MODEL COMPENSATION FOR NOISES IN TRAINING AND TEST DATA

Driss Matrouf and Jean-Luc Gauvain

LIMSI-CNRS, BP 133
91403 Orsay cedex, FRANCE
{driss, gauvain}@limsi.fr

ABSTRACT

It is well known that the performances of speech recognition systems degrade rapidly as the mismatch between the training and test conditions increases. Approaches to compensate for this mismatch generally assume that the training data is noise-free, and the test data is noisy. In practice, this assumption is seldom correct. In this paper, we propose an iterative technique to compensate for noises in both the training and test data. The adopted approach compensates the speech model parameters using the noise present in the test data, and compensates the test data frames using the noise present in the training data. The training and test data are assumed to come from different and unknown microphones and acoustic environments. The interest of such a compensation scheme has been assessed on the MASK task using a continuous density HMM-based speech recognizer. Experimental results show the advantage of compensating for both test and training noises.

INTRODUCTION

The performances of speech recognizers drop substantially when there is a mismatch between training and testing conditions. The goal of noise compensation is to minimize the effects of such a mismatch, so as to obtain a recognition accuracy as close as possible to that obtained under matched conditions. Approaches based on a channel model generally assume that the training data is noise-free, and the test data is noisy[1, 2, 3, 4, 7, 10]. In practice, this assumption is rarely correct. In this paper, we investigate an iterative procedure to compensate for noise both in the training data and in the test data. The training and test noises are expected to be of different natures with different levels. The data are also assumed to not have been recorded under the same conditions, and are likely to come from different and unknown microphones and acoustic environments. Our technique has been assessed using the LIMSI recognizer, based on continuous density HMM [6]. All experiments were carried out on the MASK[8] corpus of spontaneous speech.

In the next section we outline the compensation technique for treating noises in both the training and test data. We then provide experimental results using this method, as well as using MLLR adaptation to compensate for the residual error.

COMPENSATION PROCESS

Assuming that the training data is noise-free, that the test data is noisy, and that the recording channel can be represented by a linear filter, then the training and test signals can reasonably be modeled as follows (denoted *model 1*):

$$\begin{array}{ll} y_1(t) = h_1(t) * s(t) & \text{for training data} \\ y_2(t) = h_2(t) * (s(t) + n_2(t)) & \text{for test data} \end{array}$$

If we assume that the training data is also noisy, then the channel model becomes (denoted *model 2*):

$$\begin{array}{ll} y_1(t) = h_1(t) * (s(t) + n_1(t)) & \text{for training data} \\ y_2(t) = h_2(t) * (s(t) + n_2(t)) & \text{for test data} \end{array}$$

where $s(t)$ is the hypothetical noise-free signal, $n_1(t)$ and $h_1(t)$ represent respectively the additive and convolutional noise in the training data, $n_2(t)$ and $h_2(t)$ represent respectively the additive and convolutional noise in the test data.

Since no prior knowledge of either the channel type or the background noise characteristics is available, model compensation has to be performed using only the test data in an unsupervised mode. The compensated models are obtained by adapting models trained on y_1. This can be done using a parallel model combination (PMC) technique[3, 4] which approximates a noisy speech model by combining a speech model (trained on y_1) with a noise model (generally Gaussian trained on a sample of n_2). Various PMC techniques have been proposed, including log-normal approximation[3], numerical integration[4], and data driven approaches[5, 7]. For this work we have chosen to use a data-driven approach[7], where the approximations needed for model combination are avoided by directly using the original training speech samples instead of generating speech samples from a speech model[5]. With a proper organization of the data, the combination process can be performed as fast as PMC using the log-normal approximation. However, it should be noted that the two compensation algorithms described in this section do not rely on this particular combination scheme, and can be applied in conjunction with any (or any mixture) of the

various PMC techniques (subsequently with various degrees of approximation).

Compensation for the test noise

In this case we assume that only the test data is noisy. To carry out combination with the training frames y_1, an estimate of $h_1 * n_2$ is needed. However, for the test data only frames representing $h_2 * n_2$ are available. To obtain $h_1 * n_2$, we estimate the filter $h_2^{-1} * h_1$ which is applied to $h_2 * n_2$ to obtain an estimate of $h_1 * n_2$. This filter is iteratively refined using the following algorithm based on channel model 1:

Test Noise Estimation (algorithm 1):
Combination is done in the spectral domain
All other calculations are made in the cepstral domain
1. $\tilde{n}_2 := h_2 * n_2$, first estimate of $h_1 * n_2$
2. Calculate M_2, the cepstrum mean of $y_2 = h_2 * (s + n_2)$
3. Combine y_1 with $\tilde{n}_2$ to get the estimate, $\tilde{M}_1$, of the cepstrum mean of $h_1 * (s + n_2)$
4. Estimate $h_1 * h_2^{-1}$ in the cepstral domain as:
$\tilde{H} := \tilde{M}_1 - M_2$
5. Use the filter $\tilde{H}$ to obtain a new estimate of $h_1 * n_2$:
$\tilde{n}_2 := \tilde{h} * h_2 * n_2$
loop back to 3 until convergence

