

# TOWARDS AUTOMATIC CLOSED CAPTIONING : LOW LATENCY REAL TIME BROADCAST NEWS TRANSCRIPTION

*Murat Saraclar, Michael Riley, Enrico Bocchieri, Vincent Goffin*

AT&T Labs – Research  
180 Park Ave, Florham Park, NJ 07932 USA  
{murat,riley,enrico,vjg}@research.att.com

## ABSTRACT

In this paper, we present a low latency real-time Broadcast News recognition system capable of transcribing live television news-casts with reasonable accuracy. We describe our recent modeling and efficiency improvements that yield a 22% word error rate on the Hub4e98 test set while running faster than real-time. These include the discriminative training of a feature transform and the acoustic model, and the optimization of the likelihood computation. We give experimental results that show the accuracy of the system at different speeds. We also explain how we achieved low latency, presenting measurements that show the typical system latency is less than 1 second.

## 1. INTRODUCTION

Continuing advances in automatic speech recognition coupled with ever increasing computing power has made real-time large vocabulary speech recognition a reality. However, we are still far from being able to replace humans by computers even for real-time closed captioning where the human word error rate is about 10%.

There has been considerable amount of research in Broadcast News Transcription in recent years, thanks to the DARPA Hub 4 evaluations [1, 2]. The hub systems were developed to maximize accuracy and usually run in excess of 100 times real-time (100xRT). There was also a spoke for 10xRT systems and various sites developed systems that could run at faster than 10xRT with small degradations in accuracy. There were even a few with acceptable accuracies running at real-time. However, we have found only one reference [3] to any latency figures, and that system had a mean latency of 6 seconds at an error rate of 54.4%.

We have created a prototype system that runs faster than real-time with typical latencies less than one second and has 22% word error rate. In this paper, we present our recent modeling and efficiency improvements that have enabled us to move towards a low-latency real-time Broadcast News transcription system. In Section 2 we give a description of our baseline system. Some improvements to the front end and the acoustic models are presented in Section 3. The techniques we use were previously shown to be effective in other tasks where recognition speed was not a constraint. Here we investigate the behaviour of these methods at varying speeds. This is followed by improvements in the likelihood computation time given in Section 4. Next, in Section 5, we describe the enhancements which resulted in our low-latency real-time system and present some latency measurements.

## 2. BASELINE SYSTEM

The starting point for our system was AT&T's TREC-8/SDR99 submission [4]. The front-end computes 12 mel-frequency cepstral coefficients and energy, as well as the first and second derivatives, to yield a 39 dimensional vector per frame, at a rate of 100 frames per second. For feature normalization, cepstral mean subtraction and maximum value energy normalization are applied to the feature vectors. The acoustic models were trained on 140 hours of Hub4 Broadcast News speech data and consist of 11k HMM states and 134k gaussians. For the duration model, state or HMM level gamma distributions were used. The language model is a pruned backoff trigram model trained on a mixture of Hub4 data and the NAB corpus. The vocabulary of the system contains over 210k words. More details can be found in [4].

Our baseline system was previously used for audio indexing and spoken document retrieval purposes and is no longer state-of-the-art in terms of efficiency and accuracy.

## 3. MODELING IMPROVEMENTS

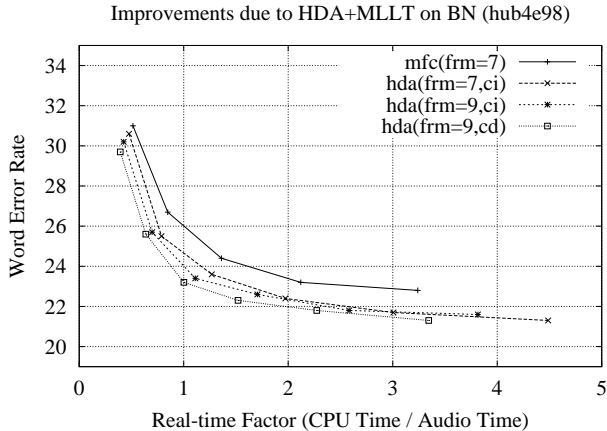
In this section we describe the modeling improvements to this baseline system and give experimental results, which are presented as word error rate versus real-time factor plots obtained by varying the decoder beam. We use the Hub4e98 test set [1] throughout the paper for our evaluations. Unless noted otherwise, the experiments in this paper were run on a 1.6 GHz Pentium P4-Xeon.

