Memory Efficient Modeling of Polyphone Context with Weighted Finite-State Transducers

Emilian Stoimenov

Institut für Theoretische Informatik
Universität Karlsruhe
76131 Karlsruhe, Germany
emilian.stoimenov@gmail.com

John McDonough

Spoken Language Systems
Saarland University
66123 Saarbrücken, Germany
john.mcdonough@lsv.uni-saarland.de

Abstract

In earlier work, we derived a transducer HC that translates from sequences of Gaussian mixture models directly to phone sequences. The HC transducer was statically expanded then determinized and minimized. In this work, we present a refinement of the previous algorithm whereby the initial HC transducer is *incrementally* expanded and immediately determinized. This technique avoids the need for a full expansion of the initial HC, and thereby reduces the random access memory required to produce the determinized HC by a factor of more than five. With the incremental algorithm, we were able to construct HC for a semi-continuous acoustic model with 16,000 distributions which reduced the word error rate from 34.1% to 32.9% with respect to a fully-continuous system with 4,000 distributions on the lecture meeting portion of the NIST RT05 data.

1. Introduction

State-of-the-art large vocabulary continuous speech recognition systems use subword units consisting of phones to model the words of a language. As coarticulation effects are prevalent in all speech, a phone must be modeled in its context to achieve optimal performance. The relevant contexts are most often chosen with a *decision tree* based on a measure of goodness such as the likelihood or entropy of a training set.

As originally proposed by Mohri *et al* [11, 9], a *weighted finite-state transducer* (WFST) that translates phone sequences into word sequences can be obtained by forming the *composition* $L \circ G$, where L is a *lexicon* which translates the phonetic transcription of a word to the word itself, and G is a *grammar* or *language model* (LM) which assigns to valid sequences of words a weight consisting of the negative log probability of this sequence. In the original formulation of Mohri and Riley [10], phonetic context is modeled by the series of compositions $H \circ C \circ L \circ G$, where H is a transducer converting sequences of Gaussian mixture models to sequences of polyphones, and C is a transducer that converts these polyphone sequences to corresponding sequences of phones. While this approach has proven effective, explicitly modeling the expansion of phones to polyphones with C introduces a great deal of redundancy, thereby causing the size of the final network $H \circ C \circ L \circ G$ to grow very large. Although this size and redundancy can be reduced through subsequent *determinization* and *minimization* [8], the mere fact that $H \circ C \circ L \circ G$ must be completely expanded before these optimizing operations can be applied effectively limited this technique to the representation of triphone context, inasmuch as $H \circ C \circ L \circ G$ could *not* be stored in random access memory for larger contexts.

Schuster and Hori [13] proposed a technique whereby all valid three-state sequences of Gaussian mixture models are enumerated, and thereafter the possible connections between these three-state sequences are determined; hence, the explicit expansion of C is avoided. Rather, Schuster and Hori derive a transducer HC that translates from sequences of Gaussian mixture models directly to phone sequences.

In previous work we demonstrated that Schuster and Hori's approach is *incorrect* for contexts larger than triphones. We also provided a correction for the error in Schuster and Hori's technique and discussed how the intermediate size of the network HC could be held in check. In this work, we present a refinement of our previous Stoimenov-McDonough algorithm whereby the initial HC transducer is *incrementally* expanded and immediately determinized. Thereby, this technique avoids the need for a full expansion of the initial HC, and thus allows for the compilation of much larger decision trees.

The balance of this work is organized as follows. In Section 2, we review our algorithm for statically constructing HC. Section 3 presents a refinement of the prior technique whereby the initial HC is expanded incrementally or on-the-fly. In Section 4, we present the results of our memory usage and timing studies comparing the static and incremental versions of the algorithm. We also present the results of a set of automatic speech recognition experiments demonstrating the improved performance obtained with the incremental construction algorithm.

2. Static Expansion of HC

Our ASR system distinguishes between regular phones and *word boundary phones*. The latter are distinguished from the former with the WB marker; e.g., {AH:WB} is an AH at a word boundary. Following Schuster and Hori [13], we begin by calculating a *bit matrix* $\mathbf{B}$ for each leaf node in a decision tree that specifies which phones are allowed in which positions. Each row of $\mathbf{B}$ corresponds to a phone and each column corresponds to a position in the polyphone context. Position (m, n) of $\mathbf{B}$ is one iff the m-th phone is allowed in the n-th position. The bit matrices are easily calculated by walking down the decision tree(s) from the root node to the leaves, and unsetting the bits corresponding to disallowed phones at each juncture.

