COMPARISON OF STANDARD AND HYBRID MODELING TECHNIQUES FOR DISTRIBUTED SPEECH RECOGNITION

Jan Stadermann, Gerhard Rigoll

University of Duisburg, Department of Computer Science
Bismarckstr. 90, 47057 Duisburg
Phone:+49-203-379-{4222, 4221},
Email: {stadermann, rigoll}@fb9-ti.uni-duisburg.de

ABSTRACT

Distributed speech recognition (DSR) is an interesting technology for mobile recognition tasks where the recognizer is split up into two parts and connected with a transmission channel. We compare the performance of standard and hybrid modeling approaches in this environment. The evaluation is done on clean and noisy speech samples taken from the TI digits and the AURORA database. Our results show that that the hybrid modeling techniques can outperform standard continuous systems on this task.

1. INTRODUCTION

The size reduction of portable computers or mobile phones demands new methods of man-machine interaction. One possible way of communication is speech, but today's speech recognizers require considerable storage and computation power to produce adequate results. The DSR framework divides the standard recognizer into the feature extraction on the client side (with low requirements regarding memory or processor power) and the classifier - in our case *hidden Markov models* (HMMs) - which can be implemented at the server side deploying complex statistical algorithms and large language models (see figure 1). The feature extraction computes 13 mel-frequency cepstrum coefficients $(c_0, \ldots, c_{12})^T$ (MFCCs) and the logarithmic frame energy E every 10 ms for a 25 ms frame of speech samples. The major issue in DSR is the transmission channel that possesses only a limited bandwidth, but is otherwise considered ideal for our investigations[1]. The features are vector quantized and transmitted over the channel to the recognizer. We take the bit rate as channel characterization which is directly related to the bandwidth [1].

Our investigated channel allows a bit rate of 4.4 kbit/s, with channel coding and header the bit rate is 4.8 kbit/s, this is half of the data transmission bit rate in GSM. More details about the channel can be found in [2]. We will show in the following sections that our hybrid modeling techniques possess certain advantages compared to traditional methods for distributed speech recognition. Section 2 gives an overview about the vector quantization of the features, sections 3 and 4 present continuous and discrete recognizers based on vector-quantized features. A hybrid recognizer based on quantized tied-posteriors is given in section 5, section 6 contains the results and section 7 summarizes this article.

[1] appropriate channel coding can protect the data in real environments

2. VECTOR QUANTIZATION OF CONTINUOUS FEATURE VECTORS

The vector quantization (VQ) step is necessary to reduce the amount of data that is sent over the transmission channel: Since all feature coefficients are `float` values (that require on most machines 4 bytes), we would have to transmit $14 \cdot 4$ bytes resulting in a bit rate of

$$\mathrm{BR} = \frac{14 \cdot 4 \cdot 8\,\mathrm{bits}}{10\,\mathrm{ms}} = 44.8\,\mathrm{kbit/s}$$

To reduce this amount a VQ is used that is based on the *k-means algorithm* with an Euclidean distance measure. For our channel (4.4 kbit/s) we use a quantization scheme taken from the ETSI standard [2], see figure 2. Two components from the original feature vector are composed into a new vector that is then quantized. Since only the indices of the

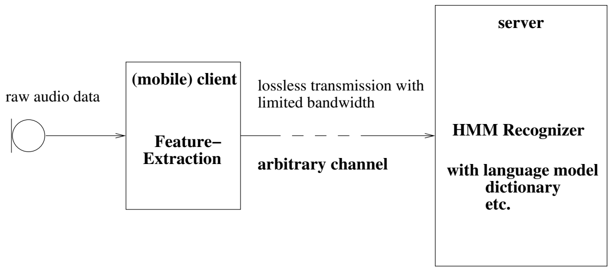

Fig. 1. DSR set-up

$$\begin{pmatrix} c_0 \\ c_1 \\ \cdot \\ \cdot \\ \cdot \\ c_{12} \\ E \end{pmatrix} => \begin{array}{ll} \vec{v_1} = (c_0, E) & \\ \quad \text{64 cdb. vectors} & \vec{v_5} = (c_7, c_8) \\ \vec{v_2} = (c_1, c_2) & \quad \text{64 cdb. vectors} \\ \quad \text{64 cdb. vectors} & \vec{v_6} = (c_9, c_{10}) \\ \vec{v_3} = (c_3, c_4) & \quad \text{64 cdb. vectors} \\ \quad \text{64 cdb. vectors} & \vec{v_7} = (c_{11}, c_{12}) \\ \vec{v_4} = (c_5, c_6) & \quad \text{256 cdb. vectors} \\ \quad \text{64 cdb. vectors} & \end{array}$$