We have observed that in practice about 5 iterations are needed to properly estimate $h_1 * n_2$. The estimated noise is then combined in the spectral domain with y_1 to build the new acoustic models. Normalization with respect to the convolutional component h is done by subtracting the cepstrum mean from the modified training data frames and the test data frames. The combination process is carried out for each utterance in the training data. This data driven approach enables straightforward calculation of the derivative parameters. For a given training frame generated by a certain state, belonging to a Gaussian in this state is determined by choosing the Gaussian with the highest posterior probability. To simplify the combination, this association between the training frames and Gaussians is determined before compensation and is assumed to remain unchanged by the combination process, even though this is not required by the approach. When all utterances have been treated, the accumulated statistics associated with a Gaussian are used to reestimate its mean vector and covariance matrix.

Compensation for the test and training noises

In this case we assume that both the training data and the test data are noisy, and use channel model 2. Our technique consists of combining the test noise with the training data frames y_1 and the training noise with the test data frames y_2. For this combination we need an estimate of $h_1 * n_2$ (test noise) and an estimate of $h_2 * n_1$ (training noise), but only frames representing $h_2 * n_2$ and $h_1 * n_1$ are available. To obtain these, we estimate the filter $h_2^{-1} * h_1$ which is applied to $h_2 * n_2$ frames to get an estimate of $h_1 * n_2$. In the same manner, the estimated filter $h_1^{-1} * h_2$ is applied to the training noise frames $h_1 * n_1$ in order to obtain the signal $h_2 * n_1$. The estimated filters $h_1^{-1} * h_2$ and $h_2^{-1} * h_1$ are obtained iteratively using the following algorithm based on channel model 2:

Test & Training Noise Estimation (algorithm 2):
Combination is done in the spectral domain
All other calculations are made in the cepstral domain
1. $\tilde{n}_1 := h_1 * n_1$, first estimate of $h_2 * n_1$
2. $\tilde{n}_2 := h_2 * n_2$, first estimate of $h_1 * n_2$
3. Combine y_1 with $\tilde{n}_2$ to get an estimate, $\tilde{M}_1$, of the cepstrum mean of $h_1 * (s + n_1 + n_2)$
4. Combine y_2 with $\tilde{n}_1$ to get an estimate, $\tilde{M}_2$, of the cepstrum mean of $h_2 * (s + n_1 + n_2)$
5. Estimate $h_1^{-1} * h_2$ in the cepstral domain as:
$\tilde{H} := \tilde{M}_2 - \tilde{M}_1$
6. Use filter $\tilde{H}$ to get new estimates of $h_2 * n_1$ and $h_1 * n_2$:
$\tilde{n}_1 := \tilde{h} * h_1 * n_1$
$\tilde{n}_2 := \tilde{h}^{-1} * h_2 * n_2$
loop back to 3 until convergence

Once $h_2 * n_1$ and $h_1 * n_2$ are estimated (as for algorithm 1 we ran 5 iterations), they are combined with y_1 and y_2 respectively to obtain the modified training data $h_1 * (s + n_1 + n_2)$ and test data $h_2 * (s + n_2 + n_1)$. Normalization with respect to the convolutional component h is done by subtracting the cepstrum mean from the modified training and test frames.

After estimating $h_1 * n_2$, the combination process with the training data is done as before. To create the modified cepstrum test data, we can either directly combine $\tilde{n}_1$ (i.e. a sample representive of the training noise) with the test utterance in the spectral domain frame by frame in circular manner, or simply combine the mean value of $\tilde{n}_1$ with the test data. For the experiments reported here, there was no significant difference between the two strategies.

Estimation of the test noise

A sample of the test noise $n_2 * h_2$ is needed for the compensation process. Our experiments used a 4s background noise sample for each speaker session. In a real-world application this test noise sample could be obtained by recording the background noise inbetween speaker sessions. To be more effective, a recognizer using the models adapted with the initial noise sample can be used to detect the silence segments in the test utterances. These silence segments can then be used during compensation.

