefficient $c_0(n)$ is the most sensitive to quantization as well as random transmission error, followed by $c_1$(n),..., $c_5$(n) and then $c_6$(n),..., $c_{11}$(n). The channel coded bit rate is 9.6kb/s, therefore, the total coded bits in a 80 ms channel encoded frame is 768.

### 5.1.3.1 Unequal Error Protection scheme

The UEP scheme that we used in this thesis is based on [6]. It consists of three levels of channel error protection denoted by L1, L2 and L3. Furthermore, to emphasize the significance of the most important bits of L1, this is separated to two levels: L1_1 and L1_2. The assignment of the bits for different UEP levels is shown to Table 2. In this notation, $c_0^0(n), c_0^1(n)$,... denote the bits of $c_0$(n) in decreasing order of significance. As seen from the table, the number of bits per speech frame in L1, L2 and L3 are 13, 24 and 23, respectively. In this case, L1_1 contains the bits that are determined to be the most important 7 bits and L1_2 contains the next 6 important bits. We employ a rate 1/2 memory 8 code on L1 level bits and thus, the total number of coded bits for 8 speech frames for L1 level is 208.

The L2 level contains the next 24 important bits and the total number of uncoded L2 bits for 8 speech frames includes the 8-bits tail is 200. In order to maintain a total bit budget of 768 coded bits, we puncture 24 bits of the 400 coded bits to give 376 bits for L2. The least important bits are in L3 and these 184 bits are transmitted without any channel coding.

Channel coding is done so that L1_1 level bits are followed by L1_2 and then L2. Note that, because of the puncturing of coded L2 bits and since the coded L1 bits are not terminated, those bits of L1_2 that are separated from L2 level by less than a decoding depth of the channel code will not be subjected to the usual rate 1/2 mother code. At the channel encoder input the L1_2 level bits for the 8 speech frames n,(n+1),...,(n+7) are arranged in the following manner: $\mathrm{c}_0^2(\mathrm{n}),\mathrm{c}_0^2(\mathrm{n+1}),...,\mathrm{c}_0^2(\mathrm{n+7})$ ; $c_1^1(n), c_1^1(n+1), ..., c_1^1(n+7)$ ;...; $c_5^1(n), c_5^1(n+1), ..., c_5^1(n+7)$. As stated previously, we have determined that the coefficients $c_1$(n) are more significant than $c_5$(n) and, therefore, this bit arrangement will assign bits of lower significance toward the end of the L1_2 frame which will be subjected to a less powerful code than the usual rate 1/2 mother code.

| Level | Speech Bits | Error Protection |
|---|---|---|
| L1_1 | $c_0^0(n), c_0^1(n), c_1^0(n), c_2^0(n), c_3^0(n), c_4^0(n), c_5^0(n)$ | rate ½ conv. code |
| L1_2 | $c_0^2(n), c_1^1(n), c_2^1(n), c_3^1(n), c_4^1(n), c_5^1(n)$ | rate ½ conv. code |
| L2 | $c_0^3(n), c_0^4(n), c_1^2(n), c_1^3(n), c_2^2(n), c_2^3(n), \ldots$ $c_6^0(n), c_6^1(n), c_7^0(n), c_7^1(n), \ldots c_{11}^0(n), c_{11}^1(n)$ | rate ½ conv. code and puncturing |
| L3 | $c_0^5(n), c_1^4(n), c_1^5(n), \ldots, c_5^4(n), c_5^5(n),$ $c_6^2(n), c_6^3(n), c_7^2(n), c_7^3(n), \ldots, c_{11}^2(n), c_{11}^3(n)$ | no code |

Table 2: Speech bit assignment for different UEP levels

## 5.2 Server side

The server side consists of two main modules:

1) Channel decoder
2) Speech Recognition unit

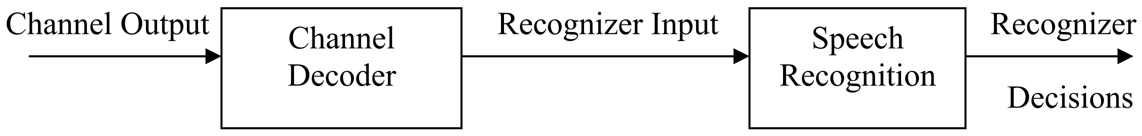

Fig. 5.2: Server side image

### 5.2.1 Channel Decoder

The channel decoder has two functions:

1. To recover the MFCC files
2. To implement Soft Feature Decoding

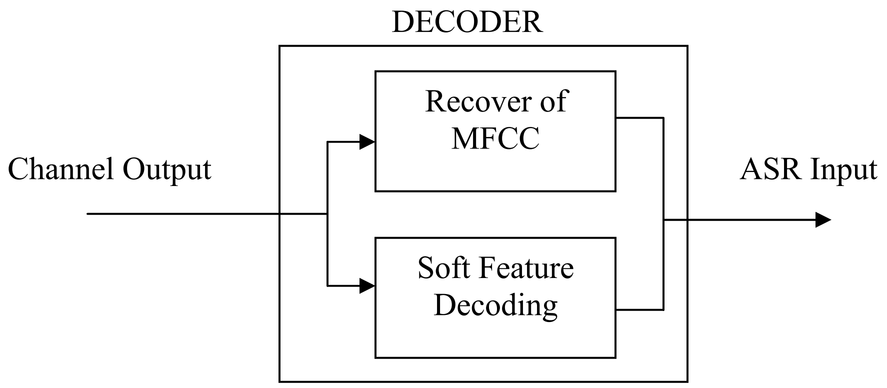

Fig. 5.3: Decoder block

The first function is the dequantization of the transmitted signal. In this stage we will recover the MFCC files which we have sent into the channel. Except from the quantization error, now we have also the channel noise. This source of errors is more important and our aim in this thesis is to minimize its effect to the speech recognition system. To achieve this, we introduced the Soft Feature Decoding procedure. We use an algorithm that maximizes the *a posteriori* probability (MAP) [5] which gives the *a posteriori* probability of each decoding bit. More specific, bit error probabilities are estimated using the Max-Log-MAP algorithm. The ASR unit utilizes this information to give improved performance gains.

The channel output except from the transmitted signal has also other information which concerns the transmitted bits. This data is input to the Soft Feature decoding function, which in this thesis, it is a simple file written in C programming language. The information we have for each bit, can tell us the probability to have in the receiver a "clean" bit or not. Furthermore, knowing the allocation bits for the features in the transmission system, we can obtain the information of how "clean" is each feature we receiver in the decoder.

The work of the Soft Feature decoder is to calculate the probability of each received feature to be clean from channel noise. So, the output of this part of the decoder is a continuous value from 0 to 1. These values will be used in place of confidence function $f(C_n)$ which we have introduced in 4.2.3 paragraph, when we explained the Exponential Feature Weights function. When the weight is 1, it means that the feature we have received in server side is "clean" and it will be used in recognition process by the speech recognizer. If the weight value is 0, then this feature has so much noise that it will not be participated in the ASR process. The rest of the values show a percentage of how correctly has been received each feature.

The Soft Feature Decoding algorithm uses the information from the Most Significant Bits (MSB) of each feature, in order to produce the confidence level of each received feature. For the $C_0$ coefficient utilizes the 5 MSB to compute the weight, for the coefficients $C_1$,$C_2$,$C_3$,$C_4$ and $C_5$ uses the 4 MSB and for the rest of the coefficients only the first and the second MSB. If the MSB in the feature is "clean" from noisy, this means that, it is more possibly this feature has been received "clean" in the server side. So, to show the importance of the MSB we put a weight with which we multiply the confidence level of each bit of the feature. The weight for the MSB is 1, 0.5 for the second MSB and so on. To calculate the confidence level of the feature we sum the confidence level of the MSB. More details about this algorithm there are in the Appendix part.

