EVALUATION OF A NOISE-ROBUST DSR FRONT-END ON AURORA DATABASES

Dušan Macho1a, Laurent Mauuary2, Bernhard Noé3, Yan Ming Cheng1a, Doug Ealey1b, Denis Jouvet2, Holly Kelleher1b, David Pearce1b, Fabien Saadoun3

1Human Interface Lab, Motorola Labs, Schaumburg, USA and 2 conscy, 3 Alcatel SEL AG, Stuttgart, Germany
dusan.macho@motorola.com, laurent.mauuary@rd.francetelecom.com, Bernhard.Noel@alcatel.de

ABSTRACT

This paper describes a noise-robust front-end designed within a collaboration of Motorola, France Télécom and Alcatel for the ETSI standardization of the advanced front-end for distributed speech recognition (DSR). The proposed algorithm is based on the cumulative knowledge in the three companies’ history in the areas of noise reduction, speech enhancement as well as other related fields. The major components of this algorithm are noise reduction, waveform processing, cepstral calculation, blind equalization, and voice-activity detection. In the evaluation of the proposed front-end on Aurora 2 and Aurora 3 databases we obtained an average error rate reduction of 52.75% and 51.51%, respectively, when compared to the WI007 ETSI MFCC-based DSR front-end performance.

1. INTRODUCTION

Robustness is an essential issue in practical deployment of automatic speech recognition (ASR) technology. In portable devices such as cell phones, different acoustic environments or channels interfere with speech and reduce the performance of the recognition system. Recent activity in the ETSI Aurora group was focused on the standardization of a robust front-end for distributed speech recognition with well-defined criteria and databases [4]. In this paper, we describe the algorithm which demonstrated the best overall performance among candidates and which was consequently selected in February 2002 to be the new standard for the Advanced DSR Front-end.

In DSR, speech features are calculated and compressed at the terminal side and then transmitted over the network to the server. At the server side, features are decompressed and the recognition itself is performed. This DSR framework and the main components of the presented front-end are displayed in Figure 1. The front-end is split between terminal and server sides. The majority of the front-end calculation is done in the terminal. Here, de-noised cepstral features are calculated in the feature extraction block. Cepstral features are then compressed in the feature compression block and processed for channel transmission in the last block called framing, bit-stream formatting, error protection. At the server side, received features are decoded in the bit-stream decoding, error mitigation block and decompressed in feature decompression. The server feature processing block performs a computationally low feature processing stage, consisting mostly of derivative calculation. Finally, features enter the back-end block, where the recognition takes place.

In the following sections, we describe in more detail those components of the presented front-end that increase the ASR system robustness. Coding and compression schemes will not be covered in this paper. At the end, we present the results we obtained on the Aurora 2 and Aurora 3 databases.

2. NOISE-ROBUST FRONT-END

In this description, we consider the 8 kHz version of the front-end. Extension to 11 and 16 kHz is described in detail in [5]. In the proposed front-end, noise-reduced cepstral features are calculated from the incoming digital signal. We use a two-stage Mel-warped Wiener filter noise reduction scheme, which is a combination of the two-stage Wiener filter scheme from [1] and

Figure 1: Block scheme of the proposed front-end. The upper part shows the components implemented at the terminal side and the lower part shows the components implemented at the server side.
the time domain noise reduction described in [9]. After noise reduction (see Figure 1), SNR-dependent waveform processing (SWP, [6]) is applied to the de-noised signal. The output signal from SWP is used for cepstral calculation. Finally, blind equalization [8] is applied to the cepstral features.

At the server side, in the server feature processing block, the dynamic parameters are calculated from cepstral features (note that for this paper we have omitted the compression and coding related blocks in order to concentrate on the front-end algorithm). Also in this block, the energy coefficient is formed and the feature vectors are selected that enter to the back-end.

The following sections describe the individual blocks of the proposed robust front-end.

2.1. Noise Reduction

Noise reduction is performed by two passes of the Wiener filter (see the block diagram on Figure 2). The first and second stages are similar but not identical. We describe the first stage and then we will explain the differences between the two stages.

Processing is done on a frame-by-frame basis. We use a 25 ms frame length and 10 ms frame shift. The signal spectrum is estimated from a Hanning windowed frame (200 samples) by using an FFT of length 256. The FFT spectrum length is reduced to 65 frequency bins by averaging every two consecutive frequency bins of the 129-bin FFT spectrum. In the next block called PSD mean (Power Spectral Density mean), the averaging of two consecutive power spectra is performed, which reduces the variance of spectral estimation.

The current frame spectrum and the corresponding speech/non-speech decision from the VADNest block (Voice Activity Detector for Noise estimation) are used in the WF design block to estimate the Wiener filter frequency characteristic.

VADNest is an energy-based voice activity detector. The current frame is labeled as speech when the difference between the current frame log energy and the long-term estimate of non-speech log energy exceeds a defined threshold. A hangover of 15 frames is used at the transitions from speech to non-speech segments, provided that the speech segment was at least 5 frames long. The frames labeled as non-speech are used for updating the noise estimate.

