Available online at www.sciencedirect.com

SciVerse ScienceDirect

Computer Speech and Language 28 (2014) 177–191

COMPUTER SPEECH AND LANGUAGE

www.elsevier.com/locate/csl

Direct construction of compact context-dependency transducers from data[☆]

David Rybach[b,*,1], Michael Riley[a], Chris Alberti[a]

[a] Google Inc., 76 Ninth Avenue, New York, NY, USA

[b] Human Language Technology and Pattern Recognition, Computer Science Department, RWTH Aachen University, Germany

Received 15 June 2012; received in revised form 18 January 2013; accepted 17 April 2013
Available online 7 May 2013

Abstract

This paper describes a new method for building compact context-dependency transducers for finite-state transducer-based ASR decoders. Instead of the conventional phonetic decision tree growing followed by FST compilation, this approach incorporates the phonetic context splitting directly into the transducer construction. The objective function of the split optimization is augmented with a regularization term that measures the number of transducer states introduced by a split. We give results on a large spoken-query task for various n-phone orders and other phonetic features that show this method can greatly reduce the size of the resulting context-dependency transducer with no significant impact on recognition accuracy. This permits using context sizes and features that might otherwise be unmanageable.
© 2013 Elsevier Ltd. All rights reserved.

Keywords: WFST; LVCSR

1. Introduction

Weighted finite state transducers (WFST) are widely used in speech recognition applications (Mohri et al., 2008). They allow for a unified representation of all knowledge sources involved as well as their combination and optimization. The language model is represented by a transducer $\mathcal{G}$, $\mathcal{L}$ is a phone to word transducer derived from the pronunciation dictionary, and $\mathcal{C}$ encodes the context-dependency of the acoustic models. These transducers are combined by the finite-state operation of composition as $\mathcal{C} \circ \mathcal{L} \circ \mathcal{G}$ to form a very efficient recognition transducer.

This paper focuses on the construction of the context-dependency transducer $\mathcal{C}$. $\mathcal{C}$ rewrites context independent (CI) phone sequences to sequences of context dependent (CD) phone models. Because of the composition with $\mathcal{L}$, the output alphabet is the set of phones used, and the CD phones or their acoustic model representations occur as input labels.

In the most straight-forward construction, p^{n-1} states (one for every $n-1$-gram) and p^n transitions (one for every CD n-phone model) are used to represent the context-dependency of p phones with n the context size, e.g. p^2 states

[☆] This paper has been recommended for acceptance by Srinivas Bangalore.
[*] Corresponding author. Tel.: +49 2418021601; fax: +49 2418022219.
E-mail address: rybach@cs.rwth-aachen.de (D. Rybach).
[1] Work was partly performed at Google. D. Rybach is now with Google Inc., New York, NY, USA.

0885-2308/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.csl.2013.04.006

and p^3 transitions for triphones (Mohri et al., 2008). For larger contexts or further dependencies like word boundaries and vowel stress, this method can become unwieldy.

Phonetic decision trees are commonly used to tie the parameters of context dependent units, because the training data is usually not sufficient to build all context dependent models and the model complexity has to be balanced against data availability (Bahl et al., 1991; Young et al., 1994). This tying of models defines classes of equivalent contexts. The construction of $\mathcal{C}$ can account for these classes of equivalent contexts, yielding a more compact transducer (Riley et al., 1997). Hence, the number of states required depends on the structure of the trees.

Several papers have proposed more efficient constructions of context-dependency transducers. A general compilation method for finite state transducers from decision trees is described in Sproat and Riley (1996). The authors of Schuster and Hori (2005a,b), Stoimenov and McDonough (2006) describe an efficient construction for high-order context dependent models that builds a minimal $\mathcal{C}$. Chen (2003) presents an on-demand transducer to represent 11-phone decision trees. An "arc minimization" technique is used in Yvon et al. (2004) to allow for a full word of cross-word contexts.