The output of the Soft Feature function is organized into vectors. Each vector of weights is grouping together with the corresponding feature frame. In our implementation, we computed the confidence level of 39 features per frame which are 13 MFC coefficients, 13 delta coefficients and 13 delta-delta coefficients. In this step there is a miss match in the length of feature vector and weight vector. This is going to be arranged by the ASR system, where we will compute the delta and delta-delta coefficients.

### 5.2.2 Speech Recognizer

First we will say few words for the speech recognition engine we used. The HTK toolkit, version 3.2.1, was used in all the experiments in this thesis. The TIMIT corpus was used for training and testing purposes. The phoneme HMMs models use 3 states with a mixture of Gaussian with 16 components and they were trained using Baum-Welch assuming a diagonal covariance matrix. The training was done with the clean data and the testing with the files which have been passed the implementation process we have described earlier. The Viterbi algorithm was used for the testing process and all the recognition results are reported as phoneme percentage correct. In order to use the outputs of Soft Feature Decoding algorithm, we made some changes in the HVite tool of HTK and in the calculation of output probability. All the details for these changes, there are in Appendix part.

The last thing we have to do in the speech frames, before entering the speech recognizer, is the augmentation of feature elements from 13 to 39. So, we use once again the HCopy tool of HTK toolkit and we add the 1st and the 2nd order time derivatives. Now feature vector has the same length with weight vector.

## 5.3 Summary

In this chapter we saw the implementation, of all the parts of this thesis. We started from the functions occur in mobile terminal. The features extraction is the first action. It follows the quantization of the speech signal and in the end the transmission system module. We saw in details the Unequal Error Protection scheme we use in our application. The new in this thesis is the implementation of a Soft Feature Decoding function, which generates an additional value giving the confidence of correctly decoding each received feature. We saw also the small changes we made in speech recognition unit, in order to use the output of Soft Feature algorithm.

# Chapter 6

# Results and Evaluation

In this chapter we will see the speech recognition accuracy we achieve with the implementation we have proposed in the previous chapter. In the first part, we will see the speech recognition results and then we will have the evaluation.

## 6.1 Speech Recognition Results

The HTK toolkit has one tool for recognition (HVite) and one for evaluation (HResults). HResults compares the transcriptions output by HVite with the original reference transcriptions and then outputs various statistics. HResults matches each of the recognized and reference label sequences by performing an optimal string match using dynamic programming. The percentage correct is computed by the HResults from the following equation:

$$\text{Percentage Correct} = \frac{N - D - S}{N} \times 100\% \qquad (6.1)$$

where N is the total number of labels, D is the number of deletion errors and S in the number of substitution errors.

TIMIT

The speech database we used for the experiments was DARPA TIMIT database. The DARPA TIMIT speech database was designed to provide acoustic phonetic speech data for the development and evaluation of automatic speech recognition systems. It consists of utterances of 630 speakers that represent the major dialects of American English. The training part was made with 3696 sentences and the testing with 1344 sentences. The whole process of speech recognition is with phonemes. The phoneme list consists of 47 phonemes.

The channel simulator we used has three main parameters. The first is the speed of the mobile terminal, the second is the Signal-to-Noise Ratio (SNR) and the third is the noise distribution function. We wanted to simulate a fading channel, so we used a Rayleigh distribution for noise in the channel which is very close to the real conditions of noise in a wireless channel.

For a Rayleigh fading channel we executed experiments for different SNR values and speeds of the mobile. We chose the speed of 10, 50 and 100 km/h and the SNR of 0dB, 1.5dB, 3dB and 5dB. First we computed the speech recognition performance without using in the speech recognition unit the outputs of the Soft Feature decoding. Then we executed again the speech recognition, but this time we took into account the outputs of Soft Feature algorithm. The first implementation we labeled it, as a Baseline Decoding scheme and the other as Soft Feature Decoding scheme. The results for both of these schemes are listed below. The clean baseline result is 60.40%.

