MODELING POLYPHONE CONTEXT WITH WEIGHTED FINITE-STATE TRANSDUCERS

Emilian Stoimenov and John McDonough

Institut für Theoretische Informatik
Universität Karlsruhe
Am Fasanengarten 5
D-76131 Karlsruhe, Germany
{emilian,jmcd}@ira.uka.de

ABSTRACT

As coarticulation effects are prevalent in all speech, a phone must be modeled in its context to achieve optimal performance in large vocabulary continuous speech recognition systems. Schuster and Hori [7] proposed a technique for modeling polyphone context with weighted finite-state transducers 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 modeling of all possible polyphones is avoided. Rather, Schuster and Hori derive a transducer HC that translates from sequences of Gaussian mixture models directly to phone sequences. The resulting network $HC \circ L \circ G$ is much smaller than the conventional network $H \circ C \circ L \circ G$ proposed by Mohri *et al* [6]. While Schuster and Hori's approach to modeling polyphone context is quite interesting, it is *incorrect* for contexts larger than triphones. In this work, we correct the errors of Schuster and Hori. Thereafter we discuss how the intermediate size of the network HC can be held in check. We also present the results of a set of experiments comparing network size and speech recognition performance for networks obtained with Schuster and Hori's technique and with the correct technique.

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* [6, 4], 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* 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 [5], 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* [3], 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.

Several methods for modeling polyphone context that seek to circumvent the necessity of explicitly modeling the polyphone expansion C have recently appeared in the literature. Chen [1] extended the ideas of Sproat and Riley [8] in using the structure of the decision tree to rewrite the states in a context independent phone model first to the intermediate nodes in the decision tree, then finally to the leaf nodes. Chen's technique first applies the questions concerning the *left* context of a phone, then reverses the graph and applies the questions concerning the *right* context. All questions are formulated as WFSTs, and the necessary rewrites are accomplished through a series of compositions, each of which applies exactly *one* question. To inhibit the uncontrolled growth of network, Chen determinizes the network after the application of every question.

Schuster and Hori [7] 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, like Chen, derive a transducer HC that translates from sequences of Gaussian mixture models directly to phone sequences. The resulting network $HC \circ L \circ G$ is much smaller than the conventional $H \circ C \circ L \circ G$ as the number of valid three-state sequences associated with a given decision tree is far smaller than the number of possible polyphones.

While Schuster and Hori's approach to modeling polyphone context is quite interesting and enjoys the advantage of being conceptually straightforward, as we show in this work, it is *incorrect* for contexts larger than triphones. Indeed, some indication of this can be found in [7], inasmuch as Schuster and Hori claim that—after determinmization and minimization—their technique results in a smaller network than obtained with the conventional expansion $H \circ C \circ L \circ G$. This flatly contradicts the claims of Mohri [3], who maintains that determinizing and minimizing $H \circ C \circ L \circ G$ results in the network with the fewest states and arcs among all deterministic transducers that implement the same function.

In this work, we address the problem of modeling polyphone context with WFSTs. We begin by correcting the errors in the approach of Schuster and Hori and thereafter discuss how the intermediate size of the network HC can be held in check. We also discuss how this approach can be modified to accommodate decision trees that allow compound questions.

2. POLYPHONE MODELING

In this section, we explain how Schuster and Hori's technique [7] can be corrected so as to produce the correct mapping between sequences of Gaussian mixture models (GMMs) and phone sequences. We also consider such details as how this technique can be modified to accommodate decision trees allowing compound questions. Our ASR system distinguishes between regular phones and *word bound-*

This work was supported by the European Union under the integrated project CHIL, *Computers in the Human Interaction Loop*, contract number 506909.

phone	polyphone position 0	1	2	3	4
AH	0	0	1	1	0
{AH:WB}	0	0	1	0	0
B	0	0	0	1	0
{B:WB}	0	1	0	0	1
⋮			⋮		
Z	1	0	0	1	0
{Z:WB}	1	0	0	0	0

Fig. 1. Typical bit matrix corresponding to center phone "AH."

ary 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 [7], 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. As shown in Figure 2, 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.