We have significantly increased the accuracy of our system by improving the front end and the acoustic model. Both improvements come from using discriminative techniques.

### 3.1. Discriminative Front End

The mel frequency cepstrum coefficients and frame energy, with first and second order time derivatives (mfc-dd) are widely used frontend features. It is useful to conceptualize the mfc-dd vector as a linear transformation of a measurement vector, assembled by concatenating a number of consecutive cepstrum and energy vectors, as needed for estimation of the derivatives. The issue arises whether it is possible to estimate a better transform, giving more accurate feature vectors.

We have confronted this problem with a discriminative feature extraction technique known as heteroscedastic discriminant analysis (HDA) [5], a particular formulation of [6]. Assuming a number of recognition classes with arbitrary gaussian distributions, the HDA transform provides features that maximize a ratio of between-class and within-class distortion measures. We also



**Fig. 1.** Improvements due to HDA+MLLT (with 39 features) on hub4e98

employ a maximum likelihood linear transform (MLLT) [5, 7, 8], to ensure minimum loss of likelihood with diagonal covariance HMM. We have investigated: (a) the advantage of HDA+MLLT over standard mfc-dd features, (b) the choice of the number of consecutive cepstrum vectors that are input to the HDA transform, (c) the choice of "recognition class" for HDA+MLLT estimation and (d) the choice of the number of HDA+MLLT features to use.

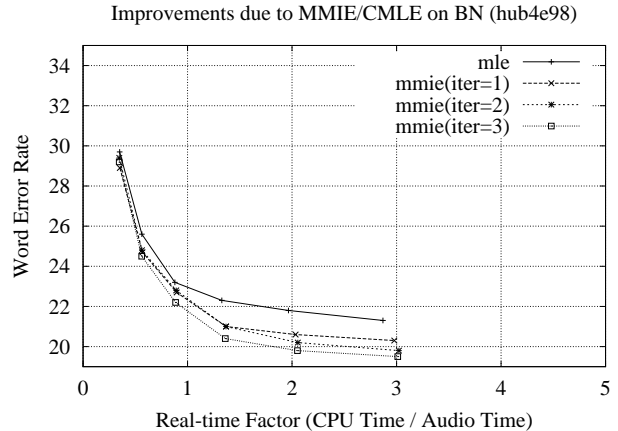
The results of our experiments are presented in Figure 1, with 12 cepstrum coefficients and energy for every speech frame, and feature vectors of 39 dimensions for both mfc-dd and HDA+MLLT features. Results are shown with either seven or nine consecutive cepstrum and energy vectors as input to the HDA+MLLT transform, and either context independent (ci) or context-dependent states (cd) as classes for HDA+MLLT estimation.

In our implementation, standard mfc-dd uses seven consecutive frames to estimate the derivative components. Therefore, curves mfc-dd(frm=7) and hda(frm=7,ci) show the advantage of HDA+MLLT over standard mfc-dd, under comparable conditions. The error rate is reduced further, and the recognition speed is improved, using nine consecutive frames (curve hda(frm=9,ci)), and context dependent states for HDA+MLLT transform estimation (curve hda(frm=9,cd)).

We have also tried using more than 39 HDA+MLLT features (not shown in Fig 1). Going to 60 features helps the asymptotic performance (0.5% absolute improvement), but for speeds faster than 1.5xRT, 39 features give the best accuracy.

### 3.2. Discriminative Acoustic Models

Recently, lattice-based Maximum Mutual Information Estimation (MMIE) has been shown to be effective in large vocabulary continuous speech recognition tasks such as North American Business News (NAB) and Switchboard [9]. Our implementation is similar to the one in [9], but differs in how the necessary statistics are collected. In the previous method phone-marked lattices are generated once and a full forward-backward search is performed on these lattices for each iteration. In contrast, for each iteration we generate state-level lattices from the word lattices and perform a search constrained by these lattices. The training typically requires 0.4xRT per iteration on Intel Pentium PIII running at 1GHz.



**Fig. 2.** Improvements due to multiple iterations of MMIE/CMLE on hub4e98

In Figure 2 we present results for multiple iterations of MMIE training starting from an MLE trained model. At each iteration only the gaussian means and variances are updated. The gain from MMIE is larger at higher beams, but a significant gain can also be observed, as iterations progress, at lower beams.