| Speed [km/h] | Decoding Scheme | 5dB | 3dB | 1.5dB | 0dB |
|---|---|---|---|---|---|
| 10 | **Baseline** | 56.53 | 53.00 | 48.97 | 43.01 |
|  | **Soft Feature** | 56.41 | 52.98 | 48.85 | 41.67 |
| 50 | **Baseline** | 58.54 | 56.71 | 52.84 | 45.97 |
|  | **Soft Feature** | 58.56 | 56.71 | 52.74 | 45.20 |
| 100 | **Baseline** | 59.24 | 58.11 | 55.15 | 47.80 |
|  | **Soft Feature** | 59.24 | 58.11 | 55.16 | 47.32 |

Table 3: TIMIT database. Phoneme level recognition accuracy with and without Soft Feature decoding for a Rayleigh fading channel with different speeds and SNRs.

As we can see, the results we take with the baseline scheme are the ones we expected. When we have the same SNR the results are better for the mobile terminals which are moving with higher speed. This is because the mobiles are passing through the “noise hole” for a shorter time and there are affected less from the noise. But, the use of Soft Feature algorithm does not improve at all the speech recognition performance. This comes in contrary with the theory we have discussed in previous chapters where we were talking for significant improvement.

## AURORA 2

The results we had with the TIMIT database were not the ones we expected. We wanted to see an improvement when we applied the Soft Feature Decoding algorithm and we used the confidence level of received features in final recognition. In TIMIT database the recognition is done in the level of phonemes and we believe that this is the main reason why the results are not following the theory of Missing Feature. So, changing just the database and keeping the rest of the implementation process exactly the same, we expect to have better results.

We choose the database of AURORA 2. Here the recognition is done in word level instead of phonemes as we had before. The word dictionary we have now is the 10 digits (zero, one, two…nine). The testing part of the speech recognition system was done with 3712 sentences. Each sentence has a different number of digits. The digits are spoken in continuous way but can also have silence pause between them.

We use again MFC coefficients and in general the whole process is the same as in the previous example with TIMIT database. We executed the experiments for a Rayleigh fading channel with SNR and speed parameters exactly the same as before. The results are listed in the following table. We can see that now there is a significant improvement using the outputs of the Soft Feature Decoding. The clean baseline result is 99.31%.

| Speed [km/h] | Decoding Scheme | 5dB | 3dB | 1.5dB | 0dB |
|---|---|---|---|---|---|
| 10 | **Baseline** | **96.23** | **85.26** | **84.26** | **71.71** |
|  | **Soft Feature** | **98.10** | **89.16** | **92.70** | **86.02** |
| 50 | **Baseline** | **98.91** | **97.55** | **93.53** | **82.08** |
|  | **Soft Feature** | **99.17** | **98.69** | **97.04** | **92.66** |
| 100 | **Baseline** | **99.21** | **98.61** | **96.68** | **87.07** |
|  | **Soft Feature** | **99.19** | **98.93** | **98.16** | **94.14** |

Table 5: AURORA2 database. Word level recognition accuracy with and without Soft Feature decoding for a Rayleigh fading channel with different speeds and SNRs.

The above results are following the theory of Missing Feature we have discussed in chapter four. Now, we can see the significant improvements over the baseline for Soft Feature algorithm. In general, the improvement is independent of the mobile speed and higher improvement is shown in lower SNR channel. The reduction of word error rate is around 50% for all the occasions, except for channels with very high SNR and speed. The improvement is substantial and is equivalent to enhancing the channel SNR by about 1.5db and even in some occasions 2db.

## 6.2 Evaluation

The system we implemented in this thesis composed of many independent parts. To be sure that there is nowhere an error, we made many testing. We began from HTK changes w made in the testing part of recognition and more specific in HVite tool. Then we checked the outputs of Soft Feature algorithm.

HTK changes

A big part of this thesis was to make the necessary changes to the HTK toolkit so it can use in the final recognition process the outputs of Soft Feature algorithm. This new function correlates each feature with a weight which comes out from the soft feature decoding. When we completed this change and before proceed to the next step of implementation, we checked that the new function works correctly.