Let A and B represent two possible questions in a decision tree, and consider a node in a decision tree in which the compound question A AND B is posed. Interpreting the YES clause in the tree is straightforward; it is only necessary to reset all bits for which either A or B is *false*. The NO clause for this question, on the other hand, is not so straightforward, as $!(A \text{ and } B) = !A$ or $!B$. This implies that for the NO clause we must reset the bits where either A or B is *true*, which means that separate bit matrices must be retained to represent the cases $!A$ and $!B$. Hence, to handle decision trees with compound contexts, it is necessary to extend the notion of a bit matrix proposed in Schuster and Hori's original work to include a *list* $\mathbf{L}$ of bit matrices.

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. Let $\mathbf{L}_n = \{\mathbf{B}_i\}$ be a *bit matrix list*. We say two bit matrix lists $\mathbf{L}_n$ and $\mathbf{L}_m$ are equivalent if $\|\mathbf{L}_n\| = \|\mathbf{L}_m\|$ and each $\mathbf{B}_i \in \mathbf{L}_n$ is equivalent to exactly one $\mathbf{B}_j \in \mathbf{L}_m$. We can assume without loss of generality that $\mathbf{B}_i \neq \mathbf{B}_j$ for any $\mathbf{B}_i, \mathbf{B}_j \in \mathbf{L}_m$.

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 $\mathbf{s} = (p, s_1, s_2, s_3, \mathbf{L})$ where $\mathbf{L}$ is the list of valid bit matrices corresponding to the state sequence s_1, s_2, s_3. Let $\mathbf{L}' = \mathbf{L} \gg$ be the list of bit matrices obtained by right shifting each $\mathbf{B} \in \mathbf{L}$ and let $\mathbf{L}'' = \mathbf{L}_n \ \& \ \mathbf{L}_m$ denote the list of *valid* bit matrices obtained by performing the bitwise & operation on each $\mathbf{B}_i \in \mathbf{L}_n$ with every $\mathbf{B}_j \in \mathbf{L}_m$. As in Schuster and Hori [7], we can enumerate a set $\mathbf{S}$ of valid metastates as follows. Begin with a bit matrix list $\mathbf{L}_{s_1}$ corresponding to the leaf node associated with the first state of a three-state sequence for a polyphone with center phone p. Similarly, let $\mathbf{L}_{s_2}$ and $\mathbf{L}_{s_3}$ be the bit matrices for the second and third states for such a three-state sequence for a polyphone with center phone p. If

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

is non-empty, then s_1, s_2, s_3 is a valid three-state sequence and the metastate $(p, s_1, s_2, s_3, \mathbf{L})$ can be added to $\mathbf{S}$. As discussed in [7], all such valid metastates can be enumerated by first enumerating the valid two-state sequences, then building three-state sequences. We say that two metastates are equivalent if they have the same phone p, the same three-state sequence s_1, s_2, s_3 and equivalent bit matrix lists $\mathbf{L}$.

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        L ← (q.L ≫) & s.L
10        if ||L|| > 0:
11          t ← (s.p, s.s1, s.s2, s.s3, L)
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 $\mathbf{L}$ of valid bit matrices in Line 09, and checking if $\mathbf{L}$ is non-empty in Line 10. Note that the right shift $>>$ in Line 09 is to be understood as shifting in a column of *ones*. If $\|\mathbf{L}\| > 0$, 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. This ensures that the connections for each $\mathbf{t} \in \mathbf{T}$ are created exactly *once*. 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.

When a metastate $\mathbf{t}$ is defined as in Line 11 of Listing 1, we will say the $\mathbf{t}$ is *derived* from $\mathbf{s}$. We will denote this relation with the functional notation $\mathbf{s} \leftarrow \texttt{from}(\mathbf{t}, \mathbf{S})$

2.3. Comparison with Schuster and Hori's Algorithm

The critical difference between the algorithm presented in Listing 1 and that of Schuster and Hori [7] lies in the definition of the new metastate $\mathbf{t}$ in Line 11. Whereas the correct algorithm assigns the