The Wiener filter frequency characteristic is estimated in two steps, as shown in Figure 3. The first estimate of the Wiener filter is obtained from the de-noised spectrum $S_{den}$ and noise estimate $S_n$ like

$$H(f,t) = \frac{\eta(f,t)}{1 + \eta(f,t)} \text{ with } \eta(f,t) = \frac{S_{den}(f,t)}{S_n(f,t)}$$

(1)

where $S_{den}$ is computed like

$$S_{den}(f,t) = \beta S_{den2}(f,t - 1) + (1 - \beta) \max \left[ S_{n, raw}(f,t) - S_n(f,t), 0 \right]$$

(2)

where $\beta = 0.98$ and the de-noised spectrum $S_{den3}$ is computed from the previous frame like

$$S_{den3}(f,t - 1) = H_1(f,t - 1) \cdot S_n(f,t - 1).$$

(3)

The second Wiener filter frequency characteristic is obtained from the second de-noised spectrum estimate $S_{den2}$ and the noise estimate $S_n$ like

$$H_2(f,t) = \frac{\eta(f,t)}{1 + \eta(f,t)} \text{ with } \eta(f,t) = \max \left[ S_{den2}(f,t) \right]$$

(4)

where $\eta_n = 0.079432823$ corresponds to the maximum filter attenuation of $-11.33$ dB and $S_{den2}$ is computed by applying the first Wiener filter to the input signal spectrum like

$$S_{den2}(f,t) = H_1(f,t) S_{n, raw}(f,t).$$

(5)

In the Mel Filter-Bank block, the Wiener filter frequency characteristic is smoothed and transformed to a Mel-frequency scale by using 23 triangular Mel-warped frequency windows. The frequency windows coincide with those used in the Cepstrum Calculation block. The impulse response of the Wiener filter is obtained by using a Mel-warped inverse cosine transform in the Mel IDCT block. This impulse response is truncated to a length of 17 and then windowed by a Hanning window. The de-noised signal is obtained by convolving the noisy input signal with the Wiener filter impulse response.

As displayed in Figure 2, the noise-reduced signal from the first stage enters the second stage, where the second Wiener filter is designed and used for noise reduction. The main difference between the two stages is the gain factorization block used in the second stage. In this block, a dynamic, SNR-dependent noise reduction is performed in such a way that more aggressive noise reduction is applied to purely noisy frames and less aggressive noise reduction is used in frames also containing...
speech. We observed that gain factorization could be performed more accurately in the second stage than in the first stage due to the better SNR properties of the noise-reduced signal in the second stage. Another difference is that in the second stage VADNext is not used so that the noise spectrum is updated at each frame.

By this two-stage approach, we gain more flexibility in the Wiener filter design. Notice that the input signal of each stage has a different SNR – in the first stage, the input signal SNR may be very low, while in the second stage the input signal SNR is higher. Thus, in each stage, different decisions are done depending on the current SNR – this non-linear behavior would be difficult to accomplish by a single-pass Wiener filter.

2.2. SNR-dependent Waveform Processing

In voiced segments of the speech signal, the speech waveform exhibits quasi-periodic maxima and minima due to the glottal excitation. By contrast, the interference noise energy can be considered relatively constant within the speech period. Therefore, within a noisy speech period, the SNR is variable; this SNR variability is observable as long as the interference noise intensity is not extremely high. In SNR-dependent Waveform Processing (SWP), which is applied after noise reduction, the high SNR portions of waveform are emphasized and the low SNR waveform portions are de-emphasized by a weighting function. The high SNR portions are detected as maxima of a smoothed energy contour computed from the waveform. The SNR has a decreasing tilt from one maximum to other, thus the first 80% of the interval between the two maxima (including maximum itself) is emphasized and the last 20% is de-emphasized by the weighting function. In this way, the overall SNR is improved and also the speech periodicity is enhanced. A more detailed description of SWP can be found in [6].

2.3. Cepstrum Calculation

The Cepstrum Calculation block is, with a few slight differences, the same like the clean speech standard MFCC front-end described in [3]. Better results were obtained with a lower pre-emphasis coefficient \( a_p = 0.9 \) instead of \( a_p = 0.97 \). Notice that the used filter-bank already includes a pre-emphasis effect because the Mel filter-bank outputs are not energetically normalized. Also, higher noise robustness is observed when using a power spectrum estimate instead of a magnitude spectrum estimate before performing the filter-bank integration. This observation coincides with that in [7].

2.4. Blind Equalization

The blind equalization scheme [8] relies on the least mean square algorithm, which minimizes the mean square error computed as a difference between the current and target cepstrum. The target cepstrum corresponds to the cepstrum of a flat spectrum (notice that as the filter-bank outputs are not energetically normalized, the filter-bank spectrum of a flat input spectrum shows an increasing tilt). Blind equalization reduces the convolutional distortion caused by the use of different microphones in training of acoustic models and testing.

