An Architecture for Rapid Decoding of Large Vocabulary Conversational Speech

George Saon, Geoffrey Zweig, Brian Kingsbury, Lidia Mangu and Upendra Chaudhari

IBM T. J. Watson Research Center, Yorktown Heights, NY, 10598
e-mail: {gsaon,gzweig,bedk,mangu,uvc}@us.ibm.com

Abstract

This paper addresses the question of how to design a large vocabulary recognition system so that it can simultaneously handle a sophisticated language model, perform state-of-the-art speaker adaptation, and run in one times real time[1] ($1\times$RT). The architecture we propose is based on classical HMM Viterbi decoding, but uses an extremely fast initial speaker-independent decoding to estimate VTL warp factors, feature-space and model-space MLLR transformations that are used in a final speaker-adapted decoding. We present results on past Switchboard evaluation data that indicate that this strategy compares favorably to published unlimited-time systems (running in several hundred times real-time). Coincidentally, this is the system that IBM fielded in the 2003 EARS Rich Transcription evaluation.

1. Introduction

Although real-time large vocabulary speech recognition has been a reality for a number of years, there is widespread recognition that the technology has not yet been totally perfected, and the ongoing popularity of DARPA sponsored Switchboard and Broadcast News transcription competitions indicates the importance of the problem.

The designer of a real-time decoding system is faced with a number of critical challenges. Among the many questions that must be answered are:

- Type of search space ? (dynamically created or statically compiled)
- Search strategy ? (stack-based or Viterbi-based)
- Lattice rescoring or multi-pass decoding ?
- Forms of speaker adaptation ?

Two important examples of significantly different real-time systems are the IBM stack decoder [7], and AT&T's WFST submission to last year's EARS competition (RT'02) [8]. The IBM system is based on a stack-search that dynamically expands the search space at runtime. In this strategy, a number of partial decoding paths exist on a stack. For a given path, a fast-match prefix-tree is used to identify possible word-extensions, and then the candidates that are returned are re-scored with a more detailed acoustic model. Because the past word history is available in the partial path, the system also exploits long-span acoustic context, and conditions the acoustic realization of phones on those in the preceding words. This results in more detailed acoustic models than typical triphone systems. A hierarchical Gaussian-evaluation strategy is used to reduce the time spent evaluating the acoustic model. Taken together, these design decisions lead to a system that is highly extensible (due to the dynamic nature of the search).

An alternative strategy based on Viterbi decoding with statically compiled decoding graphs has recently been pursued at AT&T [6, 8]. In this strategy, the acoustic-context model (decision-tree clustered triphones) is combined with the language model and lexicon at "compile time" to produce a static decoding graph that is the same for every utterance. The main advantage of this approach is that the graph can be heavily optimized (e.g. through determinization and minimization) in advance, so that minimal decoding work is required at "decode time". The AT&T strategy is distinctive in that it spends the bulk of its time doing the initial speaker-independent decoding, and produces an entire word-lattice to be used in subsequent steps. Speaker compensation consists of VTLN and MLLR, and the initial lattices are rescored with speaker-adapted models and a higher order n-gram language model.

The strategy that we explore in this paper combines the use of static decoding graphs with state-of-the-art speaker adaptation, and re-distributes the time available into an extremely fast speaker-independent decoding, followed by the estimation of VTLN [9], FMLLR [2], and MLLR [5] transformations, followed by a relatively time-consuming speaker-adapted decoding. The decoding graphs we use are also notable in that they use a full word of left-context in the context model - i.e. the realization of a phone is sensitive to all the other phones in the current word, and all the phones of the previous word that fall within a context window of ± 5 phones. The construction of such graphs involves solving an NP-hard optimization prob-

[1] According to NIST specifications, the total execution time including I/O has to be less than the duration of the speech signal.

lem, and in [10] we have described heuristics for doing this. To speed up the Gaussian computation, we have adopted a hierarchical evaluation strategy [1] and made use of the Streaming SIMD Extension 2 instruction set of the Pentium 4 processor.

2. System Overview

The operation of our system comprises the following steps depicted in Figure 1: (1) segmentation of the audio into speech and non-speech segments, (2) speaker independent decoding of the speech segments, (3) alignment-based vocal tract length normalization of the acoustic features, (4) alignment-based estimation of one maximum likelihood feature space transformation per conversation side, (5) alignment-based estimation of one MLLR transformation per speaker and (6) speaker-adapted decoding using MMIE-SAT trained acoustic models transformed by MLLR. Next, we describe each of the steps in more detail.

