Entropy-Optimized Channel Error Mitigation with Application to Speech Recognition over Wireless

Victoria Sánchez, Antonio M. Peinado, Angel M. Gómez, José L. Pérez-Córdoba

Dpto. de Electrónica y T.C., Universidad de Granada, 18071-Granada (SPAIN)

Abstract

In this paper we propose an entropy-optimized channel error mitigation technique with a low computational complexity and moderate memory requirements, suitable for transmissions over wireless channels. We apply it to Distributed Speech Recognition (DSR), getting an improvement of around 3% in word accuracy over the recognition performance obtained by the mitigation technique proposed in the ETSI standard for DSR (ETSI-ES-201-108 v1.1.2) for bad channel conditions (GSM EP3 error pattern).

1. Introduction

The concept of Distributed Speech Recognition (DSR) has appeared as an efficient way of translating Automatic Speech Recognition (ASR) technologies to mobile and IP network applications, impulsing the development of two ETSI standards for DSR (ETSI-ES-201-108 v1.1.2) [1] and (ETSI-ES-050 108 v1.1.1) that have been elaborated by the Aurora working group of the STQ technical committee.

DSR systems can be affected by several degradation sources due to the acoustic environment and the digital channel. As far as the digital channel is concerned the ETSI DSR standard includes a basic mitigation algorithm that has been shown effective for medium and good quality channels on TETRA and GSM environments. Several channel error mitigation techniques have been proposed that have a good behaviour when channel conditions get worse [2],[3]. In particular our previous proposal [3] gets a performance for a bad quality channel (GSM Error Pattern 3) very close to that obtained for a good or medium quality channels (GSM Error Patterns 1 and 2) but at the cost of a high computational complexity. It is the aim of this work to develop a low complexity mitigation algorithm that still improves in a significant way the performance of the basic mitigation algorithm used in [1] for bad channel conditions.

The front-end utilized in this work is the one proposed in the ETSI standard [1]. This front-end provides a 14-dimension feature vector containing 13 Mel Frequency Cepstrum Coefficients (MFCC) (including the 0th order one) plus log-Energy. These features are grouped into pairs and quantized by means of seven Split Vector Quantizers (SVQ). All codebooks have a 64-center size (6 bits), except the one for MFCC-0 and log-Energy, which has 256 centers (8 bits). The bitstream is generated by grouping frames into pairs (88 bits) that are protected by a 4-bit CRC.

At the back-end, error bursts are detected by means of a CRC checking and a consistency test (we will use this same method for error burst detection). The AURORA mitigation algorithm can be summarized as follows: once a burst, containing 2B frames, is detected, the first B frames are substituted by the last correct frame before the burst and the last B ones by the first correct frame after the burst.

The recognizer is the one provided by Aurora and uses eleven 16-state continuous HMM word models. The training and testing data are extracted from the Aurora-2 speech database. Training is performed with 8440 clean sentences and test is carried out over set A (4004 clean sentences distributed into 4 subsets). The performance of the recognition system will be measured in terms of the Word Accuracy (WAcc).

2. Computational complexity and memory requirement analysis

After the SVQ quantization, each feature pair is represented by a vector \( \mathbf{x} \) (or, equivalently, codeword \( \mathbf{x}^{(i)} \) ) is modeled as a first order Markov process with transition probabilities \( a_{ij} = P(\mathbf{x}_{t+1} = \mathbf{x}^{(j)} | \mathbf{x}_t = \mathbf{x}^{(i)} ) \). For notation simplicity we will express from now on \( \mathbf{x}_t = \mathbf{x}^{(i)} \) as \( \mathbf{x}^{(i)} \). The observation probabilities \( b_t(\mathbf{x}) = P(\mathbf{z}|\mathbf{x}^{(i)}) \) will be computed as a function of the Hamming distance between the hard decoded bit sequence \( \mathbf{z} \) and codewords \( \mathbf{x}^{(i)} \) as

\[
b_t(\mathbf{x}) = (1 - p_e)^{d_H(\mathbf{x}, \mathbf{x}^{(i)})} p_e^{d_H(\mathbf{x}, \mathbf{x}^{(i)})}
\]

