FRONTEND POST-PROCESSING AND BACKEND MODEL ENHANCEMENT ON THE AURORA 2.0/3.0 DATABASES

Chia-Ping Chen*, *Karim Filali*†, *Jeff A. Bilmes**
{chiaping,karim,bilmes}@ssli.ee.washington.edu
SSLI-Lab, University of Washington, Depts. of EE* and CS†

ABSTRACT

We investigate a highly effective and extremely simple noise-robust front end based on novel post-processing of standard MFCC features on the Aurora databases. It performs remarkably well on both the Aurora 2.0 and Aurora 3.0 databases without requiring any increase in model complexity. Our experiments on Aurora 2.0 have been reported in [1]. In this paper, we evaluate this technique on the Aurora 3.0 corpus, and present updated results on Aurora 2.0. Results in the past have shown that endpointing (i.e., pre-segmentation) on Aurora 3.0 can yield significant improvements. Our experiments reported herein show that our approach integrates well with this endpointing, namely we obtain additional significant improvements. Overall, on Aurora 3.0 we obtain a 47.17% improvement over the segmented baseline. Also, our most recent Aurora 2.0 results show an overall improvement of 41.09% over the baseline for the matched training conditions, and 65.07% for the mis-matched conditions.

1. INTRODUCTION

Noise-robustness is today one of the most challenging and important problems in automatic speech recognition (ASR). The performance of ASR systems often decreases dramatically when the noise level increases. Often the degradation is minor when the signal-to-noise ratio (SNR) is high, but quite significant at low SNR levels. In general, there are two types of noise, additive and convolutional. These different noise type extremes corrupt the speech signal in very different ways, so it is challenging to design a methodology that is robust to both. Even given a technique that ideally handles both types of noise, the problem of *mismatch* remains. That is, when the type of test conditions are different from the training conditions (different additive and/or convolutional noises in the training and testing environments), ASR performance is often extremely poor. The severe degradation in the noisy and/or mismatched environments is one of the major obstacles for the realization of ASR systems in the full variety of adverse acoustic environments that humans are prone to encounter.

The Aurora 2.0/3.0 databases provide excellent platforms in which to research noise-robustness techniques. A variety of sophisticated techniques have been applied to Aurora 2.0. For example, in [4], a frontend consisting of principle component analysis and a discriminative neural network applied to two types of speech features, and a backend consisting of standard Gaussian mixture acoustic models is used. In [5], missing-data theory is used by identifying reliable features in the spectral-temporal domain. In [6], voice activity detector and variable frame rate techniques are used to drop noisy feature vectors to reduce insertion errors.

In past work [1], we developed a novel feature post-processing technique for noise-robust ASR, and demonstrated that it is very effective on Aurora 2.0, even with a simple three Gaussian component per hidden Markov model (HMM) state backend system. In this paper, we apply our technique to Aurora 3.0, and provide new results on Aurora 2.0. In doing so, we show that our method, even when integrated with systems having stronger backends, still yields significant accuracy improvements.[1]

This paper is organized as follows. In section 2, we describe our feature post-processing technique. In section 3, we analyze our proposed method mathematically. In section 4, we present experimental results and compare them with results of the baseline system defined in [2][3]. In section 5, we draw conclusions and describe future research.

Aurora 3 Reference Word Error Rate					
	Finn.	Span.	Germ.	Dani.	Avg.
Well	7.26	7.06	8.80	12.72	8.96
Mid	19.49	16.69	18.96	32.68	21.96
High	59.47	48.45	26.83	60.63	48.85
Overall	24.59	20.78	16.86	31.68	23.48
Aurora 3 Word Error Rate, our MVA Post-Processing					
	Finn.	Span.	Germ.	Dani.	Avg.
Well	3.64	3.66	4.09	7.39	4.69
Mid	10.47	8.45	11.20	20.90	12.76
High	30.04	16.69	13.14	37.93	24.45
Overall	12.63	8.59	8.84	19.75	12.45
Aurora 3 Relative Percentage Improvement					
	Finn.	Span.	Germ.	Dani.	Avg.
Well	49.86	48.16	53.52	41.90	48.36
Mid	46.28	49.37	40.93	36.05	43.16
High	49.49	65.55	51.02	37.44	50.88
Overall	48.51	52.93	48.49	38.74	47.17

Table 1. Our most recent Aurora 3.0 results using MVA post-processing, given as percent word error rate (WER) results and using a 12-Gaussian-component-per-state system. These results include the four Aurora 3.0 languages (Finnish, Spanish, German, and Danish) and the Well-Matched, Medium-Mismatched, and Highly-Mismatched training/testing cases.