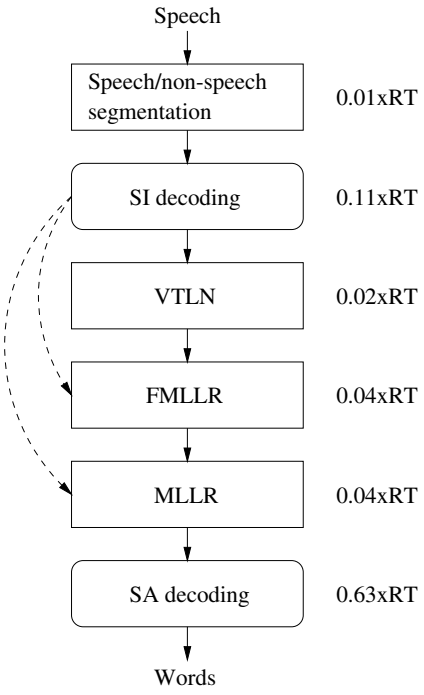

Figure 1: System diagram. Dashed lines indicate that the FMLLR and MLLR steps rely on the 1-best output of the speaker independent decoding. Runtimes are reported on a Linux Pentium 4 3.0GHz, 2.0GB machine and are exclusive of the feature computation and various data motion steps (which together account for $0.12\times$RT).

2.1. Segmentation

There are two main reasons for segmenting the audio prior to decoding. First, segmenting the data and eliminating the non-speech segments reduces the computational load during recognition. For instance, if we consider the RT'02 test set, the speech amounts to 6 hours of two-channel conversations. Processing the channels independently without segmentation would result in a 12-hour signal length whereas eliminating the silence regions on one channel (when the other channel is active) halves the amount of data to be considered for further processing. Second, all the acoustic models are trained on segmented data. It is therefore desirable to segment the test data for consistency reasons. For example, the amount of silence varies a lot between segmented and unsegmented speech and this can adversely affect the cepstral mean and variance normalization.

We use an HMM-based segmentation procedure very similar to the one described in [4]. Speech and non-speech segments are each modeled by five-state, left-to-right HMMs with no skip states. The output distributions in each HMM are tied across all states in the HMM, and are modeled with a mixture of diagonal-covariance Gaussian densities. The segmentation is performed using a log-space Viterbi decoding algorithm that can operate on very long conversation sides. A segment-insertion penalty is used during decoding to control the number and duration of the hypothesized speech segments. Following the decoding, the hypothesized segments are extended by an additional 30 frames to capture any low-energy segments at the boundaries of the speech segments and to provide sufficient acoustic context for the speech recognizer. The feature vectors used are the same as for the speaker independent decoding and are described in the next subsection.

2.2. Front-end

Speech is coded into 25 ms frames, with a frame-shift of 10 ms. Each frame is represented by a feature vector of 24 Mel frequency-warped cepstral coefficients for the speaker independent decoding and by 13 VTL-warped perceptual linear prediction cepstral coefficients for the speaker adapted decoding. For both feature sets, we perform spectral flooring by adding the equivalent of one bit of additive noise to the power spectra prior to Mel binning, and use periodogram averaging to smooth the power spectra. Every 9 consecutive cepstral frames are spliced together and projected down to 60 dimensions using LDA. The range of this transformation is further diagonalized by means of a maximum likelihood linear transform. Prior to splicing and projection, the cepstra are mean- and variance-normalized on a per-side basis, with the exception of c_0, which is normalized on a per-utterance basis.

2.3. Acoustic models

The recognition system uses a phonetic representation of the words in the vocabulary. Each phone is modeled

Number of	SI	SA
leaves	4.0K	4.6K
Gaussians	168K	158K

Table 1: Number of context-dependent acoustic units and Gaussians for the speaker independent and speaker adapted models.

with a 3-state left-to-right HMM. Further, we identify the variants of each state that are acoustically dissimilar by asking questions about the phonetic context (within an 11-phone window) in which the state occurs. The questions are arranged hierarchically in the form of a decision tree, and its leaves correspond to the basic acoustic units that we model. The output distributions for the leaves are given by a mixture of at most 128 diagonal covariance Gaussian components. The exact number of Gaussians for each leaf was determined using the Bayesian Information Criterion. Table 1 summarizes the number of leaves and the number of 60-dimensional Gaussians for the speaker-independent and SAT models. The SAT models where trained through MMIE on the following sources of data: 247 hours of Switchboard, 18 hours of Callhome and 18 hours of Switchboard cellular.

