Experiments with Tree-Structured MMI Encoders on the RM Task

Mark T. Anikst, William S. Meisel, Matthew C. Soares
Speech Systems Incorporated
18356 Oxnard Street
Tarzana, California 91356

Kai-Fu Lee
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

This paper describes the tree-structured maximum mutual information (MMI) encoders used in SSI's Phonetic Engine® to perform large-vocabulary, continuous speech recognition. The MMI encoders are arranged into a two-stage cascade. At each stage, the encoder is trained to maximize the mutual information between a set of phonetic targets and corresponding codes. After each stage, the codes are compressed into segments. This step expands acoustic-phonetic context and reduces subsequent computation. We evaluated these MMI encoders by comparing them against a standard minimum distortion (MD) vector quantizer (encoder). Both encoders produced code streams, which were used to train speaker-independent discrete hidden Markov models in a simplified version of the Sphinx system [3]. We used data from the DARPA Resource Management (RM) task. The two-stage cascade of MMI encoders significantly outperforms the standard MD encoder in both speed and accuracy.

INTRODUCTION

Most hidden Markov model systems use minimum distortion (MD) vector quantizers (encoders) to convert continuously valued speech parameters into streams of integer codes. However, MD encoders do not optimize a criterion that is directly related to recognition accuracy. Moreover, they use a single distortion measure that may not be appropriate for all speech classes. In this paper, we propose the use of maximum mutual information (MMI) encoders that are trained to extract phonetic information and thereby minimize phonetic recognition errors. We further compress the frames into larger segments and repeat the encoding.

Our MMI encoders are binary decision trees built to maximize the average mutual information between the phonetic targets and the codes assigned to them. The task of training such encoders has been extensively addressed in the theory of binary decision trees [5, 8, 2]. For example, Breiman *et al.* systematically consider binary decision trees applied to various classification tasks. The decision (interior) nodes of the tree are allowed to use linear combinations of feature vectors, as well as unordered categorical features. Training criteria ("impurity" criteria) for the binary decision trees include the average leaf-node-conditional class entropy. Training is performed in a top-down node-at-a-time fashion, adding new leaf nodes and maximizing reduction in the average leaf node impurity attained by such additions. It is demonstrated on many practical classification problems that the above procedure results in a suboptimal, but sufficiently accurate tree.

Labelled data necessary for the supervised training is obtained by aligning speech frames with phonetic transcriptions using dynamic programming. We train a two-stage cascade of binary-tree encoders. In the first stage, *frames* are encoded to extract maximum information about their target label classes. Feature vectors used in the tree encoder are frame-based. Contiguous runs of frames with the same code are compressed into segments. In the second stage, the resulting *segments* are encoded to extract maximum information about their target label classes (we assign a single target label class per segment). Segment-based acoustic feature vectors are used in the second-stage tree encoder, along with some categorical features based on the phonetic identities uncovered by the first-stage tree encoder. Segment duration features are also used. Resulting runs of segments with the same code are again compressed into larger segments.

Speech Systems Incorporated (SSI) has been using a version of this two-stage cascade of the MMI encoders in the Phonetic Engine®, an integral part of SSI's large-vocabulary, continuous speech recognition system [6, 1]. The two-stage trees are very fast; they encode one second of speech in one-third of a second on a 16 mHz 68020 microprocessor. In this study, we apply these MMI encoders in a more limited sense -- as vector quantizers for the Sphinx speech recognition system [3]. This enables a direct comparison of MMI encoders and standard MD encoders. In our experiments, for the sake of expediency, we used a simplified version of the Sphinx system limited to 48 context-independent phonetic HMMs and 26 acoustic frame features. The two-stage cascade of MMI encoders outperforms the standard MD encoder: Word error rate drops by 33% and recognition is performed roughly 1.6 times faster.

We also ran a preliminary evaluation of the MMI and MD encoders using the Sphinx 1100 context-dependent (generalized triphone) HMMs. We used the same codes without re-growing the trees for context-dependent class targets. Error rate was reduced by more than half relative to no use of context.

SYSTEM OVERVIEW

Here we briefly describe the system used in our experiments. Figure 1 summarizes the encoding process and the experiments performed.

Acoustic Processing

The speech is sampled at 16 kHz and is converted into a sequence of 10-msec frames of 26 acoustic parameters: 12 cepstrum coefficients, 12 differenced cepstrum coefficients, power and differenced power [3].