We say two bit matrices $\mathbf{B}_i$ and $\mathbf{B}_j$ are *equivalent* if all bits in all locations have equal values, which we denote as $\mathbf{B}_i == \mathbf{B}_j$. We say $\mathbf{B}$ is *valid* if at least one bit is set in each column.

2.1. Metastate Enumeration

Let p denote the center phone for any given polyphone context. Let s_i denote the leaf node in a decision tree associated with the i-th state in a hidden Markov model (HMM). Assuming for

simplicity that all HMMs have three states, define a *metastate* $\mathbf{s}$ as a quintuple $\mathbf{s} = (p, s_1, s_2, s_3, \mathbf{B})$ where $\mathbf{B}$ is the valid bit matrix corresponding to the state sequence s_1, s_2, s_3. Let $\mathbf{B}' = \mathbf{B} \gg$ be the bit matrix obtained by right shifting $\mathbf{B}$ and let $\mathbf{B}'' = \mathbf{B}_n \ \& \ \mathbf{B}_m$ denote the *valid* bit matrix obtained by performing the bitwise & operation on $\mathbf{B}_n$ with $\mathbf{B}_m$. As in Schuster and Hori [13], we can enumerate a set $\mathbf{S}$ of valid metastates as follows. Begin with a bit matrix list $\mathbf{B}_{s_1}$ corresponding to the leaf node associated with the first state of a three-state sequence for a polyphone with center phone p. Similarly, let $\mathbf{B}_{s_2}$ and $\mathbf{B}_{s_3}$ be the bit matrix lists for the second and third states for such a three-state sequence for a polyphone with center phone p. If

$$\mathbf{B} = \mathbf{B}_{s_1} \ \& \ \mathbf{B}_{s_2} \ \& \ \mathbf{B}_{s_3}$$

is valid, then s_1, s_2, s_3 is a valid three-state sequence and the metastate $(p, s_1, s_2, s_3, \mathbf{B})$ can be added to $\mathbf{S}$.

2.2. Metastate Connection

Let $\mathbf{S} = \{\mathbf{s}_i\}$ denote the set of valid metastates obtained from the state sequence enumeration algorithm of Section 2.1 and let $\mathbf{T}$ be a second, initially empty, set of metastates. Let $\mathbf{Q}$ be a queue using any discipline and let SIL denote the initial silence metastate. The start and end nodes of HC are denoted as INITIAL and FINAL respectively. Additionally, let $\mathbf{E}$ denote the set of edges in HC. Denoting an input dictionary of names of GMMs and an output dictionary of phones as $\mathbf{D}$ and $\mathbf{P}$, respectively, we can express each edge $\mathbf{e} \in \mathbf{E}$ as a four-tuple,

$$\mathbf{e} = (s_{\text{from}}, s_{\text{to}}, d, p)$$

where s_{from} is the previous state, s_{to} is the following state, $d \in \mathbf{D}$ is the input symbol and $p \in \mathbf{P}$ is either an output symbol or epsilon.

Listing 1 Metastate connection.

```
00  def connectMetastates(SIL, S):
01    push SIL on Q
02    add SIL to T
03    connect INITIAL to SIL
04    while ||Q|| > 0:
05      pop q from Q
06      if q.p == SIL:
07        connect q to FINAL
08      foreach s ∈ S:
09        B ← (q.B ≫) & s.B
10        if valid(B):
11          t ← (s.p, s.s1, s.s2, s.s3, B)
12          if t ∉ T:
13            add t to T
14            push t on Q
15          e ← (q.s3, t.s1, t.s1.g, t.p)
16          add e to E
17    return (T, E)
```

Consider now the algorithm for metastate connection in Listing 1. In this listing, $\mathbf{Q}$ is a queue of metastates whose connections to other metastates have yet to be determined. In Line 05, the next metastate $\mathbf{q}$ is popped from $\mathbf{Q}$ and connected in Lines 06-07 to FINAL if $\mathbf{q}$ corresponds to SIL. In the loop that begins at Line 08, each $\mathbf{s} \in \mathbf{S}$ is tested to find if a new metastate $\mathbf{t}$ can be *derived* from $\mathbf{s}$, as in Line 11, to which $\mathbf{q}$ should be connected. This test consists of forming the new list $\mathbf{B}$ of valid bit matrices in Line 09, and checking if $\mathbf{B}$ is valid in Line 10. Note that the right shift $>>$ in Line 09 is to be understood as shifting in a column of *ones*. If $\mathbf{B}$ is valid, then the *name* of the new metastate $\mathbf{t}$ is formed in Line 11, and $\mathbf{T}$ is searched to determine if this $\mathbf{t}$ already exists. If $\mathbf{t}$ does *not* exist, then it is added to $\mathbf{T}$ and placed on the queue $\mathbf{Q}$ in Lines 12-14. The new edge $\mathbf{e}$ from the last state of $\mathbf{q}$ to the first state of $\mathbf{t}$ is created in Lines 15-16, where $\mathbf{t}.s_1.g$ is the name of the GMM associated with the latter.