The first check was to fill all the weights with value equal to one. That means, all the features are correct and they participate with 100% in recognition process. The result we expect is exactly the same as the clean baseline result and this is exactly the same 60.40% final recognition performance. Then, we took the clean speech data and we put noise in some feature. In the same time we computed the weights for these features. So in the beginning, we chose by luck three and five features per frame, we changed their coefficients values in half and we put the corresponding weights equal to 0.5. We executed the baseline and soft feature recognition and we received the presumed results. Also, instead of choosing by luck the coefficient, we can put noise in specific features in every frame. So, we decided to check the reactions when the noise is in the feature of energy and in the first MFC coefficient. The results of the above test are listed below:

| Noise | Baseline (%) | Soft Feature Decoding (%) |
|---|---|---|
| 3features/frame | **58.97** | **59.49** |
| 5features/frame | **57.58** | **58.57** |
| Energy | **22.69** | **40.13** |
| C1 | **53.11** | **55.99** |

Table 6: Recognition results for the testing of HTK changes

As we can see, Soft Feature decoding works correctly. In all the occasions the recognition results are higher than the baseline. Also, we can conclude that the energy coefficient is the most important in speech recognition and we have to be very careful when we choose a weight for it. Being sure that the changes we accomplished in the testing part of HTK are working without any bugs, we moved on, in the implementation of the speech recognition system over wireless channel.

## Weights Computation

The next important check we had to make is for the outputs of Soft Feature Decoder. We need to have a measure of correctness of the confidence level which is computed for each feature. So, first we computed the difference between the values of coefficients from the data we have received in server side and the clear data. This difference shows how much noise was added in the transmitted features. Now, we try to compute the correlation between this difference and the weights which are produced by Soft Feature algorithm. The perfect scenario is to have correlation value equal to one, but this never happens to real world application. So, the correlation we have in our application is 0.5. This correlation between noise and weight is a realistic value and can give the improvement in speech recognition we have talked before.

We examined more the results of Soft Feature algorithm to be sure that there is nowhere an error in this part of the system. Unfortunately, we observed that in some occasions the weights which are produced have a wrong value. That means, in a correctly received feature the soft feature algorithm puts the value of the corresponding weight no close to one and the contrary; it puts to a wrongly received feature a weight no near to zero. Probably, this problem can be overcome by changing some parts in the soft feature algorithm. This is one of the most important future works of this thesis. More research is underway to tune the parameters of Soft Feature algorithm on held-out data to further improve performance.

## Pseudo Soft Feature algorithm

The bad results we received using Soft Feature algorithm in the TIMIT database made us to research more this case. We wondered, what the results would be if we had a perfect soft feature algorithm? A perfect algorithm in our case is the one which chooses correctly the weights for all the features without making any mistake. In such an algorithm the correlation between noisy features and weights has to be very close to one.

In order to achieve correlation near to one, we compared the clean data with the one which has passed through the channel. We computed the difference between the clean and the noisy feature and then taking into account this difference we set the confidence level of each feature. The correlation we achieved with this technique was a little higher than 0.9. Now, we expected to have a significant improvement in speech recognition performance but there is not.

| | Baseline | Soft Feature (Correlation:50%) | Pseudo-Soft Feature (Correlation:90%) |
|---|---|---|---|
| 3db – 50km/h | **56.71** | **56.71** | **57.10** |
| 1,5db – 10km/h | 48.97 | 48.85 | **50.08** |

Table 7: Effect of correlation between noise and weight in final recognition results using TIMIT database

The speech recognition accuracy we reached with correlation near to one it does not have significant difference from the baseline. The improvement in channels with a lot of noise is less than two percent higher. This test tells us that the Soft Feature algorithm does not improve significant the speech recognition in systems which work with phonemes instead of word.