Labelling

Training of the tree-structured MMI encoders is performed using labelled speech data. The set of label classes used for labelling contains 144 classes: there is a unique label class for each of the three pdf's (roughly corresponding to beginning, middle, and end) of each of the 48 Sphinx context-independent phones. Labelled frame data for training is obtained via Viterbi alignment using the Sphinx system.

First-Stage (Frame) MMI Encoder

At the first (frame-coding) stage, frames are encoded in such a way as to convey maximum information about their underlying label class identities. To perform frame encoding, the frame time-sequence is scanned by a "sliding window" covering W frames; in our experiments, we kept W = 1 (a constraint imposed for the sake of a fair comparison between the first-stage MMI encoder and the standard MD encoder; normally, we use a three-frame window). A set of the 26 acoustic parameters of a frame was used as a feature vector accessed by the window. The tree frame encoder takes as input this feature vector and outputs a code for the frame at the center of the window. The encoder is trained to maximize the average mutual information between its code alphabet and the alphabet comprised of the 144 target label classes.

The resulting sequence of coded acoustic frames is further processed to form acoustic segments by merging time-contiguous blocks of frames with the same code. Also, the most likely broad phonetic class is assigned to each formed segment. The stream of the acoustic segments with the assigned segmentation classes constitutes the input to the segment-coding stage.

Second-Stage (Segment) MMI Encoder

The second (segment-coding) stage processing is similar to that of the frame-coding stage. Namely, segments are encoded in such a way as to convey maximum information about their underlying phonetic classes.

To perform segment encoding, the stream of segments is scanned by a sliding time window covering three segments (W = 3). A set of pre-defined feature vectors is extracted from the acoustic parameters of all the frames encountered in the segments accessed by the window. Also, the most-likely broad phonetic classes assigned after the first stage to each of the three segments in the window comprise additional categorical variables. These variables provide phonetic features complementing the acoustic features. Segment duration features are also computed. The segment encoder tree takes as input these sets of features and outputs a code for the segment in the center of the window. The encoder is trained to maximize the average mutual information between its code alphabet and the alphabet comprised of the 144 target label classes. The target labels for segments were derived from the labels of the constituent frames.

To obtain a categorical feature for use in the tree based on the phonetic class of a segment, we combined 144 target phonetic classes into nine broad superclasses, and used the most likely superclass number for each code. The selected set of broad phonetic superclasses is shown in Table 1 (in the standard notation of the Sphinx phonetic system, [3]).

Class	Phones
0	SIL
1	S SH Z ZH TS JH CH
2	W L
3	V F TH
4	T K P H TD PD KD G B D DH DX DD
5	R ER
6	N M NG
7	AH AE AA AY AO OW OY AW
8	EH EY IH IY AX IX Y UH UW

Table 1: Superclasses used in a categorical variable for the second-stage tree.

The resulting sequence of coded segments is further processed to form larger segments as in the first stage. The stream of the enlarged segments with the assigned codes constitutes the output of the second stage.

MMI Training

The first- and second-stage MMI encoders are trained using labelled data (supervised training). The encoders are trained as binary decision trees using maximization of the average mutual information I(*classes, codes*) between the set of target label *classes* and the set of leaf-node numbers (*codes*), as the training criterion:

$$I(\textit{classes}, \textit{codes}) = \sum_{\textit{class}} \sum_{\textit{code}} \Pr(\textit{class},\textit{code}) * \log(\Pr(\textit{class},\textit{code})/(\Pr(\textit{class}) * \Pr(\textit{code}))),$$

where Pr(*class,code*) is the joint probability of the *class* and the *code* assigned to a training sample, Pr(*class*) and Pr(*code*) are the marginal *class* and *code* probabilities, respectively.

Training is performed top-down, starting from the root of the binary decision tree. The decision function associated with each decision node of the tree effects the split of the feature space with a hyperplane (for the continuous-valued feature vectors) or a dichotomy of a discrete set (for a categorical feature). The training samples at each node were those which reached that node after passing through predecessor nodes.

Training of the decision function at a node uses as an optimization criterion the reduction in the node's average class entropy. To find a decision hyperplane, we use conjugate gradient based search [9] where the gradient of the criterion function with respect to the hyperplane coefficients is computed by replacing the "hard limiter" decision function with a piecewise linear one (the threshold-logic type) and gradually annealing that non-linearity to the hard limiter.