## 4. EFFICIENCY IMPROVEMENTS

Computation time in speech recognition is divided between the traversal of the recognition search space and the acoustic model likelihood calculation. We optimized the search space using the methods described in [10, 11, 12]. In this section, we describe optimizations to the likelihood evaluation.

The computation of the (negative) log likelihood of the  $j$ -th diagonal gaussian  $\mathcal{N}(\vec{\mu}_j, \vec{\sigma}_j)$  has the form:

$$-\log P_j = k_j + \sum_{i=1}^N \frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2} \quad (1)$$

where  $N$  is the dimension of the acoustic feature vector. For moderate search beams, between 10% to 20% of the 134k gaussians in our model need to be calculated for each frame. Profiling shows that much of the time spent in computing Eq. 1 is not in the floating point operations, but in fetching the model data from main memory to the Level 2 cache. This is because the combined model and search space references per frame overflow the L2 cache. To ameliorate this problem, we use a *likelihood batch* strategy that computes and stores a requested gaussian likelihood not just for the current frame but for  $K - 1$  future frames as well, with  $K = 8$  in our current system. This is beneficial since each gaussian's parameters only have to be fetched once per batch and since most of the future likelihoods will be needed due to the continued activation of the corresponding search state (greater than 75% will be needed at moderate beams). Table 3 shows the speed up in total recognition time on a 1 GHz Pentium III with 256 kb L2 cache.

An added advantage of batching the likelihoods is that the compiler can, in principle, pipeline the computation and pre-fetch the data. Unfortunately, not all compilers (we use GNU g++ under Linux) take the best advantage of the hardware for this kind of computation (partly due to potential pointer aliasing in C/C++).

Method	$\times$ Real-Time
baseline	3.12
batch ( $K = 8$ )	2.08
P2 BLAS	1.87
P3 BLAS	1.59

**Fig. 3.** Likelihood computation optimizations: total recognition real-time factor at a moderate beam for several likelihood computation methods on a 1 GHz Pentium P3 with 256 kb L2 cache.

One alternative is to write the likelihood 'inner loop' in assembly language. There is, however, an attractive alternative – the *Basic Linear Algebra Subprogram Library (BLAS)* [13]. The BLAS library is the computation kernel of the *LinPack* and *LaPack* linear algebra libraries and there exist highly-optimized versions of BLAS for a variety of architectures. The computation in Eq. 1 is not a BLAS library routine. However, we can recast Eq. 1 as:

$$-\log P_j = \sum_{i=1}^{2N+1} \alpha_{ij} y_i \quad (2)$$

where  $\vec{y} = (1, x_1, x_2, \dots, x_N, x_1^2, \dots, x_N^2)$  and the  $\alpha_{ij}$  are the coefficients in the quadratic expansion of Eq. 1. Dot product is a component of BLAS. Since we compute all the  $M$  (e.g., 12) Gaussians in a mixture and we compute  $K$  frames in a batch, the computation in Eq. 2 for all these likelihoods becomes the product of a  $M \times N$  matrix with a  $N \times K$  matrix. Matrix multiply is a central component of the BLAS library. In Table 3, we compare two different Intel Linux BLAS libraries in the likelihood computation – one optimized for the Pentium II architecture and a second with Pentium III enhancements [14]. We have also tried these techniques on Intel P4, SGI, Sun, and Compaq Alpha computers and have seen significant improvements as well although the relative and absolute contributions depend on the hardware, the compiler used, and the quality of the BLAS implementation.

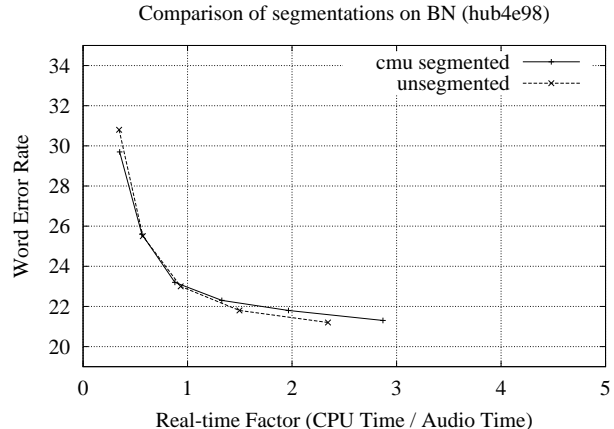
We used the P3 BLAS-based likelihood calculation for the other experiments reported in this paper. The real-time factor on the 1.6 Ghz P4 is 1.26 compared to 1.59 on the 1 GHz P3.

