IBM's 10x Real-time Broadcast News Transcription System Used in the 1999 Hub4 Evaluation

E. Eide, B. Maison, M. Gales, R. Gopinath, S. Chen, P. Olsen, D. Kanevsky, M. Novak, and L. Mangu

I.B.M. T.J. Watson Research Center
P.O. Box 218 Yorktown Heights, NY 10598 U.S.A.

ABSTRACT

We describe the system used by IBM in the 1999 HUB4 Evaluation under the 10 times real-time constraint. We detail the system architecture and show that the performance of this system is over 20 percent more accurate at the same speed than the system used in the 1998 Evaluation. Furthermore, we have closed the gap between our unlimited resource system and our 10 times real time system from 45 percent to 14 percent.

1. Introduction

In this paper we report on the 10xRT system run by IBM in the 1999 Hub4 evaluation, giving contrastive results with other system architectures we had considered as well as with the unconstrained system run in the other portion of the evaluation. Because the 10xRT constraint is somewhat arbitrary in that it is machine dependent, we have chosen to fix our machines to be those used in the 1998 Hub4 Evaluation; all programs are compiled for the AIX platform and all experiments were conducted on a 320MIPS RS/6000 SP2 node with 512MB of memory. As these are exactly the same resources we used in the 1998 evaluation, all system improvements are code and algorithmically based.

For comparison, we show our performance numbers in the 1998 Hub4 evaluation (test sets 1 and 2) in table 1 for both the baseline transcription system (A1,A2) and the baseline 10xRT system (B1,B2).

	Avg	F0	F1	F2	F3	F4	F5	FX
A1	14.5	7.8	16.8	20.9	24.7	10.0	19.4	19.7
A2	12.4	8.4	14.7	14.3	12.7	14.1	5.7	34.4
B1	21.2	11.4	21.8	33.7	32.6	14.8	26.1	31.5
B2	17.8	10.8	19.4	24.4	20.5	21.0	15.7	54.1

Table 1: Performance on the 1998 Hub4 Evaluation Test Data of the 1998 unlimited broadcast news transcription system and the 1998 10xRT system. A1=Baseline unconstrained system, test set 1. A2=Baseline unconstrained system, test set 2. B1=10xRT, test set 1. B2=10xRT, test set 2. F0=Clean,planned speech. F1=spontaneous speech. F2=Speech over telephone channels. F3=Speech with background music. F4=Speech with degraded acoustics. F5=Non-native speakers. FX=Combinations of F1-F5.

2. System Architecture

Several changes to the system architecture used for the baseline broadcast news transcription system were necessary in order to arrive at a system which would run in less than ten times real time.

The first difference we made was to change from the research code base to one closer to IBM's commercial product, ViaVoice, which has been algorithmically optimized for efficient execution. The code includes an improvement in the evaluation of the phonetic tree that represents the entire vocabulary of the recognizer. The acoustic fast match may be done much more efficiently using the fact that under certain conditions the results of branch evaluations can be used to approximate the scores of other branches of the tree [5]. The same approach has been used to speed up the detailed match [6].

Another difference between our 10xRT system and our baseline broadcast news system lies in what context is taken into account to model each phoneme. In our baseline system we consider 5 phonemes to the left and to the right of a given phoneme in building the acoustic models for that phoneme even across the end of the word under consideration, while in our 10xRT system we do not consider phonemes to the right of the word boundary when building the acoustic model for a given phoneme. This restriction eliminates the need for recomputing the acoustic observation probabilities near the end of each word as the hypothesized word string becomes available as the search proceeds towards the end of the sentence.

Another difference between the 10xRT and baseline broadcast news systems is that for the 10xRT system the Gaussian prototypes are arranged hierarchically; only those Gaussians which score well at a given level of the hierarchy are expanded for consideration at lower levels.

The architecture of the baseline transcription system relies on Rover [2]; several different recognition systems are run and a vote is taken on the recognition outputs to produce a final transcription. Furthermore, multiple passes of adaptation are performed within each of the systems upon which Rover operates. This framework proved too costly for the ten-times-real-time constraint; for the 10xRT system we operate only

a single system rather than a Rover paradigm and the system consists only of a rapid first pass, adaptation, and a single, more detailed second pass rather than multiple iterations of adaptation with re-decoding.

We have found a first pass running at roughly two times real time, an adaptation phase running at roughly three times real time, and a second pass running at approximately five times real time to perform well.