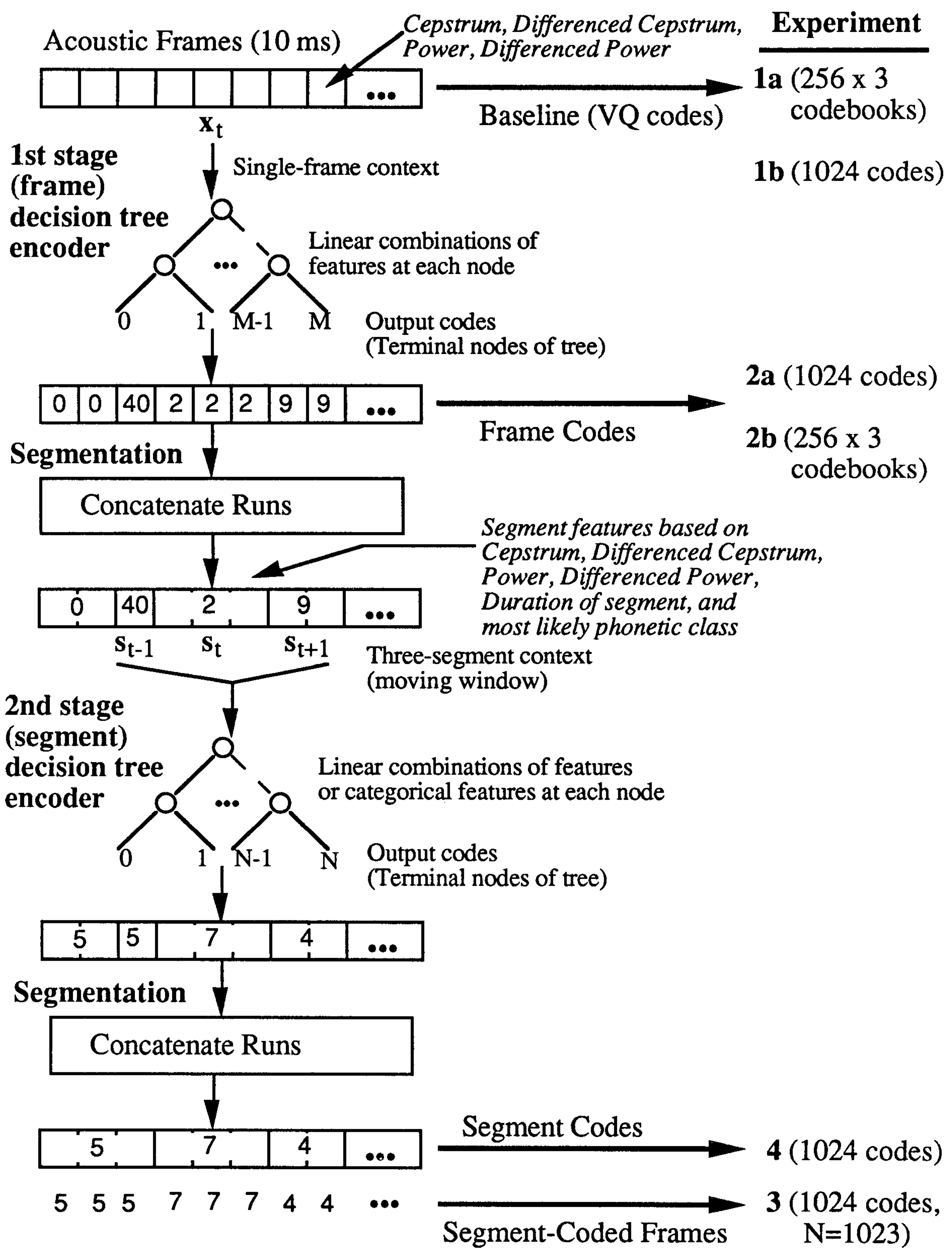
Acoustic Frames (10 ms)
Cepstrum, Differenced Cepstrum, Power, Differenced Power
Experiment
1a (256 x 3 codebooks)
Baseline (VQ codes)
x_t
1st stage (frame) decision tree encoder
Single-frame context
1b (1024 codes)
Linear combinations of features at each node
0 1 M-1 M
Output codes (Terminal nodes of tree)
2a (1024 codes)
0 0 40 2 2 2 9 9 •••
Frame Codes
2b (256 x 3 codebooks)
Segmentation
Concatenate Runs
Segment features based on Cepstrum, Differenced Cepstrum, Power, Differenced Power, Duration of segment, and most likely phonetic class
0 40 2 9 •••
s_{t-1} s_t s_{t+1}
Three-segment context (moving window)
2nd stage (segment) decision tree encoder
Linear combinations of features or categorical features at each node
0 1 N-1 N
Output codes (Terminal nodes of tree)
5 5 7 4 •••
Segmentation
Concatenate Runs
5 7 4 •••
Segment Codes
4 (1024 codes)
5 5 5 7 7 7 4 4 •••
3 (1024 codes, N=1023)
Segment-Coded Frames