2. DESCRIPTION

In this section, we describe our feature post-processing methodology. Note that the post-processing described below will at first appear quite similar to certain schemes well known to the community (namely variance normalization and mean subtraction). The crucial difference between this and past work, however, lies in the domain in which the post-processing is applied.

This work was funded by NSF Award-0086032, and DARPA contract N66001-99-2-892403

[1]Note that this paper is an updated version of [1] but with many new results.

Danish	WM	MM	HM	average	rel imp
baseline2	87.28	67.32	39.37	68.32	=
ep	87.69	70.57	48.09	71.80	10.98
ep.m	89.90	75.26	45.37	73.64	16.79
ep.m.v	90.04	78.91	52.86	76.85	26.93
ep.m.a	91.21	81.38	45.74	76.40	25.51
ep.m.v.a	91.29	80.60	60.59	79.87	36.46
Finnish	**WM**	**MM**	**HM**	**average**	**rel imp**
baseline2	92.74	80.51	40.53	75.41	=
ep	92.15	81.40	47.46	77.22	7.36
ep.m	93.22	84.47	55.41	80.71	21.55
ep.m.v	94.19	85.84	48.27	79.79	17.81
ep.m.a	94.08	84.27	65.27	83.44	32.66
ep.m.v.a	94.28	88.10	65.02	84.80	38.19
German	**WM**	**MM**	**HM**	**average**	**rel imp**
baseline2	91.20	81.04	73.17	83.14	=
ep	92.27	81.63	76.04	84.49	8.01
ep.m	92.77	82.28	76.69	85.08	11.51
ep.m.v	93.97	86.60	84.14	88.93	34.34
ep.m.a	92.51	82.94	78.82	85.74	15.42
ep.m.v.a	94.35	88.21	85.62	90.02	40.81
Spanish	**WM**	**MM**	**HM**	**average**	**rel imp**
baseline2	92.94	83.31	51.55	79.22	=
ep	94.36	84.79	64.51	83.55	20.84
ep.m	93.64	89.65	72.75	87.02	37.54
ep.m.v	95.26	91.97	77.08	89.56	49.76
ep.m.a	94.30	89.28	74.68	87.64	40.52
ep.m.v.a	95.08	92.27	80.78	90.52	54.38

Table 2. Word accuracies (as percentages) for Aurora 3.0 with endpointing, for a 3-Gaussian-component-per-state system. Average = $0.4W + 0.35M + 0.25H$. Percent relative improvement (rel imp) is relative to endpointed baseline (baseline2). ep: endpointed with HCopy as the feature extraction module. m: mean subtraction. v: variance normalization. a: ARMA. WM: Well-Matched; MM: Medium-Mismatched; HM: Highly-Mismatched.