These approaches all construct a decision tree first, not taking advantage of a key observation: slight modifications in the tree construction that do not affect the quality of the acoustic models might have a big impact on the size of the $\mathcal{C}$ transducer. In this paper a new construction method is presented that builds a compact $\mathcal{C}$ transducer directly from the training data omitting the separate explicit construction of decision trees that precedes their compilation into a transducer. The algorithm to tie parameters of CD models is modified in order to allow us to control the size of the resulting transducer. Optimizing the models based only on the transducer size would not consider the similarities among acoustic units. On the other hand, relying solely on acoustic properties might produce large transducers. Therefore, both criteria are incorporated in the objective function used to optimize the set of CD models with a parameter to control their relative contribution. In this way, we can tradeoff model accuracy and precise context-dependency transducer size in a way not possible with previous methods. We give results on a large spoken-query task for various n-phone orders and other phonetic features that show this method can greatly reduce the size of the resulting context-dependency transducer with no significant impact on recognition accuracy. The size of the right phonetic context is limited to one phone though, as explained in Section 8.

This paper continues the investigations and describes advancements of the method presented previously in Rybach and Riley (2010).

The remainder of this paper is organized as follows. Section 2 introduces finite-state transducers and the notation used. Section 3 describes conventional phonetic decision tree methods and their relation to our proposed method. Section 4 describes the proposed, direct context-dependency transducer construction, which is developed further in Section 5. Section 6 explains the implementation. Section 7 presents the experimental results on a large corpus of spoken queries. Section 8 discusses the methods and results.

2. Preliminaries

A *finite-state transducer* $\mathcal{T} = (\Sigma_i, \Sigma_o, Q, I, F, E)$ is specified by a finite input alphabet Σ_i, a finite output alphabet Σ_o, a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, and a finite set of transitions (or *arcs*) $E \subseteq Q \times (\Sigma_i \cup \{\epsilon\}) \times (\Sigma_o \cup \{\epsilon\}) \times Q$. $E[q]$ denotes the set of transitions leaving state $q \in Q$, and $I[q]$ the set of transitions to state q. An introduction to *weighted finite-state transducers* can be found in Mohri et al. (2008). Weights are trivial in the transducers considered in this work and hence omitted.

Given a transition $e \in E$, $p[e]$ denotes its origin or previous state, $n[e]$ its destination or next state, $i[e]$ its input label, and $o[e]$ its output label. We denote the transitions with an input label a as $E(a) = \{e : \quad i[e] = a\}$ and the corresponding (previous) states as $Q(a) = \{p[e] : e \in E(a)\}$.

Transducer *composition* is the generalization of finite-state automata intersection. It is used to build complex transducers by combining simpler ones. A transducer is *deterministic* if no state has two outgoing transitions with the same input label. The *weighted determinization* algorithm is similar to the powerset construction for automata. Transducer *minimization* generates for a deterministic weighted transducer an equivalent weighted transducer with minimal number of states. All transducer operations mentioned are described in Mohri et al. (2008).

In the following, Σ denotes the phone alphabet, i.e. the set of CI phone symbols, and $\pi \in \Sigma$ refers to a phone.

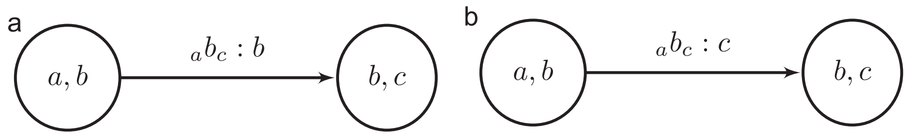

Fig. 1. Transition in a straight forward constructed $\mathcal{C}$ transducer: (a) non-deterministic and (b) deterministic.

3. Decision tree construction

The proposed method generates both the $\mathcal{C}$ transducer and the context dependent tied HMM state models in one optimization procedure. A separate explicit training of a decision tree is not required. Nevertheless, before describing the transducer construction in detail, we give a review of phonetic decision trees and conventional $\mathcal{C}$ transducer construction for a better comparison and describe which steps are shared with our new construction.