## 5. LATENCY IMPROVEMENTS

With the techniques described in the previous two sections, we were able to get a real-time system with an improved error rate. However, real-time performance does not guarantee low latency. For example, if the system has a preprocessing component that operates on the whole input, then the latency will be at least the length of the input, no matter how fast the system.

The system latency is defined as the time delay between the input and the output of the system. In our case, this is the time it takes for the system to display a word that was spoken. Latency has a fixed component due to the lookahead, and a variable component due to the search. We have only found a few examples of latency measurements [3, 15] in the speech recognition literature.

We now describe the modifications we had to make to reduce the latency of the system. We will then provide sources of fixed latency and give measurements of the variable latency.



**Fig. 4.** Comparison of systems with and without an acoustic segmenter on hub4e98

### 5.1. Achieving Low Latency

The system described in the previous sections uses an acoustic segmenter (the CMU segmenter [16]) to process the audio input and generate small segments that are then fed into the front-end. The CMU segmenter (out-of-the-box) operates on full shows, thus its latency is the length of the show. Since most segmenters use a distance metric between two adjacent speech windows of to decide on a segment boundary, the latency is at least the window length, which is typically 2 to 4 seconds. Another segmentation method is to use an energy measure to decide on the boundaries, but that can be error prone.

To avoid sentence segmentation latencies and errors, the recognition network was replaced with its closure, which permits the recognition of any number of sentences as a single input utterance, thus making end-pointing unnecessary. The closure of the network is obtained by adding an epsilon arc (a "free" transition<sup>1</sup>) from the final states to the start state. Therefore we are not forcing any silence to be present at segment boundaries. As seen in Figure 4, the new system works as well as the previous system which uses an acoustic segmenter.

The decoder was modified to give the common prefix of the hypotheses it is considering as a partial output, as soon as the common prefix is nonempty. This computation is done efficiently at garbage collection time which happens every 0.5 seconds. This avoids memory exhaustion by discarding the memory used for the prefix. In fact, after the modification, the system is able run for a day on a continuous input stream with only a small increase in memory usage.

Another latency source is the front-end normalization. Most systems use per segment feature normalization, which increases the latency by the size of each segment, typically up to 30 to 60 seconds. Our front-end normalization works in a causal fashion except for a small lookahead of 0.3 seconds.

Finally, limiting the number of active arcs during decoding proved to be a good method for reducing the standard deviation of the real-time factor and hence for reducing the latency.

<sup>1</sup>We also experimented with adding a penalty to the closure transitions, but concluded that such a penalty was not helpful.

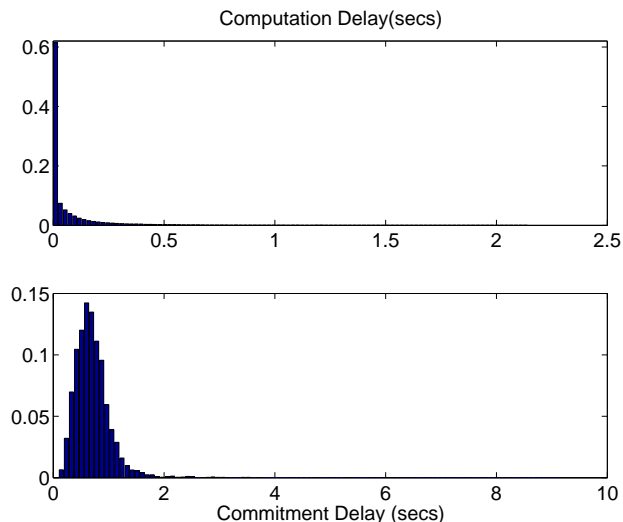


Fig. 5. Histograms of variable latency measurements on hub4e98-set1.

## 5.2. Quantifying Latency

Our system has several sources of fixed latency, all in the form of lookahead, excluding the startup cost of loading of the network and the acoustic models into memory. The front-end uses a 200 msec window for feature computation and a lookahead of 300 msec for feature normalization. The decoder has a lookahead of 7 frames (70 msec).

For the variable component of latency, we measured two different quantities – computation delay and commitment delay. Computation delay is defined as the difference between the elapsed time and audio time measured at the end of each frame of audio. Of course, since the audio is input to the system in real time the computation delay can never be negative. We define commitment delay as the duration of each partial output measured whenever the decoder has an output. Note that this number is shifted (on average by 0.25 sec) since the decoder can only produce a new output every 0.5 second.