Fig. 2. Codebook generation of the vector quantizer (7 codebook vector indices per frame

codebook vectors need to be transmitted the bit rate is

$$\mathrm{BR_{VQ}} = \frac{6 \cdot 6\,\mathrm{bits} + 8\,\mathrm{bits}}{10\,\mathrm{ms}} = 4.4\,\mathrm{kbit/s}$$

2.1. Optimizing the vector quantizer using a maximum mutual information criterion

An improvement of the vector quantizer presented in section 2 can be achieved if a neural VQ trained to maximize the *mutual information* between the VQ labels Y and the pattern class stream W (e.g. words or phonemes) is introduced [3]. This approach seems to be suited for DSR since the amount of data to be sent over the channel is unchanged compared to a standard k-means vector quantizer, but MMI-NN hybrid approaches usually outperform traditional discrete systems as shown in [3] and [4].
If we denote the parameter set of the VQ with Θ, the mutual information between the VQ stream Y and the pattern class stream W can be written as follows [4]:

$$I(W, Y_\Theta) := H(Y_\Theta) - H(Y_\Theta|W) = H(W) - H(W|Y_\Theta) \tag{1}$$

Since we are interested in modifying Y_Θ to maximize $I(Y_\Theta, W)$, the maximum mutual information (MMI) is reached if

$$\begin{aligned} H(W|Y) &= -\sum_I \sum_M p_\Theta(w_i, y_m) \cdot \log(p_\Theta(w_i|y_m)) \\ &= -\sum_I \sum_M p_\Theta(w_i, y_m) \cdot \log \frac{p_\Theta(w_i, y_m)}{\sum_R p_\Theta(w_r, y_m)} \end{aligned} \tag{2}$$

is minimized with respect to Θ.

This optimization is performed using a neural net with the parameter set Θ associated with the network weights. The weights are computed with a gradient descent algorithm based on the derivative $\frac{\partial H(W|Y_\Theta)}{\partial \Theta}$ using (2), the details of the algorithm can be found in [4]. For our experiments we have trained the MMI-NN with the same pseudo-phonemes that are introduced in section 5.

3. RECOGNITION WITH DISCRETE FEATURES

Our discrete hidden Markov model consists of a transition matrix and an array of discrete values for each state. These values represent the probabilities of every possible codebook vector given a HMM state. Since we have 7 different codebooks, we use 7 probability arrays (streams) in each state. For our recognition task (recognizing spoken digits, see section 6) we use whole word models plus two silence models for interword silence and sentence start/end silence, respectively. The exact topology is adopted from [5] and depicted in figure 3. We know from section 2 that the

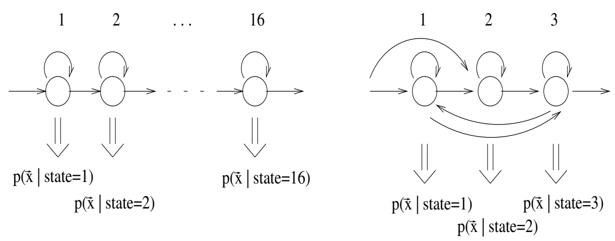

Fig. 3. HMM topology for number words (left) and 3-state silence model (right)

VQ indices extracted on the client side are transmitted over the channel. Since these indices represent the corresponding codebook vector we have the necessary information and can perform the recognition. From this point of view it is obvious that a discrete HMM recognizer does not produce any additional error when used in a distributed environment.

4. RECOGNITION WITH CONTINUOUS FEATURES

The continuous HMM uses Gaussian mixture probability density functions (pdf) to model the output pdf of the feature vector given the HMM state.

$$p(\vec{x(t)}|\mathrm{state}\,i) = \sum_{j=1}^{J} c_{ij} \frac{1}{\sqrt{(2\pi)^n \sigma_i^2}} \mathrm{e}^{-\frac{x(\vec{t}) - \vec{m}_i}{2\sigma_i}} \tag{3}$$

Since we receive only VQ indices from the client, we have to *decode* the data by replacing the VQ label with the corresponding codebook vector (this assumes that the codebook vectors are known on the server side).
Having reconstructed the continuous feature vector we can then compute additional delta and acceleration coefficients to further improve the recognition result. The final feature vector possesses then 42 elements (14 “original” components plus delta coefficients plus acceleration coefficients). The HMM topology is the same as presented in section 3, we use (as in [5]) 3 mixtures per state for the whole word models and 6 mixtures per state for the silence model. The interword silence model with only one HMM state is tied to the center state of the other silence model.