3.1. Phonetic decision trees

In many state-of-the-art LVCSR systems, phonetic decision trees are used to tie the parameters of CD phone models (Young et al., 1994). The decision tree clusters acoustically similar context dependent units using a top-down approach. A binary question is attached to each inner node of the tree, while the leaf nodes correspond to the tied models. The decision tree allows us to assign a CD model also to allophones which have not been observed in the training data.

The splitting at the inner nodes is based on phonetic properties of the phones in a specific context position (left and right in the triphone case), for example "is the left phone a vowel?". As the questions are binary, the phonetic properties can be defined as sets of phones, e.g. the set of vowels.

A widely used approach is to build separate trees for each HMM state of each phone, thereby limiting the sharing of parameters to models of the same phone and HMM state. Other approaches build a single tree and include questions about the center phone and the HMM state position (Beulen et al., 1997).

A decision tree is trained according to a maximum likelihood criterion using a greedy optimization strategy (Breiman et al., 1984). First, observations are collected for the HMM states of all allophones occurring in the training data. Each observation is assigned to a leaf node of the corresponding tree. A tree is initialized with one leaf node representing a monophonic model. In each iteration of the sequential optimization, the best split is chosen among all current models (leaf nodes), phonetic properties, and context positions. A split is rated by the gain in acoustic likelihood obtained by introducing the two new models. The likelihood of the training data is computed using the tied leaf models. Observations assigned to the split node are divided among the two new leaf nodes according to the selected phonetic property.

Several stopping conditions may be used – limiting the minimal achieved gain in likelihood, the number of seen observations for a model, or the total number of leaf nodes in the tree. Note that the total number of models can be limited only if all trees are optimized jointly or a single tree is used.

3.2. $\mathcal{C}$ transducer

The C transducer is used to substitute context independent phones with context dependent phones or phone models. The substitution is implemented by applying transducer composition as $\mathcal{C} \circ \mathcal{L}$ (Riley et al., 1997). Therefore, $\mathcal{C}$ has context independent phones as output labels and the context dependent units are used as input labels.

In order to apply a context dependent mapping, the states have to encode the context of a phone. In the straight forward construction for triphonic models we use a state for every pair of phones and a transition for every model. For example, the model for the triphone ${}_a b_c$ (center phone b with a and c as left and right context respectively) would occur as input of the transition from state (a, b) to (b, c) with output label c, as depicted in Fig. 1(a). States encode the phone most recently read and the phone to be read next in this case.

However, this construction would yield a transducer with non-deterministic output labels. Using a non-deterministic transducer, specifically a transducer having multiple arcs with the same output label leaving the same state, increases the complexity of the composition algorithm (Riley et al., 1997). Therefore, it is important to construct a $\mathcal{C}$ transducer with deterministic output labels, which can be achieved by using the right context phone as output label, as in Fig. 1(b).

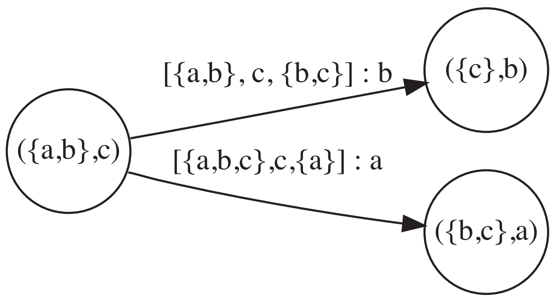

Fig. 2. Example of a part of a $\mathcal{C}$ transducer for 3 phones. Two phone models are shown.

Thus, in this case, the states encode the phones read so far. This introduces a delay of context dependent labels by one phone and requires a sequence end symbol on the context independent side (Mohri and Riley, 1999).