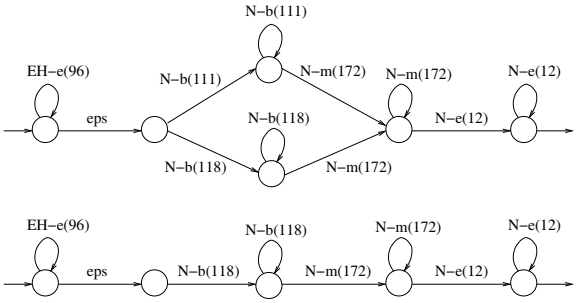

Fig. 2. Pentaphone example showing difference in network produced with Schuster and Hori's algorithm and the correct algorithm.

newly-created list $\mathbf{L}$ of bit matrices to $\mathbf{t}$, Schuster and Hori's algorithm assigns the previously-existing $\mathbf{s}.\mathbf{L}$. Hence, in Schuster and Hori's algorithm the final set of metastates $\mathbf{T}$ is identical with the initial set $\mathbf{S}$ obtained from the metastate enumeration step; i.e., Schuster and Hori's algorithm discards the phonetic context information from $\mathbf{q}.\mathbf{L}$ which is contained in $\mathbf{L}$ but *not* in $\mathbf{s}.\mathbf{L}$. From Line 11, it is clear that the set of bits in $\mathbf{L}$ that are one is a subset of the bits in $\mathbf{s}.\mathbf{L}$ that are one. As a one in $\mathbf{L}$ holds out the possibility of forming a connection when $\mathbf{t}$ is subsequently expanded, a moment's thought will reveal that Schuster and Hori's algorithm will produce metastates that have connections which should *not* exist. The correct algorithm, on the other hand, will produce *more* metastates, as many metastates will share the same phone and three-state sequence, but be differentiated by the list of bit matrices. These metastates, however, will have the correct set of connections and no others.

As explained in the introduction, Schuster and Hori's algorithm is actually *correct* for triphone contexts. This is readily seen when considering that, for triphones, $\mathbf{L}$ and $\mathbf{s}.\mathbf{L}$ as defined in Line 09 of Listing 1 are equivalent in all but the 0th position, which will be shifted out when these matrices are compared to others for the purpose of forming metastate connections.[1]

The difference between Schuster and Hori's algorithm and the correct algorithm can be readily seen with a small example. In Figure 2 is a portion of the network obtained by composing the word "PEN" with the $HC \circ L \circ G$ obtained with both Schuster and Hori's algorithm, as well as the correct algorithm. In the figure, the transitions are labeled with the names of GMMs; e.g., "EH-e(96)" is the 96th cluster for the *end* state of phone EH. As is apparent from the figure, the network generated with Schuster and Hori's algorithm allows the transition "N-b(111)," which is *not* allowed by the network generated with the correct algorithm, nor by the conventional network $H \circ C \circ L \circ G$ in which all polyphone contexts are explicitly expanded.

2.4. Bit Masks

The algorithm presented in Section 2.2 is correct but impractical, inasmuch as for any reasonably sized decision tree, the number of metastates will quickly become intractably large and deplete all available memory; the algorithm does not finish. Here we consider two modifications to the algorithm: the first is a pure speedup, the second limits the growth of metastates.

Consider a metastate $\mathbf{s} \in \mathbf{S}$ where $\mathbf{S}$ is once more the set of metastates obtained from the metastate enumeration algorithm. The set

$$\mathbf{N}(\mathbf{s}, \mathbf{S}) = \{\mathbf{n} \in \mathbf{S} : \|(\mathbf{s}.\mathbf{L} >>) \ \& \ \mathbf{n}.\mathbf{L}\| > 0\}$$

is readily seen to be the list of *possible* following metastates for any $\mathbf{t} \in \mathbf{T}$ derived from $\mathbf{s}_F \leftarrow \texttt{from}(\mathbf{t}, \mathbf{S})$. Hence, searching only over $\mathbf{N}(\mathbf{s}_F, \mathbf{S})$ in Line 08 of Listing 1 instead of over all $\mathbf{S}$ results in a significant speedup. Moreover, in so doing, we run no risk of omitting any possible connections from $\mathbf{t}$: If $\mathbf{t}$ is derived from $\mathbf{s}$ as in Line 11 of Listing 1, then the bit matrices appearing in $\mathbf{t}.\mathbf{L}$ can only have ones in a subset of the positions where $\mathbf{s}.\mathbf{L}$ has ones. The implies that $\mathbf{t}$ will connect only to a subset of those metastates derived from the elements of $\mathbf{N}(\mathbf{s}, \mathbf{S})$.