Figure 1: Overview of the Multi-Stage Decision Tree (MMI) Encoder and Experiments.

Once the optimal coefficients are estimated, we use the hard limiter decision function to send the patterns to the left or the right child node. We don't split a node if the highest reduction in the class entropy attained by the "node split" is less than a certain fraction of the node's class entropy; a final node is a *terminal* node.

After the entire binary tree is created, its performance criterion (i.e. the average mutual information between the set of target classes and the set of the terminal nodes) is evaluated with a combination of the training and independent sets of labelled data. Some nodes are then removed, starting with the current terminal nodes, i.e., the tree is "pruned," to produce a more robust subtree with more accurate estimates of the node-class probabilities. The resulting terminal node numbers are used as codes.

The above training and pruning of the trees was performed utilizing SSI tree-growing software.

EXPERIMENTS

We compared various MMI tree encoders with the standard MD encoders (quantizers), as used in Sphinx and other discrete HMM-based systems. Both MMI and MD encoders produce codes, which were used as input to the Sphinx System [3]. For this study, a simplified version of the Sphinx system was used. Instead of context-dependent modeling, we used only context-independent models. Instead of 51 features (as used in the latest version), we used only 26 features (12 cepstrum coefficients, 12 differenced cepstrum coefficients, power and differenced power). Therefore, the results should be evaluated relatively rather than absolutely. We evaluated both a three-codebook version (256 codes per codebook) and a one-codebook version (1024 codes). For the one-codebook version, we also used co-occurrence smoothing and deleted interpolation [4] to smooth rarely observed codes. We used the standard inventory of 48 phonetic models, each with 7 states and 3 output pdf's.

We also started a preliminary evaluation of the second-stage segment MMI codes for a version of the Sphinx system using context-dependent HMMs. Results are given at the end of this section.

The task for our study is the DARPA Resource Management (RM) task, with the perplexity 60 word-pair grammar [7]. We used the standard "extended training set" of 3990 sentences from 109 speakers for speaker-independent training. We trained the phonetic HMMs on all 3990 sentences. All results were evaluated on 300 independent test sentences from 12 speakers (the June 88 test set). Following that, selected cases were evaluated on the RM2 June 90 test set as a verification.

We first generated a first-stage MMI tree encoder (MMI-1024). This tree was grown using 144 target phonetic classes (48 phones x 3 distributions). All 26 features were accessible at all nodes to form linear decision boundaries (via linear combination splits). We used half of the training sentences to grow the MMI tree encoder, and all of the training sentences to prune it. This tree was grown to 1430 codes, and then pruned to 1024 codes. The average code-class mutual information and corresponding error rate (*substitution + deletion + insertion*) on the RM task (after the Forward-Backward training with co-occurrence smoothing and deleted interpolation) are shown in in Table 2.

To evaluate this result, we also generated an MD encoder (quantizer) that used the same 26 features, utilizing a weighted Euclidean distance (MD-1024) [3]. The results of this encoder (again, after the Forward-Backward training with co-occurrence smoothing and deleted interpolation) are shown in Table 2. In this experiment, the MMI-1024 encoder error rate was 3.5% lower than the MD-1024 encoder (a 15% reduction in error rate).

Experiment No. [1]	Encoder	Info (bits)	**Error Rate (%)**
2a	MMI-1024	3.42 out of 6.63	19.2
1b	MD-1024	3.16 out of 6.63	22.7

1 See Fig. 1.

Table 2: Comparison of an MD encoder with an MMI frame stage encoder: a single codebook.