Listing 2 Adjacency list expansion.

```
00  def edges(q):
01    if q.E ≠ ∅:
02      return q.E
03    if q.p == SIL:
04      connect q to FINAL
05    foreach s ∈ S:
06      B ← (q.B ≫) & s.B
07      if valid(B):
08        t ← (s.p, s.s1, s.s2, s.s3, B)
09        if t ∉ T:
10          add t to T
11        e ← (q.s3, t.s1, t.s1.g, t.p)
12        add e to q.E
13    return q.E
```

3. Dynamic Expansion of HC

It is clear from the network sizes given in Table 1 that HC is much smaller after determinization than when it is initially expanded, and smaller still after minimization. As explained by Mohri [8], weighted determinization can be performed incrementally inasmuch as it is *not* necessary to see the entire network in order to determine the adjacency list for a given node in the determinized network. Rather, only the nodes in the original network comprising the *subset* corresponding to a node in the determinized network, their adjacency lists, and residual weights are required.

The capability of performing incremental determinization introduces the possibility of incrementally expanding HC and simultaneously incrementally determinizing it. Such an incremental expansion of HC can be achieved by expanding the set $\mathbf{S}$ in the constructor of HC. The queue $\mathbf{Q}$ in Listing 1 would become unnecessary, as the adjacency lists of the nodes in HC would be expanded in the order required by the incremental determinization. To expand the adjacency list in the original HC, the steps in Lines 06-16 would be executed. A caching scheme can then be implemented whereby the connections between metastates in $\mathbf{T}$ that have not been accessed recently are periodically deleted and their memory recovered. Should the connections be needed in future, they can always be regenerated from the corresponding bit matrix list.

Consider the algorithm for expanding the adjacency list of a node in Listing 2. Lines 01-02 test if the adjacency list of $\mathbf{q}$ has already been expanded, and returns $\mathbf{q}.\mathbf{E}$ in the event that it has. Lines 03-12 are equivalent to Lines 06-16 of Listing 1. As mentioned above, the queue $\mathbf{Q}$ in Listing 1 is no longer necessary, as the edge lists are expanded in the order they are required by the determinization algorithm.

4. Experiments

Here we report the results of our experiments with the HC construction algorithms described in Sections 2 and 3.

Table 1: *Pentaphone network sizes.*

Network	States	Arcs
HC	975,838	63,178,405
$\det(HC)$	406,173	8,199,840
$\min(\det(HC))$	81,499	968,078
$HC \circ L \circ G$	12,497,566	42,492,816
$\det(HC \circ L \circ G)$	12,799,784	45,119,638
$\min(\det(HC \circ L \circ G))$	10,853,483	38,486,370

Table 2: *Memory usage and run-time requirements for static and dynamic construction of* $\det(HC)$.

Algorithm	Memory Usage (Gb)	Run Time (min)
static expansion	7.70	50
dynamic expansion	1.42	56

4.1. Static Expansion

Table 1 shows the sizes of a pentaphone recognition network after each stage in its construction. The HC transducer used here was first compiled statically from a pentaphone distribution tree containing 3,500 leaves. As explained in Section 4.2, we then compiled it dynamically to compare both implementations.

To construct the recognition network, we first built a bigram LM transducer G, trained on a combined corpus of conference proceedings, Broadcast News, and a variety of documents and meeting transcriptions collected as part of the EU integrated project CHIL, *Computers in the Human Interaction Loop*. The original LM contained 3,015,065 bigrams, which were pruned to 113,705 to construct a recognition network for the early-stage decoding passes.

From the first two lines in Table 1, it is clear that determinization reduces the number of arcs in HC by a factor of 7.7. This is the primary reason why the incremental construction algorithm described Section 3 is so effective at reducing the size of the task image required to construct HC.

4.2. Incremental Expansion Experiments

The experiments described in this section were conducted with the *Enigma* WFST library, which is developed and maintained exclusively by the current authors. As an initial test of the incremental construction algorithm described in Section 3, we expanded HC for a decision tree containing 3,500 leaves. This yielded a final $\det(HC)$ with 406,173 nodes and 8,199,840 arcs. The run-time and memory usage statistics for our various build scenarios are given in Table 2. These experiments were conducted on an AMD 64-bit work station with 8 Gb of RAM and two Opteron processors running at 1.8 GHz.