The conditional proba-
bilities can be computed from the following forward and backward recursions:

\[ a_t(i) = \sum_{j=0}^{2^M-1} a_{t-1}(j)a_{ij} \beta_i(j) / K_t \quad (t > 0) \]
\[ \beta_t(i) = \sum_{j=0}^{2^M-1} a_t(j)\beta_t+1(j) \quad (t < T) \]

with the following initializations for \( (t = 0) \) and \( (t = T) \),

\[ a_0(i) = P_i b_i(x_0) / K_0 \quad \beta_T(i) = 1 \]

where \( K_t \) is the normalization factor at time \( t \) and \( P_t \) is the a priori probability of \( e^{(i)} \).

We will refer to this approach as FBMMSE (Forward Backward MMSE) estimation. The main problem of the FBMMSE method is that the computational complexity implied is very high and, although the results obtained are spectacular, they may be difficult to implement. If we analyze the computational complexity, we observe that, for the determination of each \( a_t(i) \), we have to carry out: 1) \( 2^M + 1 \) products, 2) the summation of \( 2^M \) terms, 3) one normalization. For the determination of each \( \beta_t(i) \) we have to carry out: 1) \( 2^{M+1} + 1 \) products, 2) the summation of \( 2^M \) terms. Besides we have to carry out the computation of expression (1).

One simplified alternative consists in the use of only the forward probabilities in the expected value computation, Forward MMSE estimation (FMMSE),

\[ \hat{e}_t = \sum_{i=0}^{2^M-1} e^{(i)}a_t(i) \quad (0 < t < T) \]

Although this approach has half computational complexity, it is still rather high from an implementation point of view.

A more simple technique would be to use a standard MMSE estimation

\[ \hat{e}_t = E[e_t|x_t] = \sum_{i=0}^{2^M-1} e^{(i)}b_i|x_t||P_i \]

However, if we consider this much less computationally complex technique, the results are discouraging as we get even worse results than the AURORA mitigation technique as it can be observed in Table 1 for the GSM error patterns EP1, EP2 and EP3 (results for FBMMSE and FMMSE are also shown).

In [4] we have proposed a new approach. In that approach we perform the following estimation of the received parameter vector at time \( t \)

\[ \hat{x}_t = E[e_t|x_t] \quad (1 \leq t \leq B) \]

where \( x_0 \) is the last correctly received vector before a burst and \( x_t \) is the received vector at time \( t \). The length of the burst is \( T - 1 = 2B \). This type of estimation is carried out for the first \( B \) frames of each burst. For the last \( B \) frames of each burst the estimation performed is the following

\[ \hat{e}_t = E[e_t|x_T] \quad (B + 1 \leq t \leq 2B) \]

where \( x_T \) is the first correctly received vector after a burst and \( \hat{x}_t \) is the received vector at time \( t \). That is, the idea is to perform an estimation of \( e_t \) based on the last received parameter vector we can trust before the burst and the received vector at time \( t \) in the case of first \( B \) frames of each burst and an estimation of \( e_t \) based on the first received parameter vector we can trust after the burst and the received vector at time \( t \) in the case of the last \( B \) frames of each burst.

If we develop expressions (8) and (9) we get

\[ \hat{e}_t = E[e_t|x_0,x_T] = \sum_{i=0}^{2^M-1} e^{(i)}\hat{a}_t(i) \quad (1 \leq t \leq B) \]

with

\[ \hat{a}_t(i) = P(x_T|x_0,x_t) \beta_T(i) \]

and

\[ \hat{e}_t = E[e_t|x_T] \quad (B + 1 \leq t \leq 2B) \]

with

\[ \hat{a}_t(i) = P(x_T|x_T) \beta_T(i) \]