Since the standard Sphinx system uses three separate VQ codebooks, we also compared the performance of a 3-codebook MD encoder and a 3-codebook MMI encoder. In each case, the encoder has access only to a subset of the features (VQ1 - 12 cepstrum coefficients, VQ2 - 12 differenced cepstrum coefficients, and VQ3 - power & differenced power). The codebook size was the same for all the encoders (256 codes). Co-occurrence smoothing of the output code pdf's was not performed in these experiments, but deleted interpolation was done. The results (see Table 3) indicate that the MMI encoder gives slightly higher error than the MD encoder (despite higher information extracted), and both were worse than the MMI-1024 encoder. We conclude that effective tree encoders require access to the entire feature vector, so as to exploit the between-feature relationships.

Experiment No. [1]	Encoder	**Info (bits)** (VQ1, VQ2, VQ3)	**Error Rate (%)**
2b	MMI 3-VQ	2.23, 1.77, 1.79 out of 6.63	20.5
1a	MD 3-VQ	2.09, 1.51, 1.68 out of 6.63	20.0

1 See Fig. 1.

Table 3: Comparison of MD encoders with MMI frame-stage encoders: three codebooks.

Next, we evaluated the second-stage MMI tree encoder. We used a three-segment sliding window to compute features derived from the 26 frame acoustic parameters, and categorical features derived from the segment phonetic identities discovered by the first-stage tree encoder. Segment duration features were also computed.

The target labels for segments were derived from the labels of the constituent frames. Using those targets, we grew a second-stage MMI tree encoder to 1417 codes (using all of the training sentences) and then pruned it to 1024 codes. The codes output by the encoder were further compressed by combining runs of segments with the same codes into larger segments.

We evaluated the second-stage codes in two ways: as frame codes (every constituent frame of a segment was assigned the segment code, MMI-SF), and as segment codes (one code per segment, MMI-SS). Respectively, we trained two sets of the phonetic HMMs (standard 48 phonetic models of the SPHINX system) and ran recognition tests using streams of frame and segment codes. The code-class mutual information and corresponding error rates are shown in Table 4 (after the Forward-Backward training with co-occurrence smoothing and deleted interpolation).

Although MMI-SF extracts substantially more information, the performance was slightly lower. However, switching to segment codes (MMI-SS) resulted in a performance improvement of 4.0% (21% reduction in error) relative to the first stage alone. Performance was improved 7.5% (33% reduction) over the MD-1024 baseline (Table 2).

Experiment No.	Encoder	Info (bits)	Error Rate (%)
2a	MMI-1024	3.42 out of 6.63	19.2
4	MMI-SF	3.85 out of 6.63	19.8
3	MMI-SS	3.52 out of 6.73	15.2

Table 4: MMI encoders: different temporal units. (MMI-SF is segment codes on frames; MMI-SS is segment codes on segments.)

It was also found that MMI segment codes lead to significant frame compression (on the average, 1.6 frames/segment) and therefore to significant speed advantages (which should be roughly proportional to the reduction in segments). Table 5 illustrates this phenomenon. Thus, there was a simultaneous improvement in speed and accuracy using an MMI segment encoder rather than an MD vector quantizer.

Table 5 displays the average number of temporal units (frames or segments) per target label class in the Per Target column. The number of segments decreases with each stage of successive temporal compression. In the final segmentation, the number of temporal units per target label class is reduced by a factor of 1.6. We can measure whether the temporal compression loses target class segment boundaries, by examining the percentages of the target label classes which were merged into groups of two or more within single segments (Merged Targets column); only 3.2% of such targets were merged by the final segmentation stage.

Segmentation	Per Target	Merged Targets
before 1st stage	3.18 frames	0%
after 1st stage	2.48 segments	1.6%
after 2nd stage	2.01 segments	3.2%

Table 5: Effect of segmentation.

We conjecture that the observed improvement in the recognition accuracy for the segment codes versus frame codes is mainly due to the following. First, the underlying assumption of independence of the output code distributions given a transition in a phonetic class model (made for use of the hidden Markov models of phonetic classes) is satisfied to a greater extent when the runs of frames with the same code are merged in a single segment code, thus absorbing short-time dependencies. Therefore, the HMMs become more adequate models of the phonetic classes. Second, there remains a sufficient amount of training data for the segment codes after the data is compressed due to segmentation. Finally, segmentation does not lead to any significant merging of the target label classes within the resulting segments, thereby retaining temporal resolution of phonetic targets.