5. HYBRID TIED-POSTERIOR RECOGNIZER

The tied-posterior recognizer (see [6]) uses a *neural network* (NN) that estimates posterior probabilities $Pr(j|x(\vec{t}))$ for certain output classes j from the input vectors $x(\vec{t})$. Here, the idea is to transmit the most important posterior probabilities over the channel and to compute the state-dependent probabilities for decoding on the server. Since we are using the same HMM topology as in section 3 the output classes from the neural net are "pseudo phonemes" that are formed by grouping 4 HMM states from the whole word model to one unit (the silence models form one pseudo phoneme per HMM state).

The input layer of the NN - for our experiments we used a *multi-layer perceptron* (MLP) - consists of the feature vector $\vec{f}$ (with delta and acceleration coefficients)[2] from the current frame t and $2m$ (we chose $m = 3$) adjacent frames. Thus, the input vector is $\vec{x} = (\vec{f}(t-m), \ldots, \vec{f}(t), \ldots, \vec{f}(t+m))$.

The resulting output layer size is $n_{out} = 48$ (44 pseudo phonemes from the word models plus 4 from the silence models). Transmitting 48 posterior probabilities (`float` values) would exceed the bandwidth, so we use only the n_p highest probabilities and skip the other ones (in [6] it is stated that the important information is stored in these probabilities). Furthermore, we quantize these values using a non-linear quantizer depicted in figure 4 that uses b_{np} bits per value. To meet the bandwidth we have to choose $n_p = 4$ probability values per frame with $b_{np} = 5$ bits for the probability value plus 6 bits for the class index.

$$\mathrm{BR}_{\mathrm{MLP}} = \frac{4 \cdot (5 + 6)\, bits}{10\,\mathrm{ms}} = 4.4\,\mathrm{kbit/s}$$

On the server side we use the inverse quantizer to receive

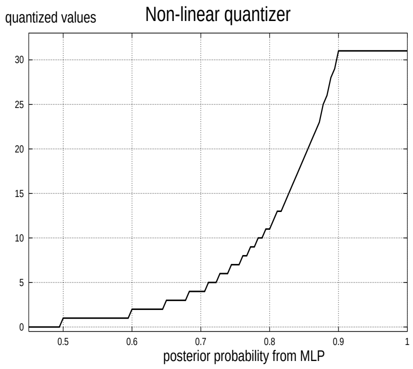

Fig. 4. Non-linear quantizer ($n_p = 4$, $b_{np} = 5$)

the original values again (of course, with some quantization error) and use these posterior probabilities as tied probabilities for all HMM states according to the following equations:

$$p(\vec{x}|S_i) = \sum_{j=1}^{J} c_{ij} \cdot \frac{Pr(j|\vec{x})p(\vec{x})}{Pr(j)} \tag{4}$$

where S_i is the HMM state, c_{ij} are the mixture coefficients ($\sum_{j=1}^{J} c_{ij} = 1$ and J is the number of pseudo-phonemes. Since $p(\vec{x})$ is independent of the HMM state S_i it can be omitted and (4) becomes

$$p(\vec{x}|S_i) \propto \sum_{j=1}^{J} c_{ij} \cdot \frac{Pr(j|\vec{x})}{Pr(j)} \tag{5}$$

The main advantage of the tied-posteriors approach in the DSR framework is the neural net's ability to concentrate the important class information in only a few probability values. Moreover we have information about the frame context processed in the NN and can extend this information without changing the amount of transmitted data. Current research activity investigates the inclusion of additional feature values e.g. extracted with the RASTA-PLP [7] algorithm.

6. RESULTS

We have evaluated the presented modeling techniques on two databases under different noise conditions:

- The TI digits database: This database contains single digits and digit chains spoken by native American speakers recorded with very low background noise (clean condition) and sampled with a frequency of 20 kHz. Additionally, we have added white Gaussian noise (WGN) with a signal-to-noise ratio (SNR) of 6 dB and 0 dB, respectively. These conditions allow a more distinguishable environment. The training and test sets are identical to the ones from the original TI digits database.

- The AURORA database [5]: This database is derived from the TI digits database, but augmented with different real noise types (e.g. babble noise, airport noise, etc.) and down-sampled to 8 kHz. The test sets contain known (from the training) and unknown noise types.