The construction described so far does not account for the context classes defined by the decision tree, but creates transitions for all untied allophones. The number of states (and therefore also the number of arcs) can be reduced by merging states with equivalent contexts. Consider for example the models of a phone b. If the decision trees for b contain no node that splits depending on whether the left context is a or c, then the states (a, b) and (c, b) can be merged. This property is used in existing compilation methods (Sproat and Riley, 1996; Schuster and Hori, 2005b) as well as in our construction method (cf. Section 4).

3.3. *Implicit decision trees*

Building a decision tree explicitly in a separate step is not necessary if the tree is used solely to construct a context dependency transducer, which rewrites sequences of phones to sequences of phone models. The tree based classifier is encoded in the transducer. The transducer assigns context dependent labels based on the context, which is encoded in its states and the transition labels.

Therefore, we can split the models in a similar way as in the decision tree training and encode the context-dependency of the models directly in the transducer. The step-wise construction of the transducer during the iterative splitting of the models makes it possible to incorporate the size of the transducer in the split selection.

The construction uses a greedy optimization strategy. The steps are identical to building a single phonetic decision tree for all phones, except the tree is not explicitly built and a different objective function is used.

The objective function used to rate a split t is

$$L(t) = G(t) - \alpha \cdot S(t)$$

where $G(t)$ is the gain in acoustic likelihood achieved by this split (equal to the gain used in decision tree growing), $S(t)$ is the number of new states required to distinguish the two split models in the $\mathcal{C}$ transducer, and α is a weight controlling the impact of the transducer size. Setting $\alpha = 0$ will produce a result equivalent to a decision tree based construction, while $\alpha = \infty$ will ignore acoustic properties of the model. $S(t)$ can be interpreted as a regularization term.

After choosing a split, the change in context dependency is applied to the transducer, which is described in the following section. The construction is initialized with monophone models, as it is done for conventional phonetic decision tree growing.

4. $\mathcal{C}$ transducer construction

In the previous section, we described the greedy optimization of phonetic splits and the objective function used to select among them. In this section we describe how the $\mathcal{C}$ transducer is built at each step in the iteration and how the number of new states $S(t)$, used in the objective function, is computed. Before providing a more formal description of the $\mathcal{C}$ construction, we begin with a simple example of part of a $\mathcal{C}$ transducer for three phones a, b, c as it might appear during its construction, as shown in Fig. 2.

An input label on a transition in Fig. 2 is a specification of a CD phone model. It is denoted, in general, as $[C_{-L}, \ldots, C_{-1}, \pi, C_1]$, describing a set of $(L+2)$-phonic models for phone π with tied parameters. For example, the label

$[\{a, b\}, c, \{b, c\}]$ is the model for the triphones ${}_a c_b$, ${}_a c_c$, ${}_b c_b$, ${}_b c_c$, Such phone models correspond to leaf nodes in a decision tree (more specifically to leaf nodes in the decision trees for a single phone and all its HMM states). The context sets $C_l \subseteq \Sigma$ reflect the phone properties used so far for splitting or – in the decision tree analogy – to the path leading to a leaf node. Note that we only consider a right context of one phone in this paper (see Section 8).

An output label on the transitions in Fig. 2 is a CI phone taken from the rightmost CD context. Choosing the rightmost phone ensures that $\mathcal{C}^{-1}$ is deterministic and thus that $\mathcal{C}$ composes efficiently with the lexicon.

A state in Fig. 2 represents the sequence of phones read so far required to disambiguate CD phone models. For instance, state $(\{a, b\}, c)$ denotes that the phone labels of all paths reaching it end with either ac or bc. In the conventional construction of $\mathcal{C}$ described in Section 3.2, there would be a state for every $n-1$ phones read. However, not all possible phone contexts have to be considered for all phone models. Consider for example the phone c which is represented by two triphonic models $m_1 = [\{a, b\}, c, \{b, c\}]$, $m_2 = [\{c\}, c, \{b, c\}]$. Only two states, $(\{a, b\}, c)$ and $(\{c\}, c)$, are required two distinguish between these two phone histories.