# Chapter 7

# Conclusions and Future Work

## 7.1 Conclusions

The aim of this thesis was to apply a Soft Feature Decoding algorithm for speech recognition purposes over wireless channels. First, we made the necessary changes to the speech recognizer unit. Then we worked in the process of a distributed speech recognition system. We separated the whole process into two main parts, the part of the mobile terminal and the one of the server side. In the first one, there are the feature extraction function, the quantization and the transmission system. In the second part, there are the channel decoder and the speech recognizer.

In the beginning of this thesis, we worked with the speech recognizer unit, which in our case is HTK toolkit. We managed to change the Testing part of the recognizer, so it will be possibly to use the information that is generated from any Soft Feature Decoding algorithm. So, the speech recognition process is independent from the Soft Feature Decoding algorithm. The only constraint we have is that the soft feature algorithm to produce a weight value for every feature component which is used for the speech recognition. That means, the weight vector and the feature vector that enter the speech recognizer have the same length.

Then, we worked on the functions in the mobile terminal. The first of them is the speech parameterization. We used a Front-End feature extraction algorithm and we obtained the parameterized signal. The features which we worked with were the 12 MFC coefficients and the 0’th coefficient instead of signal energy. The next step is the quantization of the signal. We used a scalar non-uniform quantizer, which does not necessarily minimize quantization noise. Final we transmitted the quantized signal, using an Unequal Error Protection scheme, which protects the first 6 MFCCs more than the rests. This is logical because the first coefficients are more important for the speech recognition, they are carrying with them more “perceptual important” information so they need better protection.

The last component of the system is the server side functions. Here, there are placed the channel decoder and the speech recognition unit. In our work, the channel decoder except from the decoding of the speech signal has a more important work to do. A part of the decoder is the Soft Feature Decoder function which generates an additional value giving the confidence of correctly decoding each received feature from the server. This is a way to minimize the effect of the channel noise. The weights which are produced from the Soft Feature Decoder are used from the speech recognizer to obtain better performance.

As a conclusion to this thesis, we managed to apply a Soft Feature Decoding algorithm for speech recognition purposes over wireless channels. The algorithm employs the bit probabilities at the channel decoder to assign confidence on the ASR features. This information is used during decoding to improve speech recognition performance. The proposed algorithm can enhance speech recognition performance for any wireless channel, ASR feature encoding scheme, channel protection scheme, cepstral coefficients and speech recognizers.

## 7.2 Future Work

The whole system of speech recognition over wireless channel has many parts which can be developed separately. So, there are many things we can work with them, for the improvement of the final results.

Into the mobile terminal we can use different algorithms for feature extraction from the voice signal. Parameterization of the signal is an important procedure for the speech recognition and many researchers are working in this area. But in the case of wireless communications, it is also very important the quantization of the signal. The channel constrains in transmission rates define the bit allocation scheme for the transmitted features, and unfortunately we cannot have significant changes in this part. But, we can do a lot of work in the quantization algorithm. In this thesis it was not an aim to find the best quantization algorithm, so we use a simple scalar and non-uniform one. For future work we can try to change the quantization algorithm and use, for example, vector quantizer instead of scalar and uniform instead of non-uniform. Also, we can choose another error protection scheme for the transmission over the channel.

We can see this thesis as a component of a bigger project. With this point of view we can apply the same Soft Feature Decoding algorithm in similar systems. Thus, we will have better evaluation results. We can make changes in some part of the whole process and see the difference in the final results. For example we can use LPC coefficient for voice parameterization and a different feature extraction algorithm. Also, a very good idea it would be the speech recognizer to make the recognition in word level and not in phonemes. In the last occasion, we expect much better results using the Soft Feature Decoding algorithm.

An important future task which should be done is to make the necessary changes to other speech recognizers so they can accept the outputs of the Soft Feature Decoder and to use them in the testing part of the recognition. Also, it will be a very good idea to use another database for the training and the testing parts. With the above experiments we will have a more general image of the application of Soft Feature Decoding in speech recognition over wireless channels.