Danish	WM	MM	HM	average	rel imp
baseline1	80.20	51.17	33.07	58.26	=
hcp	82.80	57.55	46.28	64.83	15.74
hcp.m	81.74	64.58	35.91	64.28	14.42
hcp.m.v	79.46	60.81	41.51	63.45	12.43
hcp.m.a	84.86	66.80	37.56	66.71	20.24
hcp.m.v.a	82.86	65.62	51.05	68.87	25.42
Finnish	**WM**	**MM**	**HM**	**average**	**rel imp**
baseline1	90.39	72.37	31.06	69.25	=
hcp	92.36	76.40	35.58	72.58	10.82
hcp.m	92.92	86.39	39.54	77.29	26.15
hcp.m.v	79.49	71.55	15.65	60.75	-27.64
hcp.m.a	94.44	86.94	42.37	78.80	31.05
hcp.m.v.a	78.15	73.39	48.45	69.06	-0.62
German	**WM**	**MM**	**HM**	**average**	**rel imp**
baseline1	90.58	79.06	74.28	82.47	=
hcp	91.18	80.31	76.27	83.65	6.70
hcp.m	91.73	78.77	72.90	82.49	0.08
hcp.m.v	91.22	79.28	78.35	83.82	7.71
hcp.m.a	91.08	77.23	74.10	81.99	-2.77
hcp.m.v.a	92.07	83.89	81.82	86.64	23.80
Spanish	**WM**	**MM**	**HM**	**average**	**rel imp**
baseline1	86.85	73.74	42.23	71.11	=
hcp	89.65	75.26	60.27	77.27	21.33
hcp.m	90.28	82.46	60.99	80.22	31.54
hcp.m.v	91.84	85.25	67.22	83.38	42.47
hcp.m.a	88.88	82.15	65.53	80.69	33.16
hcp.m.v.a	91.99	86.51	75.43	85.93	51.31

Table 3. Word accuracies (as percentages) for Aurora 3.0 without endpointing, for a 3-Gaussian-component-per-state system. Average = $0.4W + 0.35M + 0.25H$. Relative improvement is relative to non-endpointed baseline (baseline1). hcp: use HCopy as the feature extraction module. m: mean subtraction. v: variance normalization. a: ARMA.

We start with standard mel-frequency cepstral coefficients (MFCC) generated by a feature extraction module as our raw features. For a given utterance, we represent the data by a matrix C whose element C_{td} is the dth component of the feature vector at time t, $t = 1 \ldots T$, the number of frames in the utterance and $d = 1 \ldots D$, the dimension of the feature space. In other words, each row of C represents a feature vector and each column represents a time sequence. The first step is mean subtraction (MS) defined by:

$$C'_{td} = C_{td} - \mu_d \tag{1}$$

where

$$\mu_d = \frac{1}{T}\sum_{t=1}^{T} C_{td} \tag{2}$$

This is followed by variance normalization (VN) defined by:

$$\bar{C}_{td} = \frac{C'_{td}}{\sigma_d} = \frac{C_{td} - \mu_d}{\sigma_d} \tag{3}$$

where

$$\sigma_d = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(C_{td} - \mu_d)^2} \tag{4}$$

The third step is processing by a mixed auto-regression moving

$$\begin{tikzcd} wave \arrow[r] & \boxed{\text{FE}} \arrow[r, "C"] & \boxed{\text{M+V}} \arrow[r, "\bar{C}"] & \boxed{\text{A}} \arrow[r, "\breve{C}"] & \boxed{\text{Recognizer}} \end{tikzcd}$$

Fig. 1. Block diagram of our feature post-processing technique. Abbreviations are : FE – feature extraction; M – mean subtraction; V – variance normalization; A – mixed auto-regression and moving average filter.

average (ARMA) filter, defined by:

$$\breve{C}_{td} = \begin{cases} \frac{\sum_{i=1}^{M} \breve{C}_{(t-i)d} + \sum_{j=0}^{M} \bar{C}_{(t+j)d}}{2M+1} & \text{if } M < t \leq T - M, \\ \bar{C}_{td} & \text{otherwise} \end{cases} \tag{5}$$

where M is the order of the ARMA filter. The special case of $M = 0$ degenerates to no ARMA filtering.

A block diagram of the post-processing technique is provided in Figure 1. We will refer to this method as *MVA post-processing*. Note that the MVA post-processing is the same for each time sequence and is not trained from specific data. This means that the filter is not dependent or tuned to a particular type of noise.