In the following, a state in $\mathcal{C}$ is represented by a tuple $(H_L, \ldots, H_1, \pi)$ of history sets $H_l \subseteq \Sigma$ and a center phone $\pi \in \Sigma$, where Σ the set of CI phones, and $\mathcal{L}$ is the number of left contexts as discussed above.

While the model construction deals with HMM state models, the $\mathcal{C}$ transducer handles phone models consisting of a sequence of HMM state models. Each of the HMM state models may occur in several phone models. For example, the phone models for the triphones ${}_a c_b$ and ${}_a c_c$ may share the same models for HMM states 1 and 2, but have distinct state models for their third HMM state. Splitting a state model therefore involves splitting all phone models sharing this state model. The transducer state splitting algorithm is applied for each of the split phone models.

4.1. Triphonic contexts

We consider triphone models first and generalize the algorithm afterwards.

Given a split of a phone model m to models m_1 and m_2, the first step is to identify the affected states $q \in Q(m)$. Those are the states that have to be modified, because they have an outgoing transition with the split model as input label.

For splits based on the right context, it suffices to change the input labels of the affected transitions, because the right context is solely encoded (as output label) in the transition itself. We relabel the outgoing transitions $e \in E[q]$ depending on the membership of the output label $o[e]$ in one of the split context sets:

$$i[e] \leftarrow \begin{cases} m_1 & \text{if} \quad i[e] = m, o[e] \in C'_1 \\ m_2 & \text{if} \quad i[e] = m, o[e] \in C''_1 \\ i[e] & \text{if} \quad i[e] \neq m \end{cases} \tag{1}$$

with $m_1 = [C_{-1}, \pi, C'_1]$, $m_2 = [C_{-1}, \pi, C''_1]$.

If the split is performed on the left context with $m_1 = [C'_{-1}, \pi, C_1]$, $m_2 = [C''_{-1}, \pi, C_1]$ we must ensure that states with outgoing transition e having $i[e] \in \{m_1, m_2\}$ restrict the left contexts to C'_{-1} and C''_{-1} or to a respective subset. Therefore, for a state $q = (H, \pi)$, the states $q_1 = (H \cap C'_{-1}, \pi)$ and $q_2 = (H \cap C''_{-1}, \pi)$ are created if they do not exist yet and q is removed.

An incoming transition $e = (p[e], [\hat{C}_{-1}, \sigma, \hat{C}_1], \pi, q) \in I[q]$ is redirected to one of the new states depending on whether the center phone σ in the arc's input label model complies with the particular history set of the state:

$$n[e] \leftarrow \begin{cases} q_1 & \text{if} \quad \sigma \in C'_{-1} \\ q_2 & \text{if} \quad \sigma \in C''_{-1} \end{cases} \tag{2}$$

The outgoing transitions of q_1 and q_2 are the same as for q, but the input labels have to be adjusted. An outgoing transition $e_j = (q_j, i_j, o[e_j], n[e_j]) \in E[q_j], j \in \{1, 2\}$ is updated by

$$i_j \leftarrow \begin{cases} m_j & \text{if} \quad i[e_j] = m \\ i[e_j] & \text{if} \quad i[e_j] \neq m \end{cases} \tag{3}$$

Self-loop transitions $e = (q, [\hat{C}_{-1}, \pi, \hat{C}_1], \pi, q)$ require a special treatment. Arcs $e_j = (q_j, i_j, o[e_j], n[e_j])$ are added with i_j according to Eq. (3) and $n[e_j]$ corresponding to Eq. (2).