# Appendix

## Analysis of output probability function DOutP

Function DOutP is defined in the *HModel.c* file in HTKLib folder. HTKLib folder includes all the files which are used from the tools of HTK. The programming language that is used for all the files is C. DOutP is placed in the part of *HModel.c* for the calculations of the Output Probabilities. It computes the probability, for the Diagonal case, of an observation vector *x* in the given mixture.

### Declaration

```
static LogFloat DOutP (Vector x, int vecSize, MixPDF *mp)
    {
          int i;
          float sum, xmm;

          sum = mp->gConst;
          for (i=1; i<=vecSize; i++)
                {
                      xmm=x[i] - mp->mean[i];
                      sum += xmm*xmm*mp->cov.var[i];
                }
          return -0.5*sum;
    }
```

### Computation of probability

The general type which computes the probability for the normal density is:

$$p(x)=\frac{1}{\sqrt{(2\pi)^d\,|\Sigma|}}\exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right] \tag{1}$$

where d is the length of vector **x, μ** is the mean vector and **Σ** is the covariance matrix dxd. For the Diagonal case the equation (1) is converted to

$$p(x)=\frac{1}{\prod_{i=1}^{d}\sqrt{2\pi\sigma^2{}_i}}\exp\left[-\frac{1}{2}\left[\frac{(x_i-\mu_i)}{\sigma_i}\right]^2\right] \tag{2}$$

and if we express the above equation in *ln*

$$\ln(p(x))=-\frac{d}{2}\ln(2\pi)-\frac{d}{2}\ln(\sigma_i^2)-\frac{1}{2}\left[\frac{x_i-\mu_i}{\sigma_i}\right]^2 \tag{3}$$

$$\ln(p(x))=-\frac{1}{2}\sum_{i=1}^{d}\left[\ln(2\pi)+\ln(\sigma_i^2)+\left[\frac{x_i-\mu_i}{\sigma_i}\right]^2\right] \tag{4}$$

The first two components are calculated inside the *FixDiagGConst* function. So

*mp->gConst* = d ln2π + d ln$\boldsymbol{\sigma}_i^2$

in *DOutP* compute the rest of the probability $\left[\frac{x_i - \mu_i}{\sigma_i}\right]^2$ and we sum all the components. In the end we multiply the final sum with -0.5 to take the ln(p(x)).

Modified function DOutPWGT

In this thesis we changed the way of computation of output probability, so we modified the DOutP function. The new function DOutPWGT uses the weights which have been calculated by the Soft Feature Decoding algorithm. So with the addition of weights in output probability function, we have to compute the following equation:

$$\ln(p(x)) = -\frac{1}{2}\sum_{i=1}^{d} w_i \left[ \ln(2\pi) + \ln(\sigma_i^2) + \left[\frac{x_i - \mu_i}{\sigma_i}\right]^2 \right] \tag{5}$$

```
static LogFloat DOutPWGT(Vector x,Vector wgt, int
                         vecSize, MixPDF *mp)
{  int i;
   float sum1,sum2,sum,xmm;
   LogFloat z;
   Vector v;

   v = mp->cov.var;
   sum = 0;
   for (i=1;i<=vecSize;i++)
      {
            z = (v[i]<=MINLARG)?LZERO:log(v[i]);
            sum1 = log(TPI)+ z;
            xmm = x[i] - mp->mean[i];
            sum2 = xmm*xmm/mp->cov.var[i];
            sum +=wgt[i]* (sum1 + sum2);
      }
   return -0.5*sum;
}
```

## Soft Feature Decoding Algorithm

Each feature is represented with a number of bits in the transmission channel. The coefficients $C_0$ until $C_5$ have 6 bits for the transmission and the rest of the coefficients have 4 bits. The algorithm which computes the confidence level of each feature is the following:

```
For the MSBs of each feature
{
    $prob = \frac{BIT}{10} - 10$

    $prob = \frac{\exp(prob)}{1+\exp(prob)}$

    If   prob >= 0.5  // Transmitted bit is 1
        $dd += (1 - prob) * MSBweight$
    else     // Transmitted bit is 0
        $dd += prob * MSBweight$
}

Return   WeightFactor * dd

If  dd > 1              weight  =  0
If dd <= 1              weight = 1 - dd
```