Note, for the Aurora 2.0 results presented in [1] and presented in this work, we apply MVA post-processing *after* delta and double-delta feature processing. For all of our Aurora 3.0 results presented herein, however, we applied MVA post-processing after the MFCC calculation but *before* the delta and double-delta feature processing. In both cases, MVA post-processing yielded substantial improvements. We plan to do a side-by-side same-corpus comparison to determine where it is most beneficial to do MVA post-

Danish	WM	MM	HM	average	rel imp
3/6gc	91.29	80.60	60.59	79.87	=
4/8gc	91.12	79.66	62.54	79.96	0.45
8/16gc	91.93	79.38	64.54	80.69	4.07
12/24gc	92.61	79.10	62.07	80.25	1.89
16/32gc	93.03	80.51	60.62	80.55	3.38
Finnish	WM	MM	HM	average	rel imp
3/6gc	94.28	88.10	65.02	84.80	=
4/8gc	94.45	88.51	65.76	85.20	2.63
8/16gc	95.77	89.12	67.28	86.32	10.00
12/24gc	96.36	89.53	69.96	87.37	16.91
16/32gc	96.44	89.33	71.17	87.63	18.62
German	WM	MM	HM	average	rel imp
3/6gc	94.35	88.21	85.62	90.02	=
4/8gc	94.91	89.02	86.12	90.65	6.31
8/16gc	95.41	89.02	86.96	91.06	10.42
12/24gc	95.91	88.80	86.86	91.16	11.42
16/32gc	95.77	88.21	86.08	90.70	6.81
Spanish	WM	MM	HM	average	rel imp
3/6gc	95.08	92.27	80.78	90.52	=
4/8gc	95.56	92.19	82.20	91.04	5.49
8/16gc	95.90	92.10	83.61	91.50	10.34
12/24gc	96.34	91.55	83.31	91.41	9.39
16/32gc	96.46	91.48	83.67	91.52	10.55

Table 4. Word accuracies (as percentages) for Aurora 3.0 with various number of Gaussian components per state, all using MVA post-processing. Average = $0.4W + 0.35M + 0.25H$. Relative improvement (rel imp) is relative to 3/6gc baseline system.

SNR/dB	test A	test B	test C	average	baseline
clean	99.34	99.34	99.17	99.31	99.49
20	99.29	99.24	99.10	99.23	98.79
15	98.89	98.81	98.56	98.79	97.81
10	97.64	97.52	97.24	97.51	95.54
5	93.59	92.76	93.03	93.15	88.18
0	79.39	78.04	79.61	78.89	64.69
-5	49.25	45.20	49.80	47.74	27.25
avg 0-20	93.76	93.27	93.51	93.52	89.00

SNR/dB	test A	test B	test C	average	baseline
clean	99.65	99.65	99.60	99.64	99.69
20	98.48	98.80	98.39	98.59	94.41
15	96.61	97.38	96.56	96.91	84.10
10	92.17	93.21	91.89	92.53	64.55
5	80.65	82.22	80.37	81.22	36.22
0	57.89	59.22	55.46	57.93	12.26
-5	27.99	28.12	26.67	27.78	2.52
avg 0-20	85.16	86.17	84.54	85.44	58.31

Table 5. Our most recent Aurora 2.0 results, using MVA processing and a 16-Gaussian-components-per-whole-word-state system. These results are compared to 20 Gaussian component "baseline" results (right-most columns) that were reported in [7]. Top: multi-condition training; bottom: clean training.

processing, but these results are not reported in this paper (also see Section 5 for future work)

3. ANALYSIS

In equation (5), each mean-subtracted and variance-normalized time sequence is further processed by an ARMA filter. The ARMA filter used is essentially a low-pass filter, smoothing out any spikes in the time sequence. The idea of smoothing out a spiky time sequence is quite natural. While in clean speech the spikes might contain important information about the speech utterance, in noisy speech these spikes are more likely to be caused by noise. Therefore, there is an inherent trade-off in choosing the order M of the filter. A small M will retain the short-term cepstral information but is more vulnerable to noise, while a large M will make the processed features less corrupted by noise, but the short-term cepstral information will be lost. Intuitively, the most extreme cases of $M = 0$ or $M \gg 1$ would have the poorest performance when the speech is noisy. This suggests that the optimal M^* will be a small positive integer. We have verified this conjecture on Aurora 2.0 [1].