Listing 2 Bit mask function.

```
00  def bitMask(s, S):
01    set all bits in M to zero
02    foreach n in N(s, S):
03      foreach M' in n.L:
04        M ← (M | M')
05    return (M <<)
```

Consider the definition of the function `bitMask` in Listing 2. The left shift operation in Line 05 of this listing is to be understood as shifting in a column of *zeros*. It is not difficult to see that Line 09 of Listing 1 can be replaced with

$$\mathbf{L} \ \leftarrow \ (\mathbf{q}.\mathbf{L} \ \gg) \ \& \ \mathbf{s}.\mathbf{L} \ \& \ \mathbf{M}$$

where $\mathbf{M} \leftarrow \texttt{bitMask}(\mathbf{s}, \mathbf{S})$: Applying $\mathbf{M}$ to the prior definition of $\mathbf{L}$ unsets those bits that will be unset in any event as soon $\mathbf{L}$ is right shifted and multiplied with any $\mathbf{n} \in \mathbf{N}(\mathbf{s}, \mathbf{S})$. Leaving these bits set only causes an unneeded increase in $\|\mathbf{T}\|$, inasmuch as metastates in $\mathbf{T}$ which are essentially equivalent will be treated as different; these metastates will be combined when HC is determinized and minimized.

With the two changes described here, the metastate connection algorithm can now be reformulated as in Listing 3. The most important differences between Listings 1 and 3 lie in the loop over $\mathbf{N}(\mathbf{s}_F, \mathbf{S})$ in Lines 08-09 of the latter, which provides a speedup, and the application of the bit mask in Lines 10-11, which inhibits the growth of $\|\mathbf{T}\|$. For efficiency, $\mathbf{N}(\mathbf{s}, \mathbf{S})$ and $\texttt{bitMask}(\mathbf{s}, \mathbf{S})$ are precalculated for all $\mathbf{s} \in \mathbf{S}$ and stored, so that Lines 09 and 10 only involve table lookups.