EXPERIMENTAL RESULTS

Evaluation conditions

To evaluate these compensation schemes, we used 22148 utterances from the MASK corpus[8] from 460 speakers. The data from 450 speakers were used for training and data from 10 speakers were reserved for test. The data was collected using two microphone channels, a close-talking microphone (S_a, SNR $\simeq$ 35dB) and tabletop PCC microphone (S_b, SNR $\simeq$ 21dB). The speech signal is bandlimited to 8kHz and sampled at 16 kHz. Users can ask the MASK system about

train travel information, such as timetables, tickets and reservations for train travel among 500 cities in France. The speaker-independent, continuous speech recognizer is capable of recognizing spontaneous speech, and runs in real-time with recognition vocabulary of 1500 words and a bigram language model. Such a system could be placed in a variety of different locations, having different acoustic environments, whose characteristics may vary at different times of the day. Successful deployment of such systems requires training acoustic models with field training data, ie., data collected from real users interacting with the system. This data can be expected to be quite noisy and variable compared to typical laboratory data.

The feature vector is composed of 13 MFCC cepstrum coefficients and their first and second derivatives, estimated every 10ms. Cepstral mean removal is performed for each sentence. 608 gender-independent, context-dependent phones (including silence) are modeled. Each context-dependent phone model is a left-to-right, CDHMM with Gaussian mixture observation densities (typically 20 components).

For all experiments the acoustic models were trained on the signal S_a. To simulate different SNRs for the training data, the Lynx noise on the NOISEX-92 database[11] was added to S_a at different levels. Three SNR values were used for training: 35dB, 20dB and 12dB. Noise recorded in a Parisian train station was added to S_b to form the test signal with three different SNRs: 20dB, 12dB and 9dB. The first row in each table uses the clean signal, S_a for the test. For all train-test SNR pairs, experiments were carried out with *no compensation*, *test noise compensation* and *test & train noise compensation*.

Results discussion

Tables 1, 2 and 3 respectively show the results obtained for the three different configurations: *no compensation*, *test noise compensation* and *test & train noise compensation*. Compensating for the noise in the test data always is advantageous, even with a relatively high SNRs (compare Tables 1 and 2). When there is no noise in the test data, compensation for test noise does not disturb the system. Training with and SNR of 12dB and testing data with an SNR of 20dB, the word error decreases from 24.1% to 18.6% after compensating for the test noise. This error rate is still 2.5 times as high as the error rate under clean matched conditions. Table 1 shows that the performance degrades substantially when the system is trained on noisy data and tested on clean data, or inversely, by training on clean data and testing on noisy data.

Comparing Tables 2 and 3, an improvement is obtained by compensating for the noise in the training data, especially when the test data is relatively clean. Training on S_a with a 12dB SNR and testing on S_b with a 20dB SNR, compensating only for the test noise decreased the word error rate by 23%, from 24.1% to 18.6%, whereas compensation for both test and training noises decreased the word error by over

Test Condition	Training Condition SNR=35dB	SNR=20dB	SNR=12dB
S_a with SNR=35dB	6.9%	11.9%	35.5%
S_b with SNR=20dB	13.1%	15.7%	24.1%
S_b with SNR=12dB	28.3%	26.5%	26.0%
S_b with SNR=9dB	51.8%	49.8%	42.0%

Table 1: No compensation

Test Condition	Training Condition SNR=35dB	SNR=20dB	SNR=12dB
S_a with SNR=35dB	6.8%	11.9%	34.2%
S_b with SNR=20dB	11.2%	12.1%	18.6%
S_b with SNR=12dB	11.8%	12.9%	13.6%
S_b with SNR=9dB	14.3%	16.3%	17.6%

Table 2: Test noise compensation

Test Condition	Training Condition SNR=35dB	SNR=20dB	SNR=12dB
S_a with SNR=35dB	6.6%	7.2%	7.6%
S_b with SNR=20dB	10.7%	11.8%	11.2%
S_b with SNR=12dB	11.8%	11.9%	12.5%
S_b with SNR=9dB	14.6%	16.0%	15.2%

Table 3: Train&test noise compensation

Test Condition	Training Condition SNR=35dB	SNR=20dB	SNR=12dB
S_b with SNR=9dB	14.0%	14.9%	14.2%

Table 4: Train&test noise compensation + unsupervised MLLR

Tables 1, 2, 3 show the average word error rates for various SNR pairs in the training and test data with 1: no compensation, 2: test noise compensation using algorithm 1, 3: test & training noise compensation using algorithm 2. The columns vary the SNRs of the training data, the rows vary the SNRs of the test data. Training was carried out using the same channel (S_a) for all SNR values. For the test data, the first row uses the channel (S_a) and the remaining rows use channel (S_b). Table 3 contains results using additional unsupervised MLLR adaptation.

50%, from 24.1% to 11.2%. As expected when the test data is clean, test data compensation does not affect the system performance when training on noisy data (compare the first row of tables 1 and 2). In contrast, compensation for both test and training noises (see table 3) leads to significantly reduced word error rates compared to the *no compensation* condition in Table 1 and *test noise compensation* condition in Table 2. We can conclude that compensating for the noise in the training data is essential when the test data is clean, and the more noisy the training data, the more important this is.