The following abbreviations are used in the result table:

- MFC42-MLP - 13 mel-cepstrum coefficients (including c_0) plus log. frame energy, with delta and acceleration coefficients, then the quantized posterior probabilities are computed

- MFC14-VQ7 - mel-cepstrum features (13 mel-cepstrum coefficients (including c_0) plus log. frame energy)

[2] this time we compute them on the client side

quantized to seven vector indices (System 1 in figure 2)

- MFC14-VQ7 MMI same as above, but the codebook is generated using a MMI-NN vector quantizer
- WER - word error rate

The continuous (cont.) and the tied-posterior (tied-post.) recognizers always use MFC42 on the server side.
From the AURORA database we have only used the multicondition training set [5]. The first column of all tables ("feature extraction") describes the feature extraction method on the client side, the second column ("recognizer") shows the type of the HMM recognizer.

feature extraction	**recognizer**	**SNR (dB)**	**WER (%)**
MFC14-VQ7	**cont.**	**clean**	**0.91**
MFC14-VQ7	discrete	clean	3.94
MFC14-VQ7 MMI	discrete	clean	3.85
MFC14-VQ7 MMI	cont.	clean	0.94
MFC42-MLP	tied-post.	clean	1.86
MFC14-VQ7	cont.	6	4.69
MFC14-VQ7	discrete	6	11.98
MFC14-VQ7 MMI	discrete	6	11.61
MFC14-VQ7 MMI	cont.	6	5.26
MFC42-MLP	**tied-post.**	**6**	**4.14**
MFC14-VQ7	cont.	0	10.15
MFC14-VQ7	discrete	0	20.51
MFC14-VQ7 MMI	discrete	0	20.07
MFC14-VQ7 MMI	cont.	0	10.00
MFC42-MLP	**tied-post.**	**0**	**7.75**

Table 1. Results on TI digits' test set

feature extraction	**recognizer**	**Test set (%)**	**WER (%)**
MFC14-VQ7	cont.	A	12.23
MFC14-VQ7	discrete	A	34.96
MFC42-MLP	**tied-post.**	A	**10.16**
MFC14-VQ7	**cont.**	B	**14.23**
MFC14-VQ7	discrete	B	33.67
MFC42-MLP	tied-post.	B	17.03

Table 2. Results on the AURORA test sets A and B (multicondition training)

Comparing the tied-posterior recognizer with the continuous recognizer shows that the hybrid approach achieves a better performance (see bold results in table 1 and 2), if the statistical properties of the noise is known in the training process. In case of the TI digits database the MMI approach shows a slightly better performance than the k-means algorithm if noise is added and the recognizer uses continuous features. The discrete recognizer cannot compete with the other algorithms at all.

7. CONCLUSION

We have compared different acoustic modeling techniques for DSR systems. The results have been evaluated on the TI digits database under clean and noisy conditions and on the AURORA database. We achieved an improvement of the word error rate of 24% relative compared to a standard Gaussian recognizer, if using a hybrid tied-posterior recognizer and artificial noise added with a SNR of 0 dB. Under real noise conditions we reduced the error rate by 17% relative if the noise conditions for training and testing are similar.

8. REFERENCES

[1] K. David and T. Benkner, *Digitale Mobilfunksysteme*, Teubner, 1996.

[2] ETSI standard document, "Speech Processing, Transmission and Quality aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms," in *ETSI ES 201 108 v1.1.1 (2000-02)*, 2000.

[3] Gerhard Rigoll, "Maximum Mutual Information Neural Networks for Hybrid Connectionist-HMM Speech Recognition Systems," *IEEE Transactions on Speech and Audio Processing, Special Issue on Neural Networks for Speech*, vol. 2, no. 1, pp. 175–184, Jan. 1994.

[4] Cristoph Neukirchen and Gerhard Rigoll, "Advanced Training Methods and New Network Topologies for Hybrid MMI-Connectionist/HMM Speech Recognition Systems," in *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Munich, Apr. 1997, pp. 3257–3260.

[5] H. G. Hirsch and D. Pearce, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ISCA ITRW ASR2000*, 2000.

[6] Jörg Rottland and Gerhard Rigoll, "Tied Posteriors: An Approach for Effective Introduction of Context Dependency in Hybrid NN/HMM LVCSR," in *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Istanbul, Turkey, June 2000.

[7] H. Hermansky and N. Morgan, "RASTA Processing of Speech," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, pp. 578–589, 1994.