Listing 3 Efficient 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      s_F ← from(q, S)
09      foreach s ∈ N(s_F, S):
10        M ← bitMask(s, S)
11        L ← (q.L ≫) & s.L & M
12        if ‖L‖ > 0 :
13          t ← (s.p, s.s_1, s.s_2, s.s_3, L)
14          if t ∉ T:
15            add t to T
16            push t on Q
17          e ← (q.s_3, t.s_1, t.s_1.g, t.p)
18          add e to E
19    return (T, E)
```

[1] For our system, Schuster and Hori's algorithm produces incorrect results even for triphones due to the presence of the word boundary markers. This can cause AH and {AH:WB}, for example, to both be valid center phones for a given metastate. This in turn can lead to a subsequent difference when the full phonetic context is considered.

Network	S&H States	S&H Arcs	Correct States	Correct Arcs
HC	112,878	4,378,729	1,979,871	85,610,928
$\det(HC)$	53,920	747,748	812,533	12,740,381
$\min(\det(HC))$	26,902	257,430	91,784	894,163
R	138,628	320,891	167,632	372,501
$\det(R)$	1,836,077	3,970,263	2,122,747	4,317,925
$\min(\det(R))$	99,183	236,149	102,420	234,769

Table 1. Pentaphone network sizes for Schuster and Hori's (S&H) algorithm and the correct algorithm. Here, $R = \min(\det(HC)) \circ \det(L \circ G \circ W)$, where W is a word minimized lattice with 14,239 states and 27,993 arcs.

Given that state clustering sometimes ignores state boundary information, which means that a regular phone and its state boundary version in a given context will have the same three-state sequence and same bit matrix list, we made a final modification of the metastate connection algorithm to efficiently handle word boundary phones: If a metastate $\mathbf{t}$ has a center phone with a word boundary marker, and $\mathbf{t}$ does not exist in $\mathbf{T}$, we search for the same metastate but for a non-boundary center phone. If this is found, then both versions of the metastate can share the same three-state and bit matrix list, and these structures need not be allocated. This modification reduces both the size of $HC \circ L \circ G$ as well the computational expense of the subsequent determinization.

3. EXPERIMENTS

Table 1 shows the sizes of a pentaphone network HC after each stage in the construction using both Schuster and Hori's (S&H) algorithm as well as the correct algorithm. Observe that the network HC produced by the correct algorithm initially has more than an order of magnitude more states and arcs than that produced by Schuster and Hori's algorithm. This difference is greatly reduced, however, by subsequent determinization and minimization, so that the final minimized networks differ in size by a factor of approximately 3.4. For rescoring purposes, we next composed a word lattice W with G and L, then with HC. Once more, the initial difference in $R = \min(\det(HC)) \circ \det(L \circ G \circ W)$ was reduced through subsequent minimization and determinization. In the end, the correct recognition network had only *marginally* more states than the S&H network, and actually had *fewer* arcs. These results are in agreement with our contention that the HC transducer produced with the S&H algorithm has too few metastates, and these metastates allow a superset of the correct connections.

The experiments reported below were conducted with the *Millenium automatic speech recognition* system, which is developed and maintained by the authors and students at the University of Karlsruhe in Karlsruhe, Germany. The recognition and training modules of Millenium are based entirely on a FST library. For the experiments reported here, a fully-continuous ASR with 3,500 codebooks was trained. The features for ASR were obtained by extracting vectors 13 cepstral coefficients, then concatenating 15 consecutive frames together. Linear discriminant analysis was used to reduce the dimensionality of the final feature to 42. Semi-tied covariances and cepstral mean normalization were also used.

The test set used to evaluate the algorithms proposed here contains approximately 2.5 hours of audio and video data recorded during five seminars by students and faculty at the University of Karlsruhe (UKA) in Karlsruhe, Germany. The seminar 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*. The total size of the test set was approximately 17,000 words.

	% Word Error Rate S&H	% Word Error Rate Correct
Pentaphone	37.7	37.6

Table 2. Word error rates from a set of lattice rescoring experiments.

Table 2 shows word error rates from a set of lattice rescoring experiments conducted on the CHIL data. For these experiments, we built a combined pentaphone HC using both the correct method, as well as the S&H technique. The unadapted first pass system used to generate the lattices achieved a word error rate (WER) of 39.9%. For rescoring, a system adapted with maximum likelihood linear regression (MLLR) [2] was used. The system had been adaptively trained with MLLR parameters.

As is apparent from Table 2, the final difference in WER achieved by the correct and S&H systems is vanishingly small, with a slight advantage going to the correct algorithm. This seems to be in good agreement with the small difference in final network sizes seen in Table 1.

4. CONCLUSIONS

Schuster and Hori [7] proposed a technique for modeling polyphone context with weighted finite state transducers whereby all valid three state-sequences of Gaussian mixture models are enumerated, and thereafter the possible connections between these three-state sequences are calculated. Hence, the explicit modeling of all possible polyphones is avoided. While Schuster and Hori's approach to modeling polyphone context is interesting and conceptually straightforward, it is incorrect for contexts larger than triphones. In this work, we have corrected the errors of Schuster and Hori. We have also presented the results of a set of experiments comparing network size and speech recognition performance for networks obtained with both Schuster and Hori's technique and the correct technique.

5. REFERENCES

[1] S. F. Chen. Compiling large-context phonetic decision trees into finite-state transducers. In *Proc. Eurospeech*, pages 1169–1172, 2003.

[2] C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech and Language*, pages 171–185, 1995.

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

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

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

[6] 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.

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

[8] R. Sproat and M. Riley. Compilation of weighted finite-state transducers from decision trees. In *Proc. ACL*, Santa Cruz, California, June 1996.