In order to examine the relationship between $\bar{C}$ and $\breve{C}$ in the frequency domain, we can rewrite equation (5) as:

$$(2M+1)\breve{C}_{td} - \breve{C}_{(t-1)d} - \cdots - \breve{C}_{(t-M)d} = \bar{C}_{td} + \cdots + \bar{C}_{(t+M)d} \tag{6}$$

From (6), the transfer function is:

$$H(z) = \frac{1 + z + \cdots + z^M}{2M + 1 - z^{-1} - \cdots - z^{-M}} \tag{7}$$

The frequency response of the ARMA filter of order M is:

$$\begin{aligned} H(e^{j\omega}) &= \frac{1 + e^{j\omega} + \cdots + e^{jM\omega}}{2M + 1 - e^{-j\omega} - \cdots - e^{-jM\omega}} \\ &= \frac{1 - e^{j(M+1)\omega}}{2M + 2 - (2M+1)e^{j\omega} - e^{-jM\omega}} \end{aligned} \tag{8}$$

Note that for $\omega = 0$, $H(e^{j\omega}) = 1$. There are $\lfloor \frac{M+1}{2} \rfloor$ zeros in the interval $[0, \pi]$ equally spaced at

$$\omega = 2n\pi/(M+1), n = 1, 2, \ldots, \lfloor \frac{M+1}{2} \rfloor \tag{9}$$

The equation (8) (9) state that the number of zeros in the frequency response of the ARMA filter is approximately proportional to its order. They support the intuition that a large M will perform poorly since it could filter out important speech information. This and further analysis is reported [1]. Note also that for our Aurora 3.0 results presented herein, we consistently use a 2^{nd} order ARMA filter.

4. EXPERIMENTS

In this section, we evaluate our MVA post-processing scheme on both the Aurora 3.0 and Aurora 2.0 corpora. For our Aurora 3.0 results, we comply with the system specified in [3], meaning we vary only the frontend and apply different forms of post-processing using identical backends. As stated above, MVA post-processing is applied to feature streams consisting of 13 features $(c_1, \ldots, c_{12}, \log E)$ per frame on which delta and double-deltas are computed. For Aurora 2.0, MVA post-processing is applied after the delta computations.

In Table 1, we present the spreadsheet giving our Aurora 3.0 results on the four languages (Finnish, Spanish, German, and Danish) and training/testing conditions. The results reported are for the case of 12/24gc (meaning 12 Gaussian components for each wholeword state and 24 Gaussian components for each silence state). As can be seen, our overall improvement over the baseline is 47.17%.

Our results are produced using a speech endpointing (i.e., pre-segmentation) algorithm prior to recognition, following the suggested procedure given in the Aurora 3.0 evaluations guidelines [3]. In this algorithm, a first-pass forced alignment is performed to determine speech/non-speech endpoints on the distribution waveforms (both training and test sets). The beginning and ending non-speech portion of the endpointed waveforms are then striped off, and only the primarily speech portions are used for an additional complete pass of training and recognition, yielding the final reported results.

In Table 2, we present the results on the endpointed speech using different frontends. This endpointing algorithm is done only once using the *non-MVA* baseline features and is used for all of the reported frontends. These experiments show that MVA post-processing is very effective. Qualitatively, each stage of processing improves the performance in addition to previous processing stages. Quantitatively, the all-language average relative improvement with MVA post-processing is 42.46% over the endpointed baseline. Note that there is a gain when we use a different base MFCC feature extraction program (HTK's HCopy, listed as "ep" in the figure) than the one (FrontEnd, listed as "baseline2" in the figure) provided with the Aurora 3.0 distribution. Compared to the HCopy baseline features, the all-language average relative improvement with post-processing is 34.98%. This number can be attributed strictly to MVA post-processing.

For comparison, Table 3 shows the results without the endpointing algorithm. One can see that for most of the languages (outside of Finish), MVA post-processing in this case is as effective as in the endpointed case. Furthermore, when comparing the "baseline2" rows in Tables2 and "baseline1" rows in Table 3, one can see the effect only of the endpointing algorithm, and the fact that the improvements given by MVA post-processing are cumulative.