3. Acoustic Training

In this section we describe the construction of the speaker-adapted training (SAT) model and give performance numbers for the final model versus other models tested. The SAT training algorithm transforms the SI model means for each speaker; with the adapted means a full-variance linear transform that maximizes the likelihood of the data from that speaker is computed [1]. The computation can be implemented efficiently as a feature space transformation [1]. A single iteration of SAT training consists of computing a transformation for each training speaker and then performing two iterations of the EM-algorithm to adjust the speaker-adapted models; the final model used in the evaluation was the result of two iterations of SAT training.

	Avg	F0	F1	F2	F3	F4	F5	FX
C1	16.3	8.7	18.7	27.9	25.0	10.3	23.0	22.8
C2	13.6	8.6	15.3	21.0	14.5	15.9	5.7	37.4
D1	16.4	8.6	18.7	26.1	24.3	10.8	22.4	23.2
D2	13.5	8.4	15.4	20.4	14.9	15.9	5.7	38.2
E1	16.2	8.3	18.6	26.1	26.4	10.8	20.6	22.7
E2	13.3	8.2	15.1	20.3	14.0	15.6	5.7	38.0

Table 2: Performance on the 1998 Hub4 evaluation sets 1 and 2 using decoded training data transcriptions to perform one iteration of SAT training (C1,C2) versus one (D1,D2) and two (E1,E2) iterations of SAT training using the true transcriptions. Hand segmentation of the test is used in all cases.

We also considered decoding the training data and using the decoded script rather than the truth to calculate the transformation for each speaker, so as to more closely match the testing procedure. The truth continued to be used for the EM processing. Results of this experiment for a single iteration of SAT training are shown in the rows C1 and C2 of table 2 and are to be compared with the baseline experiment of using the true transcription for computing the transformation for each speaker and running a single iteration of SAT training, shown in the rows D1 and D2 of the table. Rows E1 and E2 of the table show the improvement over baseline D1 and D2 by running a second iteration of SAT training using the true transcription in calculating speaker transforms. It is this model which was used in the 1999 Hub4 Evaluation.

4. Language Model

We used different language models for the first and second pass decodes. For the first pass we used a mixture of three components, two trigrams and one maximum entropy model. For the second pass decode we used a larger model comprised of six components, the three from the first pass plus one additional trigram and two additional maximum entropy models. Mixture weights were chosen to minimize perplexity on a development test set.

5. Decoding

The first decoding results we present will be to justify the two-pass architecture of our 10xRT system. We compare a single decoding pass tuned to run at ten times real time with the SAT system wherein the most costly step, the second-pass decode, runs at less than 4.9 times real time. The results, shown in table 3, clearly justify our choice of the SAT architecture over a single-pass decode.

	Avg	F0	F1	F2	F3	F4	F5	FX
G1	17.9	8.9	19.6	30.2	27.3	12.3	20.6	25.7
G2	14.8	9.4	17.0	21.1	13.9	18.2	8.6	38.2
H1	16.2	8.3	18.6	26.1	26.4	10.8	20.6	22.7
H2	13.3	8.2	15.1	20.3	14.0	15.6	5.7	38.0

Table 3: Performance on the 1998 Hub4 evaluation data sets 1 and 2, comparing a single pass decode running at ten times real time (G1,G2) with the SAT architecture in which the most costly step runs at less than 4.9 times real time (H1,H2). Hand segmentation is used in all cases.

The remaining subsections in this section describe in more detail the individual steps required to decode in "evaluation mode" the data of the 1998 or 1999 Hub4 evaluation. Because the data is provided as one long continuous audio stream, we first segment it into manageable chunks, identifying and discarding regions of pure music in the process, as described in section . The segments are then clustered according to the algorithm described in section for the purposes of accumulating enough self-similar data within each cluster to robustly estimate transformations for adaptation. The segmentation, music detection, and clustering together run at less than 0.3 times real time. Having segmented and clustered the data, a rapid first pass is run as outlined in section , followed by two passes of transformation estimation for the data in each cluster as described in section . The more-detailed, second pass decode which makes use of the adaptation transformation calculated for each cluster is described in section .

5.1. Segmentation and Music Detection

The Bayesian Information Criterion (BIC) is used to detect acoustic changes in the data [3]; the unpartitioned audio stream is divided into segments based on the times at which

changes are detected. Once segmented, the data is classified as one of five acoustic conditions, one of which is pure music, by means of a Gaussian-mixture classifier [4]. The single model for music segments competes with four models of speech in various noise levels and conditions; all five of the mixture models consist of 156 Gaussians. Those segments identified as pure music are discarded from further processing. The effect of automatic segmentation is fairly severe, as seen by comparing the results in table 4 with the hand segmentation baseline (H1,H2) presented in table 3. The segmentation and music detection step runs at 0.2 times real-time.