As is clear from the results in Table 2, the dynamic expansion of the network reduces memory usage from 7.70 to 1.42 Gb, which represents a factor of 5.42 reduction. This large decline in the size of the task image is accompanied by a modest increase in run time from 50 to 56 minutes. Hence, dynamic expansion of HC is very worthwhile.

4.3. Speech Recognition Experiments

The ASR experiments reported below were conducted with the *Millenium automatic speech recognition* system, which is based on the Enigma WFST library. Like Enigma, Millenium is developed and maintained solely by the current authors. Two token passing decoding algorithms are implemented in Millenium. The *word trace decoder* was developed along the lines suggested by Saon *et al.* [12]; a list of word hypotheses is maintained by each token, whose maximum length N_{hypos} is a parameter of the decoding, along with the beam width. Typical values of N_{hypos} are 5 or 10. During Viterbi search, only the highest scoring word hypotheses is propagated forward when a non-epsilon output (i.e., *word*) symbol is encountered in the recognition network, but links are retained to the other partial hypotheses. When two tokens meet at the same state, their lists of hypotheses are merge sorted such that the final list contains only unique word hypotheses.

For the initial experiments reported here, a fully-continuous acoustic model (FCAM) with 4,000 codebooks was trained. The features for ASR were obtained by extracting frames of 13 cepstral coefficients, then concatenating 15 consecutive frames together. *Linear discriminant analysis* (LDA) [3] was used to reduce the dimensionality of the final feature to a length of 42. The LDA transformation was followed by a global STC transform [5] and global cepstral mean subtraction (CMS).

The training data used for the experiments reported here was taken from the ICSI, NIST, and CMU *meeting corpora*, as well as the *Transenglish Database* (TED) corpus, for a total of 100 hours of training material.

Conventional training of the FCAM system using both *vocal tract length normalization* (VTLN) [2] and *constrained maximum likelihood linear regression* (CMLLR) [4]. Maximum likelihood speaker-adapted training (MMI-SAT) was conducted using the approximation described in [6] to conserve disk space. *Maximum mutual information speaker-adapted training* (MMI-SAT) was conducted as described in [7].

The test set used for the experiments described below was the *lecture meeting* portion of the NIST RT-05s evaluation set. It consisted of 22,258 words, a total of 3.5 hours of data. The lecture speakers spoke in English, but often had pronounced German or other accents. The subject matter was technical in nature, typically about topics related to automatic speech recognition. This data was collected as part of the European Union integrated project, CHIL, *Computers in the Human Interaction Loop* at the Universität Karlsruhe in Karlsruhe, Germany.

In a *semi-continuous acoustic model* [1], several Gaussian mixture models share a single set of Gaussian densities or *codebook*, but have their own unique mixture weights. We undertook a final set of experiments to compare the performance of a semi-continuous acoustic model (SCAM) to that of a fully-continuous acoustic model (FCAM) under identical training condition and testing conditions. For these experiments, we began with a pentaphone FCAM with 4,000 codebooks, and a total of 194,063 Gaussian densities. For the FCAM, we first performed conventional Viterbi training, then maximum likelihood SAT, and finally MMI-SAT as described in Section 4.1. The SCAM systems were obtained by performing additional divisive clustering [15] beginning from the decision tree used for the FCAM systems to produce a final decision tree with 16,000 Gaussian mixture models sharing the same 4,000 codebooks contained in the FCAM system. Conventional, ML-, and MMI-SAT parameter estimation were also performed on the SCAM system.

Because of its size, the HC transducer for the large SCAM system could only be expanded incrementally using the algorithm described in Section 3. The dimensions of the determinized HC networks for the FCAM and SCAM systems are shown in Table 3, along with the execution time and task image size for *incremntal* expansion algorithm. All dimensions of the determinized HC increased by a factor comparable to four-fold

Table 3: *Dimensions of HC for the FCAM and SCAM systems.*

Dimension	FCAM	SCAM	Factor
Nodes	609,302	3,535,569	5.80
Arcs	12,351,544	20,745,716	1.68
Metastates	356,702	1,746,894	4.90
GMM Sequences	60,683	379,156	6.25
Construction Time	2:33 hrs.	82:36 hrs.	32.4
Task Image	2.96 Gb	11.7 Gb	3.95

Table 4: *Word error rates obtained with FCAM and SCAM systems.*

Training and Testing	% Word Error Rate	
	FCAM	SCAM
Unadapted conventional	45.2	
Adapted conventional	37.0	
Adapted ML-SAT	34.1	32.9
Adapted ML-SAT, 4-gram LM		31.9
Adapted MMI-SAT	32.0	31.3

increase in the number of leaves in the distribution tree, with the exceptions of the number of arcs, for which the factor was significantly less, and the construction time, for which the factor was dramatically larger.