In both cases we have two terms. The first one only depends on the source model and can be recursively determined from the transition probabilities \( a_{ij} \) and the a priori probabilities \( P_i \), in the following way

\[ P(x_T|x_0,x_t) = \sum_{i=0}^{2^M-1} P(x_T|x_0,x_t)a_{ij} \]

These probabilities will be precalculated and will not imply any computational cost at the decoder. The other term is the observation probability \( b_i|\hat{x}_t = P(\hat{x}_t|x_0)^{N(M)} \) that depends on the received observation \( \hat{x}_t \) at time \( t \). As we can see the computational complexity of this approach is the same as the computational complexity of the MMSE estimation. We will call this technique Source Model MMSE (SM-MMSE) as we are substituting the a priori probabilities \( P_i \) of the standard MMSE estimation (7) by probabilities \( P(x_T|x_0)^{N(M)} \) or \( P(x_T|x_T)^{N(M)} \) only determined from the source model.

The main drawback is that the reduction in computational complexity is obtained at the cost of an increase in memory requirements. In particular, we need six matrices of dimension \( 64 \times 64 \times 2N \) (one for each codebook of dimension 64) and one matrix of dimension \( 256 \times 256 \times 2B \) (for the codebook of dimension 256) in order to store the probabilities \( P(x_T|x_0)^{N(M)} \) and \( P(x_T|x_T)^{N(M)} \) for each codebook. In [4] we fixed a maximum burst length of 2B of 15 frames, in case we get longer bursts we simply repeat the distribution probability at time \( t = 5, P(x_T|x_0) \), for the frames at times \( 6 \leq t \leq B \) and the distribution probability at time \( t = 2B - 4, P(x_T|x_T) \), for the frames at times \( B + 1 \leq t \leq 2B - 5 \).
Table 1: Performance of the MMSE methods over GSM Error Patterns
EP1 (BER=0%), EP2 (BER=1.76%) and EP3 (BER=3.48%).

<table>
<thead>
<tr>
<th>Wacc (%)</th>
<th>EP1</th>
<th>EP2</th>
<th>EP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AURORA</td>
<td>99.04</td>
<td>98.94</td>
<td>93.40</td>
</tr>
<tr>
<td>FBMMSE</td>
<td>99.04</td>
<td>99.02</td>
<td>98.83</td>
</tr>
<tr>
<td>FMMSE</td>
<td>99.04</td>
<td>99.00</td>
<td>96.73</td>
</tr>
<tr>
<td>MMSE</td>
<td>99.04</td>
<td>98.99</td>
<td>98.18</td>
</tr>
<tr>
<td>SM-MMSE</td>
<td>99.04</td>
<td>99.01</td>
<td>97.03</td>
</tr>
<tr>
<td>SM-AproMMSE</td>
<td>99.04</td>
<td>99.01</td>
<td>96.99</td>
</tr>
<tr>
<td>SM-MMSE(1)</td>
<td>99.04</td>
<td>99.01</td>
<td>97.33</td>
</tr>
<tr>
<td>SM-MMSE(2)</td>
<td>99.04</td>
<td>99.00</td>
<td>96.40</td>
</tr>
</tbody>
</table>

Table 2: Entropies and number of candidates $L^1$ and $L^2$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$H_{a}$</th>
<th>$L^1$</th>
<th>$L^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.79</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3.30</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3.94</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4.40</td>
<td>21</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>4.74</td>
<td>27</td>
<td>14</td>
</tr>
</tbody>
</table>

The results obtained by the SM-MMSE mitigation technique for GSM EP error patterns are shown in table 1 and can be compared with those obtained by the FBMMSE, FMMSE and MMSE techniques. As it can be observed SM-MMSE fills the gap between AURORA and the computationally complex FBMMSE and FMMSE techniques and shows an improvement over AURORA of 3.5% in word accuracy for EP3.