2.5. Server Feature Processing

Three operations are performed in the server feature processing block. An energy coefficient is formed, derivative features are appended to static cepstral features and the relevant feature vectors are selected and sent to the back-end.

2.5.1. Energy Coefficient

Both the log energy and the zero-th cepstral coefficient \( c(0) \) yield information about the whole band energy level in each frame. We obtained the best recognition results by using an energy coefficient \( E_n \) that is a linear combination of both log energy \( bE \) and \( c(0) \) computed for each frame like

\[
E_n = 0.6 \frac{c(0)}{23} + 0.4 \ln(E)
\]

2.5.2. Dynamic Features

It is a well-known fact that adding dynamic information to the static features improves the robustness of speech feature representation. We appended velocity and acceleration features to the 13 static features \( (c(1) \ldots c(12) \text{ and } En) \), both computed over 9 frames. In total, 39 features are used for recognition.

2.5.3. Feature Vector Selection

In noisy speech recognition, the long non-speech segments of signal tend to increase the number of insertion errors. These errors are mostly caused by mismatch between features from non-speech portions of the signal and the silence model. One way to deal with this kind of error is to drop non-speech frames from the recognition process and use mainly the speech frames. This approach can significantly improve the recognition performance of speech surrounded by long noisy segments. For this purpose, we used a voice activity detector described in [2].

3. RECOGNITION EXPERIMENTS

3.1. Databases

The proposed front-end was evaluated on Aurora 2 and Aurora 3 databases. Aurora 2 is the TI digit database artificially distorted by adding noise and using a simulated channel distortion. Two kinds of training are used: clean speech training (denoted as Clean in results tables), and training by using both clean and noisy speech (denoted as Multi). For each training, three tests are realized: A – matched training and testing noises, B – mismatched training and testing noises, and C – test data with both channel (i.e. convolutive) and additive distortions.

Aurora 3 is a set of multi-language SpeechDat-Car databases recorded in-car under different driving conditions with close-talking and hands-free microphones. Three recognition experiments are defined for Aurora 3 with different levels of training and testing mismatch: well-matched, medium mismatched, and highly mismatched (denoted as Well, Mid, and High, respectively, in results tables).

3.2. Acoustic Model Configuration

We tested the proposed front-end by using the back-end configuration as defined by the ETSI Aurora group [4]. The digit models have 16 states with 3 Gaussians per state. The silence model has 3 states with 6 Gaussians per state. Also, a one-state short pause model is used and is tied with the middle state of the silence model.
We also used more complex models, where digit models had 20 Gaussians per state and silence model had 36 Gaussians per state. The number of states was kept the same as in the previous configuration.

### 3.3. Results with Back-End fixed by ETSI Aurora Group

Table 1 and Table 2 show the word error rates and improvements obtained by the proposed front-end on Aurora 2 and Aurora 3 databases, respectively. We can observe that significant improvement is achieved in both databases. In Aurora 2, word error rate was reduced from 12.97% to 8.26% for the multi-condition training experiment (34.82% in relative terms) and from 41.94% to 13.11% for the clean training experiment (70.69% relatively). A similar tendency can be observed in Aurora 3 database: we obtained 47.72% and 73.44% relative improvements in well-matched and high-mismatched experiments, respectively.

### 3.4. Results with Complex Back-End

Table 3 shows the relative improvements we obtained when using the complex back-end configuration. Only Aurora 2 was tested. In comparison to the previous results, a large increase in relative terms can be observed for multi-condition training (from 34.82% to 54.24%) and gave an absolute overall WER of 6.75%. This improvement is attributed to the fact that larger models represent better the variability of the multi-condition training data.

Using the complex back-end, we also measured the improvement this front-end provides relative to the mel-cepstrum and "perfect" endpoints. It gave 52.3% improvement, which is a similar gain to that found for the baseline system (52.75%).

### 4. CONCLUSIONS

In this paper, we presented and evaluated a noise-robust front-end designed for the ETSI distributed speech recognition advanced front-end standard. The proposed front-end contains several components that improve the robustness of ASR systems against both additive noise and channel distortions: Wiener filter based noise reduction, SNR-dependent waveform processing, blind equalization and voice activity detector based feature vector selection. Compared to the previous WI007 ETSI MFCC-based DSR front-end, the relative improvements are on average 52.13%, which surpasses all thresholds set by the ETSI Aurora working group. Additionally, when using more complex back-end modeling, a relative improvement of 65.14% was achieved for the Aurora 2 database.

### 5. REFERENCES


[2] ETSI draft standard doc. “Speech Processing, Transmission and Quality aspects (STQ); Distributed speech recognition; Advanced Front-end feature extraction algorithm; Compression algorithm”, ETSI ES 202 050 v0.1.0 (2002-04), April 2002. (In preparation)