In Table 4, we present the (once again endpointed) results varying the backend, i.e., the number of Gaussian components in the HMM system. The goal here is to identify the backend that performs best with MVA features. Four epochs of parameter re-estimation are performed at each step when increasing the number of Gaussian components, so the total number of epochs is 24 for 3/6gc (3 Gaussian components for each wholeword state and 6 Gaussian components for each silence/sp states) and 40 for 16/32gc. It is noteworthy that as a general trend, the improvements are significant over the 3/6gc system, but they saturate around 12/24gc. Different languages have slight variations in behavior. We choose the 12/24gc case for our final Aurora 3.0 results given in Table 1.

In [1], we investigated MVA post-processing using a 3gc baseline recognizer. In this work, we re-evaluate our Aurora 2.0 using a stronger backend [8].[2] In Table 5, we present our results with a 16-Gaussian-components-per-state system. One can see that the MVA post-processing yields significant improvements compared to the case without MVA post-processing. The overall average improvement on the $0-20$dB test cases is 41.09% for the multi-condition training and 65.07% for the clean training. This shows that the MVA processing can be integrated with stronger backends to yield additional improvements on Aurora 2.0.

[2]For these results, this system simulated only an HMM.

5. SUMMARY AND FUTURE WORK

In this paper, we presented a feature post-processing methodology which is extremely simple yet extremely effective on the Aurora 2.0/3.0 databases, extending on work initially presented in [1]. Our technique achieves the same level of performance as systems which use much more sophisticated acoustic modeling and/or speech enhancements techniques. Our method does not require any knowledge about the noise types, and yields significant improvements in the well-matched, medium-matched, and highly-mismatched training and testing environments.

Our method also yields improvements when combined with stronger backends. With the post-processing and endpointing algorithm in place, increasing the number of Gaussian components per state further improves the performance.

Comparing the performance of different languages, one can see that the Danish and Finnish corpora have worse word accuracies than German and Spanish. Seeing that the system is designed irrespective of the differences between languages (such as the duration of digits or the amounts of the available data), further improvements might be obtained if we were able to modify the system in a language-dependent way. Furthermore, we expect that some of the novel back-end statistical modeling available in [8] could improve results further.

6. REFERENCES

[1] C. Chen, J. Bilmes, K. Kirchhoff, "Low-Resource noise-robust feature post-processing on Aurora 2.0", submitted to ICSLP 2002.

[2] H. G. Hirsch and D. Pearce, "The AURORA Experimental Framework for the Performance Evaluations of Speech Recognition Systems under Noisy Conditions", ISCA ITRW ASR2000, Paris, September 2000.

[3] Motorola Au/374/01, "Small vocabulary evaluation: Baseline Mel-Cepstrum Performances with Speech Endpoints", Oct. 2001.

[4] D. Ellis and M. Gomez, "Investigations into Tandem Acoustic Modeling for the Aurora Task", pp. 189-192, Proceedings Eurospeech 2001.

[5] J. Barker, M. Cooke, and P. Green, "Robust ASR Based on Clean Speech Models: An Evaluation of Missing Data Techniques for Connected Digit Recognition in Noise", pp. 213-216, Proceedings Eurospeech 2001.

[6] Johan de Veth, Laurent Mauuary, Bernhard Noe, Febe de Wet, Juergen Sienel, Louis Boves and Denis Jouver, "Feature Vector Selection to Improve ASR Robustness in Noisy Conditions", pp. 201-204, Proceedings Eurospeech 2001.

[7] David Pearce, Ansela Gunawardana "Aurora 2.0 Speech Recognition in Noise: Update 2", email from David Pearce providing spreadsheet with most recent 20 Gaussian-component Aurora 2.0 results, April 19, 2002. Also see item 18, `http://icslp2002.colorado.edu/special_sessions/aurora`.

[8] Jeff Bilmes and Geoff Zweig, "The Graphical Models Toolkit: An Open Source Software System for Speech and Time-Series Processing", in International Conference on Acoustics, Speech, and Signal Processing, May 2002, Orlando Fl.