3. Entropy based reduced memory and computational complexity

As it was previously pointed out the main drawback of the proposed mitigation technique is the large amount of memory required. In order to reduce it, in this paper we have studied how the entropies of the probability distributions $P(x^{(i)}_t | x^{(j)}_0)$ and $P(x^{(i)}_{T-t} | x^{(j)}_T)$ evolve as $t$ goes from $t = 1, \ldots, B$ and from $t = B+1, \ldots, B+1$, respectively. These entropies are defined as,

$$H_a(x^{(j)}_t, t) = - \sum_{i=0}^{2M-1} P(x^{(i)}_t | x^{(j)}_0) \log_2 P(x^{(i)}_t | x^{(j)}_0)$$ (14)

$$H_{\beta}(x^{(j)}_{T-t}, t) = - \sum_{i=0}^{2M-1} P(x^{(i)}_{T-t} | x^{(j)}_T) \log_2 P(x^{(i)}_{T-t} | x^{(j)}_T)$$ (15)

As an example we show in table 2 the evolution of the entropies $H_a(x^{(0)}_0, t)$ and $H_{\beta}(x^{(0)}_0, t)$ for the first codeword of codebook $Q_0$ of [1]. From that analysis we have decided to consider only a few $L$ candidates (the most probable according to the distribution) when estimating the error $e_t = E[e_t | x_0, x_1] = \sum_{i=0}^{2M-1} c^{(i)} e_t(i)$ and the estimation $\hat{e}_t = E[e_t | x_T, \hat{x}_1] = \sum_{i=0}^{2M-1} c^{(i)} \hat{h}_t(i)$. The number of candidates $L$ will be determined as a function of the entropies $H_a(x^{(j)}_0, t)$ or $H_{\beta}(x^{(j)}_{T-t}, t)$ and will, consequently, vary with the codebook $Q$, the codeword in the codebook $x^{(j)}$ and the time instant $t$.

We have studied the performance of two possible expressions for $L$

$$L_a(x^{(j)}_t, t) = \text{round}(2^{H_a(x^{(j)}_t, t)}) \quad (1 \leq t \leq B)$$ (16)

$$L_{\beta}(x^{(j)}_{T-t}, t) = \text{round}(2^{H_{\beta}(x^{(j)}_{T-t}, t)}) \quad (B+1 \leq t \leq 2B)$$ (17)

and

$$L_a(x^{(j)}_t, t) = [L_a(x^{(j)}_0, t)/2] \quad (1 \leq t \leq B)$$ (18)

The number of candidates $L$ for each codebook $Q$, each codeword in the codebook $x^{(j)}$ and each the time instant $t$ and their corresponding probabilities would be precalculated and stored. The approaches corresponding to $L^1$ and $L^2$ for the number of candidates will be called SM-MMSE(1) and SM-MMSE(2), respectively.

3.1. Influence of the burst length

Before we study the performance of the proposed approach, we should analyze the following question. As it was previously indicated, in [4] we fixed a maximum burst length $2B$ of 10 frames, in case we got longer bursts we simply repeated the probability distribution at time $t = 5$, $P(x^{(i)}_5 | x_0)$, for the frames at times $0 \leq t \leq B$ and the probability distribution at time $t = 2B - 4$, $P(x^{(i)}_{2B-4} | x_T)$, for the frames at times $B + 1 \leq t \leq 2B - 5$. For the entropy based approaches we will not have the complete distributions $P(x^{(i)}_t | x_0)$ at time $t = 5$ and $P(x^{(i)}_{2B-4} | x_T)$ at time $t = 2B - 4$, but only those probabilities corresponding to the most probable candidates $L$ at times $t = 5$ and $t = 2B - 4$, respectively. So we have to study other possible solutions for the estimation of the parameters at times $5 < t < 2B - 4$ in case of bursts longer than 10 frames. We have tested two possibilities: one of them is to perform an standard MMSE estimation using expression (7) for $5 < t < 2B - 4$ (SM-AproMMSE) and the other one is to use AURORA approach of substituting $x_0$ or $x_T$ for $5 < t < 2B - 4$ (SM-AUMMSE). In table 1 we show the behaviour of the two different proposals in comparison with the proposal of [4] (we are still considering the complete probability distributions at time instants $t \leq 5$ and $t \geq 2B - 4$ for the comparison) for the EP patterns. As it can be observed the solution of performing a standard MMSE estimation for time instants $5 < t < 2B - 4$ in case of bursts longer than $2B = 10$ introduces practically no degradation with respect to the original SM-MMSE proposal. However if we adopt the AURORA approach for those time instants, the degradation introduced is of 1.25% in word accuracy, a significant one.