Comparing Tables 1 and 2 shows that even for only slightly noisy test data, compensation for the test data noise leads to a significant reduction in word error compared to the *no compensation* condition. In contrast, when both the test data and training data are only slightly noisy, little is gained by compensating for the noise in the training data (compare Tables 2 and 3).

Table 3 also shows that we are better able to compensate for training noise when the test data is clean (top row) than for the test noise when the training data is clean (first column). Training on S_a with and SNR of 20dB or 12dB, and testing on S_a with SNR=35dB, the resulting word error rates are 7.2% and 7.6% respectively, which are comparable to the error rates obtained under matched conditions (S_a with SNR=35dB). Training on S_a with an SNR of 35dB and testing on S_b with 20dB SNR, leads to a word error rate of 11.2%, with *test noise compensation* (Table 2) and 10.7% compensating for both *train & test noises*. When there is no noise in the training data (SNR=35dB), compensation for the training noise does not disturb the system.

After compensation based on a channel model, unsupervised adaptation based on the Maximum Likelihood Linear Regression technique (MLLR)[9] can be carried out to compensate for the residual mismatch between the training and test data. This residual mismatch is due to the noises which are not represented by the channel model, such as inter-speaker variability. A single regression matrix (43×42) is used to transform the Gaussian means of the models. Experiments were carried out for the following configurations: for the test data, the channel S_b with SNR=9dB was used; for the training data, the channel S_a with SNRs of 35dB, 20dB and 12dB were used. An additional 6% reduction in the word error rate is obtained using transcription mode MLLR adaptation as shown in Table 4.

CONCLUSION

In this paper we have described a compensation technique which takes account of noise in both the training and test data. The training and test data are assumed to be recorded with different microphones in a variety of background noise conditions. Environmental adaptation based on the channel model $y = (s+n1)*h1$ for the training data and the channel model $y = (s+n2)*h2$ for the test data is demonstrated to be effective since it significantly reduces the word error rate. The gain obtained by compensating the training noise is particularly important when the test data is clean. When testing on S_a with an SNR of 35dB and training on S_a with 12dB and 20dB SNR, compensating for the training and test noises decreases the word error rate by 77% and 39%, respectively, compared to compensating only for the test noise.

The gain is still observed when testing on moderately clean data and training on noisy data. When models were trained on S_a with a 12dB SNR and tested on S_b with a 20dB SNR showed that compensating for the training and test noises decreased the word error rate by 40% compared to compensating only for the test noise (from 18.6% to 11.2%). Compensating for both test and training noises was found to always be fruitful, without degradation even if the data are noise-free. After compensation for the test and training noises, MLLR adaptation was found to compensate for the residual mismatch between the training and test data.

REFERENCES

[1] A. Acero, R.M. Stern, "Enviromental Robustness in Automatic Speech Recognition," *IEEE Acoustics, Speech & Signal Processing*, pp. 849-852. April 1990.

[2] D. VanCompernolle, "Noise adaptation in a hidden Markov model speech recognition system", *Computer Speech & Language*, **3**(2), pp. 151-167, 1989.

[3] M. Gales, S. Young, "An improved approach to hidden Markov model decomposition of speech and noise," *ICASSP-92*, pp. 233-236, March 1992.

[4] M.J.F. Gales, S.J. Young, "Robust speech recognition in additive and convolutional noise using parallel model combination," *Computer Speech & Language*, **9**(4), pp. 289-307, 1995.

[5] M.J.F. Gales, S.J. Young, "A fast and flexible implementation of parallel model combination," *ICASSP-95*, pp. 133-136.

[6] J.L. Gauvain, L.F. Lamel, G. Adda, M. Adda-Decker, "Speaker-Independent Continuous Speech Dictation," *Speech Communication*, vol. 15, pp. 21-37, 1994.

[7] J.L. Gauvain, L. Lamel, G.Adda, D.Matrouf, "Developments in Continuous Speech Dictation using the 1995 ARPA NAB News Task," *ICASSP-96*, pp. 73-76.

[8] L. Lamel, S. Rosset, S. Bennacef, H. Bonneau-Maynard, L. Devillers, J.L. Gauvain, "Development of Spoken Language Corpora for Travel Information," *Eurospeech'95*, pp. 1961-1964.

[9] C. Legetter, P. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Computer Speech & Language*, **9**(2), pp. 171-185, 1995.

[10] F. Martin, K. Shikano, Y. Minami, "Recognition of Noisy Speech by Composition of Hidden Markov Models," *Eurospeech'93*, pp. 1031-1034.

[11] A.P. Varga et al. "The NOISEX-92 study on the effect of additive noise on automatic speech recognition," *Technical Report, DRA Speech Research Unit*, 1992.