	Avg	F0	F1	F2	F3	F4	F5	FX
I1	16.8	8.6	19.0	27.7	25.2	11.2	19.4	24.0
I2	14.1	8.6	15.9	19.9	15.3	16.3	1.4	46.2

Table 4: Effect of automatic segmentation on the 1998 Hub4 evaluation data sets 1 and 2. Compare with baseline hand segmentation (table 3 rows H1,H2).

5.2. Clustering

After segmentation, clustering is performed in order to accumulate enough data to robustly perform adaptation, with one adaptation transformation estimated for each cluster. The segments are clustered using a maximum-linkage, bottom-up clustering procedure with a single Gaussian model for each segment and a log-likelihood-ratio distance measure [3]. The bottom-up clustering procedure terminates where the BIC criterion reaches its maximum. The real-time factor is approximately 0.1.

5.3. First Pass Decode

The first pass decode, whose output serves as the input script for adaptation, is tuned to run at slightly less than 1.8 times real time. The system uses the same 286K-Gaussian, left-context system as the more detailed second pass decode which runs at slightly less than five times real time. The differences between the two systems lie in the language model as described in section 4, a more aggressive hierarchy in the first pass than in the second pass, and more aggressive pruning in the search of the first pass decoder than in the second. Results of the first pass alone on the unpartitioned evaluation data of 1998 are shown in table 5.

	Avg	F0	F1	F2	F3	F4	F5	FX
J1	21.2	10.9	22.0	37.0	33.8	15.4	24.8	30.2
J2	17.5	10.7	19.2	23.5	19.1	20.8	11.4	53.4

Table 5: Performance of the first pass decode (J1,J2) on the data from the 1998 Hub4 evaluation test sets 1 and 2. Automatic segmentation is used.

By comparing table 5 with rows B1 and B2 of table 1 we note that the performance from this first-pass decode is already better than that obtained by our 10xRT system used in the 1998 Hub4 evaluation.

5.4. Transform Computation

The transformation calculation detailed in [1] proved to run too slowly for the ten-times-real-time constraint. We made several algorithmic approximations to increase its speed, as will be described in this section. The first approximation was in the computation of observation probabilities. Rather than summing over all Gaussians in a mixture, we approximate the sum with the maximum probability from the individual Gaussians within the cluster. The second approximation is introduced in the trellis calculation. Rather than summing over all predecessor nodes, we again approximate the sum with the maximum of the individual members going into the summation. Both of these approximations eliminate the need for a costly linear addition in the log domain. Further gains in speed were obtained by thresholding the number of counts attributed to a Gaussian before including it in the transformation calculation.

One additional method of speeding up the adaptation is to perform a block-diagonal transformation rather than a full matrix one. We tried constraining the transformation to consist of two blocks and found an increase in speed from 1.5xRT to 0.8xRT for each iteration at the cost of lack of recognition accuracy, especially in the F0 and F1 conditions, as shown in table 6.

	Avg	F0	F1	F2	F3	F4	F5	FX
K1	16.8	8.3	18.7	27.7	32.6	10.7	21.8	24.1
K2	13.5	8.5	15.5	19.6	14.2	16.1	4.3	36.9
L1	16.5	8.9	19.4	25.7	25.0	11.1	21.2	22.5
L2	14.0	8.8	15.8	17.5	14.2	16.9	11.4	38.7

Table 6: Performance using a full-matrix transformation (K1,K2) vs. a 2-block diagonal transform for adaptation (L1,L2) on test sets 1 and 2 from the 1998 Hub4 evaluation after one pass of EM-training. Hand segmentation is used in all cases.

Although the overall degradation in performance is perhaps acceptable given the increase in speed, we decided not to pursue the use of a two-block transform in our 10xRT system due to its severe impact on the F0 and F1 conditions.

Another consideration was the number of iterations run in calculating the transform for the test set. We found that performance for a second iteration increased over the first iteration and then stayed flat for the third as indicated in table 7, so we opted to run two iterations in test.

	Avg	F0	F1	F2	F3	F4	F5	FX
M1	16.5	8.5	18.7	26.7	25.8	10.7	21.8	23.7
M2	14.2	8.3	16.3	20.3	14.6	17.6	8.6	39.3
N1	16.4	8.6	18.7	26.1	24.3	10.8	22.4	23.2
N2	13.5	8.4	15.4	20.4	14.9	15.9	5.7	38.2
P1	16.3	8.7	18.8	25.5	23.3	10.7	23.0	23.1
P2	13.6	8.4	16.2	20.4	14.2	15.8	5.7	37.7

Table 7: Performance for 1 (M1,M2), 2 (N1,N2), and 3 (P1,P2) passes of calculating the adaptation matrix for each cluster on the data from the 1998 Hub4 evaluation. Hand segmentation is used in all cases.