This may seem surprising if we consider that the MMSE technique gets worse results than the AURORA approach as shown in table 1. One explanation for this behaviour is that in the MMSE based techniques we are using a combination of two types of information one comes from the source model and the other one from the channel model. In the case of the basic MMSE technique the information we use from the source is only the a priori probabilities $P_i$ of each parameter vector $c^{(i)}$ and we are mainly using the information from the channel model to perform the parameter estimation in this technique. In the case of AURORA we are only using information from the source to perform the parameter estimation although in a very simple way: replacing $x_0$ by $x_0$ (or $x_T$ in the assumption that $x_0$ (or $x_T$) is the most probable received codeword at time $t$ once we know that we correctly received $x_0$ at time $t = 0$ or $x_T$ at time $t = T$). Let us now study the histogram of burst durations in the EP3 pattern as determined by the AURORA burst detection algorithm and shown in figure 1. As we can see most of the bursts are of duration smaller than 10. The results obtained by the basic MMSE technique and AURORA for the EP3 pattern seem to indicate that for short burst lengths we get more information from the source (AURORA) than from...
the channel model (basic MMSE). However in the case of the SM-AaprioMMSE and SM-AUMMSE techniques we are using the basic MMSE technique and AURORA, respectively, only for bursts of duration $2B$ longer than 10 and for time instants $5 < t < 2B - 4$. The results obtained seem to indicate that, in that case, when we are far from the last or first correctly received vectors, we get more information from the channel model than from the source in the simple way that AURORA extracts it. So we will use the basic MMSE technique for bursts of duration $2B$ longer than 10 and for time instants $5 < t < 2B - 4$ in the SM-MMSE(1) and SM-MMSE(2) techniques described before.

### 3.2. Performance evaluation

As it was already indicated, in our previous proposal [4] we needed six matrices of floats of dimension $64 \times 64 \times 10$ (one for each codebook of dimension $64$) and one matrix of floats of dimension $256 \times 256 \times 10$ (for the codebook of dimension $256$) in order to store the probabilities $P(x_v^p|\bar{x}^0)$ and $P(x_v^p|\bar{x}_T)$ for each codebook. In the present approach we only need to store $L_1(x_v^p|\bar{x}^0)$, $L_1(x_v^p|\bar{x}_T)$, $L_2(x_v^p|\bar{x}^0)$, $L_2(x_v^p|\bar{x}_T)$, $L_3(x_v^p|\bar{x}^0)$, $L_3(x_v^p|\bar{x}_T)$ floats for the probabilities $P(x_v^p|\bar{x}^0)$ and $P(x_v^p|\bar{x}_T)$ at each time instant $t$ and each codebook element $x_v^p$. In addition we also need to store the indices (short integers) of the $L_1$ or $L_2$ most probable codevectors and six additional matrices of short integers of dimension $64 \times 10$ (one for each codebook of dimension $64$) and one matrix of dimension $256 \times 10$ (for the codebook of dimension $256$) to store the number of candidates $L_1$ or $L_2$ for each time instant $t$ and each codebook element $x_v^p$. Six matrices of dimension $64$ and one matrix of dimension $256$ to store the apriori probabilities of each codebook element are also required. If we use four bytes to store one float and one byte for each short integer we need four times less memory than the original SM-MMSE approach if we use expression $L_1$. We should also compare with the memory requirements of the FBBMMS or FMMSE techniques where six matrices of dimension $64 \times 64$ and one matrix of dimension $256 \times 256$ are required to store the transition probabilities $\alpha_{ij}$ for each codebook element and, additionally, six matrices of dimension $64$ and one matrix of dimension $256$ to store the a priori probabilities of each codebook element. In the case we use expression $L_2$ for the number of candidates and six times less memory than the original SM-MMSE approach if we use expression $L_2$. We should also compare with the memory requirements of the FBBMMS or FMMSE techniques, requirement that can be further reduced to only 1.7 times more memory if we use expression $L_2$ for the number of candidates. This is a large reduction in comparison with the SM-MMSE technique that requires around ten times more memory than the FBBMMS or FMMSE techniques.