In Figure 5, histograms of computation and commitment delays measured on the Hub4e98 Test Set 1 are given for the system. The histograms show that although the maximum latency can be higher, 90% of the time the computation delay is under 0.22 secs and the commitment delay is under 1.05 secs, and 99% of the time the computation delay is under 0.93 secs and the commitment delay is under 1.78 secs. Since commitment delay is much larger than the computation delay, in our final system we display the best hypothesis so far at each frame, which is subject to change, until a partial hypothesis is committed.

## 6. CONCLUSION

In this paper we have presented a low latency real-time Broadcast News transcription system. We have described the modeling and efficiency improvements that resulted in a real-time system with improved accuracy. The word error rate of the system was decreased by more than 4% absolute at speeds faster than real-time. We have also explained how we achieved a typical system latency of less than a second through various enhancements.

## 7. ACKNOWLEDGEMENTS

The authors thank Michiel Bacchiani, Don Hindle, Andrej Ljolje and Mehryar Mohri for the baseline system. We also thank Greg Henry of Intel for the BLAS library.

## 8. REFERENCES

- [1] *Proc. of the DARPA Broadcast News Workshop*, 1999, <http://www.nist.gov/speech/publications/index.htm>.
- [2] *Proc. of the Speech Transcription Workshop*, 2000, <http://www.nist.gov/speech/publications/index.htm>.
- [3] G. Cook, J. Christie, P. Clarkson, M. Hochberg, B. Logan, A. Robinson, and C. Seymour, “Real-time recognition of broadcast radio speech,” in *Proc. ICASSP*, 1996, pp. 141–144.
- [4] A. Singhal, S. Abney, M. Bacchiani, M. Collins, D. Hindle, and F. Pereira, “AT&T at TREC-8,” in *Proceedings of the Eight Text REtrieval Conference (TREC-8)*, 1999, pp. 317–330.
- [5] G. Saon, M. Padmanabhan, R. Gopinath, and S.Chen, “Maximum likelihood discriminant feature spaces,” in *Proc. ICASSP*, 2000, pp. 1129–1132.
- [6] N. Kumar and G. Andreou, “Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition,” *Speech Communication*, vol. 26, pp. 283–297, 1998.
- [7] R.A. Gopinath, “Maximum likelihood modeling with gaussian distributions for classifications,” in *Proc. ICASSP*, 1998, pp. 661–664.
- [8] M.J.F. Gales, “Maximum likelihood linear transformations for HMM-based speech recognition,” Tech. Rep. CUED/FINFENG/TR291, Cambridge Univ., 1997.
- [9] D. Povey and P.C. Woodland, “Improved discriminative training techniques for large vocabulary continuous speech recognition,” in *Proc. ICASSP*, 2001.
- [10] Mehryar Mohri and Michael Riley, “Integrated context-dependent networks in very large vocabulary speech recognition,” in *Proc. Eurospeech*, 1999.
- [11] Mehryar Mohri, Fernando Pereira, and Michael Riley, “Weighted finite-state transducers in speech recognition,” in *Proc. ISCA ITRW ASR 2000*, 2000.
- [12] Mehryar Mohri and Michael Riley, “A weight pushing algorithm for large vocabulary speech recognition,” in *Proc. Eurospeech*, 2001.
- [13] J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarlin, “A set of level 3 basic linear algebra subprograms,” *ACM Trans. Math Soft.*, vol. 16, no. 3, pp. 1–17, 1990.
- [14] Greg Henry, “Linux BLAS for Intel architecture,” <http://www.cs.utk.edu/~ghenry/distrib/archive.htm#blas>.
- [15] J. Glass, T.J. Hazen, and I.L. Hetherington, “Realtime telephone-based speech recognition in the Jupiter domain,” in *Proc. ICASSP*, 1999, pp. 61–64.
- [16] M. Siegler, U. Jain, B. Raj, and R.M. Stern, “Automatic segmentation and clustering of broadcast news audio,” in *Proc. DARPA Speech Recognition Workshop*, 1997, pp. 97–99, Code available at [http://www.nist.gov/speech/tools/CMUseg\\_05targz.htm](http://www.nist.gov/speech/tools/CMUseg_05targz.htm).