5.5. Second Pass Decode

The second pass decode makes use of the feature-space transformation calculated for each cluster. Although it uses the same acoustic models as the first pass, it uses a larger language model and more costly search parameters. The final system performance on the unpartitioned 1998 Hub4 evaluation data is shown in rows Q1 and Q2 of table 8, reflecting an improvement in performance of 21.8% over last year (table 1, B1 and B2).

6. Lattice-Based Word Error Minimization

This step was not included in the 1999 evaluation but was found to significantly improve to performance within the ten times real-time constraint. It uses the word lattices generated by the second pass decoding in order to find the words with maximal posterior probabilities (the consensus hypothesis), thereby reducing the Word Error Rate [7]. Rows R1 and R2 of table 8 summarize the results and are to be compared with rows Q1 and Q2. This improvement of 3 percent brings the overall improvement over last year's system to 24 percent.

	Avg	F0	F1	F2	F3	F4	F5	FX
Q1	16.5	8.3	18.6	27.9	26.2	10.7	22.4	23.7
Q2	14.0	8.6	15.8	19.4	15.3	16.0	5.7	44.8
R1	16.0	8.5	18.1	26.1	25.8	10.5	18.8	22.5
R2	13.6	8.7	15.7	18.6	14.6	15.3	5.7	41.1

Table 8: Performance on the 1998 Hub4 Evaluation Test Data of the 1999 10xRT system and that system with lattice word consensus. Q1,Q2=1999 10xRT system. R1,R2=1999 10xRT system with lattice word consensus.

7. Performance on 1999 Evaluation Data

In this section we show the performance of our systems on the 1999 Evaluation data.

In table 9 line (S1,S2) we show the performance of our 1999 Unlimited resource system, followed by that of our 10xRT system in lines (T1,T2). Lines (U1,U2) indicate the performance of the 10xRT lattice word consensus system.

	Avg	F0	F1	F2	F3	F4	F5	FX
S1	15.8	7.7	16.0	27.6	19.0	12.6	22.2	42.1
S2	14.5	8.1	18.3	12.0	12.0	12.7	11.9	32.9
T1	18.3	9.1	17.3	35.7	20.7	14.2	22.2	51.0
T2	17.1	8.6	18.4	17.9	12.0	15.2	14.6	41.0
U1	17.8	9.0	17.2	34.6	20.8	14.2	19.4	46.5
U2	16.7	8.3	18.5	17.1	12.4	15.0	14.4	39.7

Table 9: Performance on the 1999 Hub4 Evaluation Test Data of our 1999 systems. S1,S2=Unlimited resource system. T1,T2=10xRT system without lattice word consensus. U1,U2=10xRT system with lattice word consensus.

By comparing the degradation incurred by the ten times real time constraint in the 1999 evaluation [(S1,S2) vs. (U1,U2)] with that incurred in 1998 [(A1,A2) vs. (B1,B2)], we note that we have closed the gap between the unconstrained and the 10xRT system from 45 percent to 14 percent.

References

1. M. J. F. Gales, "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition", *Technical Report Cambridge University*, CUED/FINFENG/TR291, 1997.
2. J. G. Fiscus, "A post–processing system to yield reduced word error rates: recognizer output voting error reduction (rover)", *Technical Report National Institute of Standards and Technology*, 1997.
3. S. Chen et al., "Speaker, Environment and Channel Change Detection and Clustering via the Bayesian Information Criterion", *Proc. of DARPA Speech Recognition Workshop*, Feb 8-11, Lansdowne VA, 1998.
4. L. R. Bahl et al., "Performance of the IBM large vocabulary continuous speech recognition system on the ARPA Wall Street Journal task", *Proc. ICASSP*, pp. 41–44, 1995.
5. M. Novak and M. Picheny, "Speed improvement of the time-asynchronous acoustic fast match", *Proc. Eurospeech*, Vol. 3, pp.1115-1118, 1999.
6. M. Novak and M. Picheny, "Speed improvement of the tree-based time-asynchronous search", *Proc. ICSLP*, 2000.
7. L. Mangu, E. Brill and A. Stolcke, "Finding Consensus Among Words: Lattice-Based Word Error Minimization", *Proc. Eurospeech*, 1999.