This significant reduction in memory requirements is obtained at a small cost in performance. In table 1 we show the performance of the SM-MMSE(1) and SM-MMSE(2) techniques for the EP patterns. As it can be seen there is only a reduction of 0.3% in word accuracy for the SM-MMSE(1) technique and of 0.63% for the SM-MMSE(2) technique with respect to the SM-MMSE technique. As far as computational complexity is concerned there is also a strong reduction with respect to the proposal in [4]. The original computational complexity of SM-MMSE, implied by expressions (10) and (11), of the product, normalization and summation of 64 terms for the codebooks of dimension $64$ and of 256 terms for the codebook of dimension $256$ is reduced in the entropy based approach to the product, normalization and summation of $L_1(x_v^p, t)$, $L_2(x_v^p, t)$ or $L_3(x_v^p, t)$, $L_2(x_v^p, t)$, $L_3(x_v^p, t)$ terms. In this case the computational complexity is not fixed but depends on the codebook, the first $\bar{x}^0$ or last $\bar{x}_T$ correctly received vector and the time instant $t$. In table 3 we show the mean value of the number of candidates $L_1$ and $L_2$ for two of the codebooks, $Q^{2,1}$ and $Q^{1,2,13}$, and each time instant $t$ which can give an idea of the computational complexity involved. Logically the computational complexity increases with $t$ and for burst lengths longer than 10 the computational complexity is that of an MMSE estimation for time instants $5 < t < 2B - 4$. Looking at the histogram of burst durations in the EP3 pattern in figure 1 we saw that most of the bursts are of duration smaller than 10 and, in that case, the entropy based mitigation technique has a smaller computational complexity than the original SM-MMSE technique (much smaller for the $L_2$ approximation and for codebook $Q^{2,1,13}$ of dimension $256$). For bursts of length $2B > 10$ only for time instants $5 < t < 2B - 4$ the computational complexity is similar to the SM-MMSE technique.

### 4. References

[1] “ETSI ETS 201 108 v1.1.2 Distributed Speech Recognition; Front-end Feature Extraction Algorithm; Compression Algorithms”, April 2000.


---

**Table 3: Mean number of candidates $L_1$ and $L_2$ for two codebooks**

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$t = 1$</th>
<th>$t = 2$</th>
<th>$t = 3$</th>
<th>$t = 4$</th>
<th>$t = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q^{1,2}$</td>
<td>9.5</td>
<td>5.7</td>
<td>3.1</td>
<td>2.6</td>
<td>2.3</td>
</tr>
<tr>
<td>$Q^{2,1,13}$</td>
<td>17</td>
<td>30</td>
<td>49.5</td>
<td>60</td>
<td>68.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$L_2$</th>
<th>$2B - 4$</th>
<th>$2B - 3$</th>
<th>$2B - 2$</th>
<th>$2B - 1$</th>
<th>$2B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q^{1,2}$</td>
<td>31.6</td>
<td>27</td>
<td>22.1</td>
<td>16</td>
<td>9.5</td>
</tr>
<tr>
<td>$Q^{2,1,13}$</td>
<td>68.5</td>
<td>60</td>
<td>49.5</td>
<td>35</td>
<td>17.9</td>
</tr>
</tbody>
</table>

---

**Figure 1: Histogram of burst durations in the EP3 pattern**