The BIT variable has values from 0 to 200. 0 indicates the transmitted bit is 0 and 200 indicates the transmitted bit is 1. The MSBweight variable shows the importance of the transmitted bits for each feature. So for the MSB of every feature the MSBweight has the value 1, the second MSB has the value 0.5, the third MSB has the value 0.25 and so on. The variable WeightFactor is an integer which multiplies the confidence level of each feature. The final confidence level is stored in the variable weight and this is the output of the Soft Feature algorithm.

# Bibliography

[1] The HTK book, version 3.2.1, December 2002.

[2] A. Potamianos, and V. Weerackoky, ”*Soft-feature decoding forspeech recognition over wireless channels”*, presented at the Int. Conf. Acoust., Speech and Signal Processing, Salt Lake City, Utah, May 2001

[3] M. Cooke, P. Green, L. Josifovski, and A. Vizinho, *“Robust ASR with Unreliable Data and Minimal Assumption”*, in Proceeding, Robust Methods for Speech Recognition in Adverse Conditions, (Tampere, Filand), pp. 195-198, 1999.

[4] G. N. Ramaswancy, and P. S. Gopalakrishman, *“Compression of Acoustic Feature for Speech Recognition in Network Enviroments”*, in 1998 International Conference on Acoustic, Speech abd Signal Processing, (Seattle, Washington), 1998.

[5] P. Robertson, P. Hoeher, and E. Villebrun, *“Optimal and Sub-Optimal Maximum A Posteriori Algorithms Suitable for Turbo-Decoding”*, European Trans. Telecommun. (ETT), vol. 8, pp. 119-125, March/April 1997.

[6] V. Weerackody, W. Reichl, and A. Potamianos, *“An Error-Protected Speech Recognition System for Wireless Communications”*, submitted to IEEE JSAC: Wireless Communications, 2000.

[7] A. Bernard, and A. Alwan, *“Joint Channel Decoding – Viterbi Recognition for Wireless Applications”*, Dept. of Electrical Engineering, University of California, Los Angeles.

[8] J. Baker, L. Josifovski, M. Cooke, and P. Green, *“Soft decisions in missing data techniques for Robust Autimatic Speech Recognition”*, Department of Computer Science, University of Sheffield

[9] A. Bernard, and A. Alwan, *“Source and channel coding for remote speech recognition over error-prone channels”*, Dept. of Electrical Enginerring, UCLA

[10] R. Haeb-Umbach, and V. Ion, *“Soft Features for Improved Distributed Speech Recognition over Wireless Networks”*, Dept. of Communications Engineering, University of Paderbon, Germany.

[11] Chin-Hung Sit, Man-Wain Mak, and Sun-Yuan Kung, *“Maximum Likelihood and Maximum A Posterior Adaptatin for Distributed Speaker Recognition Systems”*, Center of Multimedia Signal Processing, Dept. of Electronic and Information Engineering, The Hong Kong Polytechnic University, China.

[12] N. Srinivasamurthy, A. Ortega, and S. Narayanan, *“Efficient Scalar Encoding for Distributed Speech Recognition”*, Department of Electrical Enginerring-Systems, University of Southern California, January 2003.

[13] M. Cooke, P. Green, L. Josifovski, and A. Vizinho, *“Robust Automatic Speech Recognition with Missing and Unreliable Acoustic data”*, Submitted to Speech Communication, 24th June 1999.

[14] Juan M. Huerta, *“Speech Recognition in Mobile Enviroments”*, Dept. of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburg, April 2000.

[15] L. Josifovski, *“Robust Automatic Speech Recognition with Missing and Unreliable Data”*, Department of Computer Science, University of Sheffield, UK, August 2002.

[16] A.P. Bernard, *“Source and Channel Coding Speech Transmission and Remote Speech Recognition”*, University of California, Los Angeles, 2002.