2.4. Speaker compensation

An initial speaker-independent (SI) decoding of the data produces hypotheses and forced alignments that are used to estimate frequency warping factors for VTLN decoding [9]. Our particular VTLN implementation uses 21 different warp scales allowing for a $\pm 20\%$ stretching of the frequency axis. Jacobian compensation is performed by adding the log determinant of the sum of outer-products of the warped cepstra to the average frame log-likelihood. The Viterbi alignments from the SI decoding are used again to estimate one affine feature-space maximum likelihood linear regression (FMLLR) transform [2] for each conversation side. This FMLLR transform maps the VTL-warped test data to a canonical speaker-adaptively trained (SAT) feature space. The SI alignments are re-used a third time to estimate one MLLR transform [5] per speaker. In order to accelerate the computations, all three compensation algorithms detect speaker changes on the fly, reinitialize the sufficient statistics automatically and minimize the amount of I/O by reading all static information at the beginning of the execution and by writing a minimum amount of data per speaker (VTL warp scale, FMLLR and MLLR matrices). Last but not least, all matrix and vector operations are written using routines from the Intel Math Kernel Library (an efficient implementation of the BLAS for Pentium).

Number of	SI	SA
ngrams	0.2M	3.3M
states	0.6M	9.6M
arcs	1.7M	23.9M

Table 2: Number of n-grams, states and arcs for the speaker independent and speaker adapted decoding graphs.

2.5. Viterbi decoding on static FSM graphs

Both the speaker independent and speaker adapted decodings operate on static FSM decoding graphs [6]. These FSMs are obtained by successively expanding the words in an n-gram language model in terms of their pronunciation variants, the phonetic sequences of these variants and the context dependent acoustic realizations of the phones. We have shown recently in [10] that it is possible to handle a whole word of cross-word acoustic context to the left by efficiently factorizing the sets of arcs between word pronunciation variants and left context dependent pronunciation variants using arc minimization techniques. In conjunction with these arc minimization techniques, we use standard determinization and minimization algorithms to further reduce the size of the resulting graphs. In Table 2 we show a comparison between the graph sizes of the speaker-independent and speaker-adapted decodings. The main difference lies in the choice of the language model: for the SI decoding we opted for a bigram LM whereas the final decoding step uses a 4-gram LM. The latter was trained on the following corpora: 3M words of Switchboard, 58M words from web scripts publicly available from the University of Washington, 3M Broadcast news words relevant to Switchboard topics and 7M words of the English Gigaword corpus.

By analyzing the runtimes of the two decoding steps in Figure 1, one can see that the SI decoding is 6 times faster. This is more a consequence of the tightness of the pruning parameters than of the size of the decoding graphs. The main pruning threshold is the maximum number of active states (or hypotheses) per frame. This parameter was set to 500 for the speaker independent-decoding and to 3500 for the speaker-adapted decoding. In Figure 2, we illustrate the influence of the number of active states on the word error rate and run-time factor (RTF) for the speaker-adapted decoding on the RT'02 test data.

The second pruning threshold has to do with the maximum number of evaluated Gaussians per frame. First, we perform a top-down clustering of all the mixture components in the system using a Gaussian likelihood metric until we reach 2048 clusters (Gaussians). At runtime, we evaluate all the 2048 components for every frame and then only evaluate those Gaussians which map to one of the top N clusters for a particular frame [1]. N is set to