We also made a preliminary evaluation of the 2nd-stage segment MMI codes for a version of the Sphinx system using context-dependent HMMs (1100 generalized triphone models for within- and between-word triphones). The results are shown in Table 6. Results for a comparable Sphinx configuration using 3 MD codebooks (using subsets of the 26 features) is shown for comparison. In both cases, co-occurrence smoothing was performed along with deleted interpolation. Although the word accuracy is close for both cases, the decoding speedup for the segment codes gives the advantage to the MMI encoder. We view these results as rather encouraging, in view of the following limitations: (a) the encoder tree's topology was not utilized for pdf smoothing, and (b) training of the MMI encoders was done on the pdf labels of the 48 phones, and not on the generalized triphones. Further investigation of the use of MMI encoders with context-dependent HMMs will be conducted in the future.

Encoder	Error Rate (%)
MMI-SS-Context Dependent	7.1
MD 3-VQ-Context Dependent	7.0

Table 6: Context-dependent decoding.

Results for the RM2 (June 90) test set are shown in table 7. They show the same trend.

Encoder	Error Rate (%)
MD-1024	19.5
MMI-SS	15.6
MMI-SS-Context Dependent	8.0

Table 7: Results on June 90 RM test set.

CONCLUSION

We compared vector quantization (the MD encoder) with no segmentation to a multi-stage decision-tree encoder (the MMI encoder) with and without segmentation. We found that the MMI encoder (1) extracts a significantly larger amount of information than the MD encoder; (2) works better with a combined feature set (as a single tree); and (3) yields higher accuracy with faster decoding time when segment codes are used.

In order to make a controlled comparison, neither the best decision tree technology nor the best Markov model technology was used. In decision trees, we did not use wider context in the frame tree, as in previous work [1]. In addition, we have found that a third segmentation stage helps, creating even larger yet "clean" segments (unpublished work at SSI). The decision tree can easily use more features simultaneously, providing the prospect of more informative codes. Since the trees make dichotomous decisions, more extensive smoothing of the codes (utilizing tree topology) should help. Further, several iterations of the entire process of labelling the frames and tree-growing can be repeated to improve accuracy (as long as the resulting recognizer provides more accurate decoding than that of the previous iteration). Finally, due to temporal compression of frames and resulting data reduction, a reduced topology of the phonetic HMMs (e.g., fewer states/transitions) may yield a better fit to the segment codes. Future research will include trying some of these variations.

In our experiments, we have not fully explored the context dependency of the phonetic models. Further investigation of the use of MMI encoders with context-dependent HMMs will be conducted in the future.

REFERENCES

1. Anikst, M.T., Meisel, W.S., Newstadt, R.E., Pirzadeh, S.S., Schumacher, J.E., Shinn, P., Soares, M.C., Trawick, D.T. A Continuous Speech Recognizer Using Two-Stage Encoder Neural Nets. *Proc. International Joint Conference on Neural Networks*, Washington D.C., pp. II-306 - II-309, January 1990.

2. Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. *Classification and Regression Trees*, Wadsworth International Group, Belmont, Calif., 1984.

3. Lee, K.-F. *Automatic Speech Recognition: The Development of the SPHINX System.* Kluwer Academic Publishers, Boston, 1989.

4. Lee, K.F., Hon, H.W. Speaker-Independent Phone Recognition Using Hidden Markov Models, *IEEE Transactions on ASSP*, November, 1989.

5. Meisel, W.S., Michalopoulos, D.A. A Partitioning Algorithm with Application in Pattern Classification, Piecewise-Constant Approximation, and the Optimization of Decision Trees, *IEEE Trans. on Computers*, January 1973.

6. Meisel, W.S., Fortunato, M.P., Michalek, W.D. A Phonetically-Based Speech Recognition System, *Speech Technology*, pp. 44-48, Apr/May 1989.

7. Pallett, D. S. Benchmark Tests for DARPA Performance Evaluations, *Proc. ICASSP 98*, pp. 536-539, May 1989.

8. Payne, H.J., Meisel, W.S. An Algorithm for Constructing Optimal Binary Decision Trees, *IEEE Trans. on Computers*, September 1977.

9. Press, W.H., Flannery, B.P.,Teukolsky, S.A., Vetterling, W.T. *Numerical Recipes*, Cambridge University Press, 1986.