The bigram LM used for the experiments described below originally contained 3,015,065 bigrams, which we pruned to 113,705 to construct a recognition network. For lattice rescoring, we also trained a four-gram LM on the same data as the bigram; the four-gram contained a total of 4,714,631 bigrams, 3,559,905 trigrams, and 2,260,500 four-grams.

Word error rates for the FCAM and SCAM systems on the same lecture meeting portion of the NIST RT-05s described in Section 4.1 are shown in Table 4. For each pass of decoding save for the initial unadapted pass, speaker adaptation parameters were estimated using word lattices from the prior pass. As we were unable to statically expand the entire recognition network, the experiments with the SCAM systems were conducted by rescoring the lattices obtained from decoding with the corresponding FCAM system. The word trace decoder was used for lattice writing. After the raw lattice was obtained from the decoder, it was projected onto the output side to discard all arc information except for the word identities, then compacted through epsilon removal, determinization and minimization. The minimum word lattice was then composed with either the appropriate LM, then with the lexicon, and finally with HC to produce a constrained recognition network. The final recognition network was compacted through epsilon removal, determinization and minimization [10], and then used for both MLLR parameter estimation as described by Uebel and Woodland [14], as well as speech recognition. As the word lattice represents a highly constrained search space, we were also able to apply the full four-gram LM during lattice rescoring.

From the results in Table 4 it is clear that the application of both the SCAM system as well as the four-gram LM provided significant improvements over the FCAM system with a bigram LM. Rescoring the word lattices with the SCAM system reduced the WER from 34.1% to 32.9%, which was further reduced to 31.9% with the four-gram The reductions in WER provided by the MMI-SAT models were 2.1% and 1.6% absolute for the FCAM and SCAM systems, respectively.

5. Conclusions

In earlier work, we derived a transducer HC that translates from sequences of Gaussian mixture models directly to phone sequences. The static expansion of HC used in our prior work enabled us to compile a decision tree for a fully-continuous hidden Markov model (HMM) with approximately 3,500 leaves into a transducer. In this work, we have presented an algorithm whereby the initial HC transducer is incrementally expanded and immediately determinized. This technique avoids the need for a full expansion of the initial HC, and thereby provides a large reduction in the size of the task image with only a marginal increase in run time. In future work, we will probe the upper limits of how large a decision tree can be expanded with the incremental build algorithm. We will also provide a run-time analysis of the algorithms described here.

6. References

[1] J. Duchateau, K. Demuynck, D. Van Campernolle, and P. Wambacq. Improved parameter tying for efficient acoustic model evaluation in large vocabulary continuous speech recognition. In *Proc. ICSLP*, 1998.

[2] Ellen Eide and Herbert Gish. A parametric approach to vocal tract length normalization. In *Proc. ICASSP*, 1996.

[3] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1990.

[4] M. J. F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12, 1998.

[5] M. J. F. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions Speech and Audio Processing*, 7:272–281, 1999.

[6] J. McDonough, T. Schaaf, and A. Waibel. On maximum mutual information speaker-adapted training. In *Proc. ICASSP*, 2002.

[7] J. McDonough, Matthias Wölfel, and Emilian Stoimenov. On maximum mutual information speaker-adapted training. *Computer Speech and Language*, to appear.

[8] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2), 1997.

[9] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16:69–88, 2002.

[10] M. Mohri and M. Riley. Network optimizations for large vocabulary speech recognition. *Speech Communication*, 25(3), 1998.

[11] M. Mohri, M. Riley, D. Hindle, A. Ljolje, and F. Periera. Full expansion of context-dependent networks in large vocabulary speech recognition. In *Proc. ICASSP*, volume II, pages 665–668, Seattle, 1998.

[12] G. Saon, D. Povey, and G. Zweig. Anatomy of an extremely fast LVCSR decoder. In *Proc. Interspeech*, Lisbon, Portugal, 2005.

[13] M. Schuster and T. Hori. Efficient generation of high-order context-dependent weighted finite state transducers for speech recognition. In *Proc. ICASSP*, pages 201–204, 2005.

[14] L. Uebel and P. Woodland. Improvements in linear transform based speaker adaptation. In *Proc. ICASSP*, 2001.

[15] S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proc. ICASSP*, 1994.