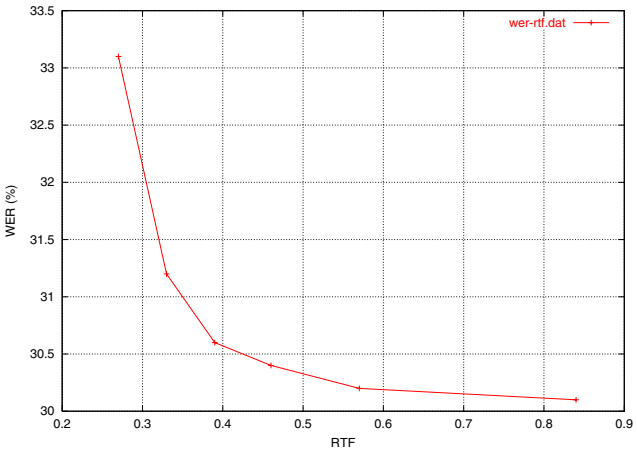

Figure 2: Word error rate versus real-time factor for the speaker-adapted decoding on the RT'02 test data.

20 for the speaker independent decoding and to 110 for the speaker adapted decoding.

In addition to the previously mentioned hierarchical likelihood scheme, the following techniques were found beneficial for accelerating the Gaussian evaluation part of the decoder. First, we made use of the Streaming SIMD[2] Extension 2 of the Pentium 4 processor through the C++ wrapper classes provided by the Intel compiler. This is similar in spirit to [3] although here we used floating point SIMD instructions directly as opposed to scaling and integerizing the code. These instructions operate on four floating point numbers in parallel and their use resulted in a 30% speed-up over a straightforward loop-unrolled implementation of the likelihood calculation. The second major speed improvement came from sorting the Gaussians according to the top-level cluster indices. Then, the way the likelihood is computed can be written as follows:

```
for cluster from 1 to 2048
  for frame from 1 to T
    if isactive[cluster][frame] then
      for gaussian from first[cluster] to last[cluster]
        likelihood(gaussian, frame)
```

The benefit from sorting the Gaussians based on the top-level cluster index is now apparent. The Gaussians accessed in the innermost loop are stored contiguously in memory thus minimizing the number of cache misses. Sorting the Gaussians and using this algorithm resulted in an additional 40% speed-up in the likelihood evaluation.

2.6. System performance

The experiments were conducted on the RT'02 test data which consists of 60 conversations (120 speakers) totaling 6 hours of speech. After segmentation, the average amount of data per speaker is roughly 3 minutes. Table 3 indicates the word error rates on the reference (manual) segmentation after the various processing steps (in the $1\times$RT system, only the first and the last decoding step are actually performed). When tested on the automatic segmentation, the speaker-adapted word error rate increased to 31.7%.

[2] Single Instruction Multiple Data.

System	SI	VTLN	FMLLR	MLLR
WER	50.3%	34.1%	30.6%	30.1%

Table 3: Word error rates on the reference segmentation.

3. References

[1] E. Bocchieri. Vector quantization for efficient computation of continuous density likelihoods. In Proc. ICASSP'93, Minneapolis, 1993.

[2] M. J. F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. Technical Report CUED/F-INFENG, 1997.

[3] S. Kanthak, K. Schuetz, H. Ney. Using SIMD Instructions For Fast Likelihood Calculation in LVCSR. In Proc. ICASSP'00, Istanbul, 2000.

[4] B. Kingsbury, G. Saon, L. Mangu, M. Padmanabhan and R. Sarikaya. Robust speech recognition in noisy environments: the IBM 2001 SPINE evaluation system. In Proc. ICASSP'02, Orlando, 2002.

[5] C. J. Leggetter and P. C. Woodland. Speaker adaptation of HMMs using linear regression. Technical Report CUED/F-INFENG, 1994.

[6] M. Mohri, F. Perreira and M. Riley. Weighted finite state transducers in speech recognition. In ISCA ITRW ASR'00, Paris, 2000.

[7] M. Novak and M. Picheny. Speed improvement of the tree-based time asynchronous search. In Proc. ICSLP'00, Beijing, 2000.

[8] M. Riley, E. Bocchieri, A. Ljolje and M. Saraclar. The AT&T 1x real-time Switchboard speech-to-text system. In Proc. NIST RT'02 Workshop, 2002.

[9] S. Wegman, D. McAllaster, J. Orloff, and B. Peskin, "Speaker normalization on conversational telephone speech", In Proc. ICASSP'96, 1996.

[10] G. Zweig, G. Saon and F. Yvon. Arc Minimization in Finite State Decoding Graphs with Cross-Word Acoustic Context. In Proc. ICSLP'02, Denver, 2002.