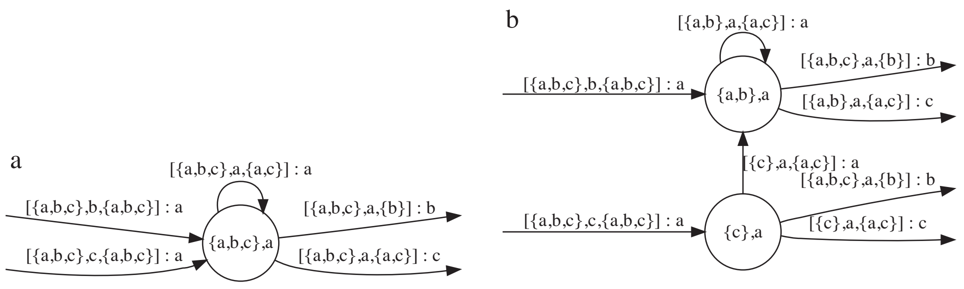

Fig. 3. A state before (left) and after (right) splitting the model $[\{a, b, c\}, a, \{a, c\}]$ at position -1 into $\{a, b\}$ and $\{c\}$.

An example of a state split is shown in Fig. 3. Here the model to split occurs on two transitions, one of them is a self-loop. The state $(\{a, b, c\}, a)$ is split into states with history sets $\{a, b\}$ and $\{c\}$ respectively, according to the left context of the new models. The incoming transitions are redirected based on the center phone of their input label models. The outgoing transitions are replicated for both new states, with updated input labels for the transition with output label c. The former self-loop transition is duplicated as well with updated input label. A loop on state $(\{c\}, a)$ is not valid, so the transition has state $(\{a, b\}, a)$ as next state.

The initial transducer has one state for each phone π with an unconstrained history set $H = \Sigma$. For every pair of phones π and σ the transducer has a transition $(\pi, m_\pi, \sigma, \sigma)$ with a monophonic model $m_\pi = [\Sigma, \pi, \Sigma]$.

4.2. Wider contexts

For larger left contexts, we have to ensure that only valid paths are included in the transducer. For two consecutive states $p = (G_L, \ldots, G_1, \sigma)$ and $q = (H_L, \ldots, H_1, \pi)$ with $\exists e \in E[p] : n[e] = q$ the history sets have to be compatible: $G_{l+1} \subseteq H_l$ for $l = 1, \ldots, L-1$ and $\sigma \in H_1$. That is, for a state encoding a certain set of phones H_l as left context at position $-l$, all predecessor states have to limit the phone history at position $-l+1$ accordingly. See Fig. 4 for an illustration.

In order to maintain these properties, a split of a state q based on the left context $-l$ requires splitting all states that can reach q with at most $(l-1)$ transitions. Splitting the history set H_l of q implies splits of the history set G_{l-1} of all predecessor states p. We perform the splitting of predecessor states recursively on l until $l = 1$ is reached. Due to loop transitions and larger cycles in the transducer, a state may be split several times on different context positions. Thus, splitting a single state based on context position $-l$ may be split into up to 2^l new states.

An example of a split in a 4-phone transducer is shown in Fig. 5. The state $(\{a, b, c\}, \{a, b, c\}, b)$ is split into 4 new states, because of the self-loop transitions. State $(\{a, b, c\}, \{a, b, c\}, a)$ is split into 2 states on context position -1.

4.3. Counting states

Counting the number of new states required for an HMM state model split, i.e. computing $S(t)$ in the objective function, is performed very often during the optimization procedure and should therefore be efficient. We can compute the number of states required without modifying the actual transducer and without computing transitions.

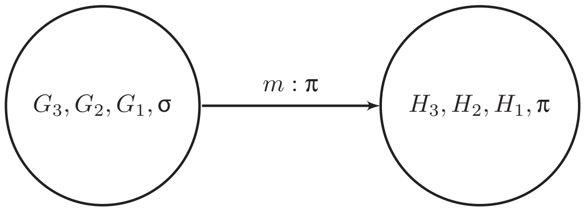

Fig. 4. Transition in a 5-phonic $\mathcal{C}$ transducer.

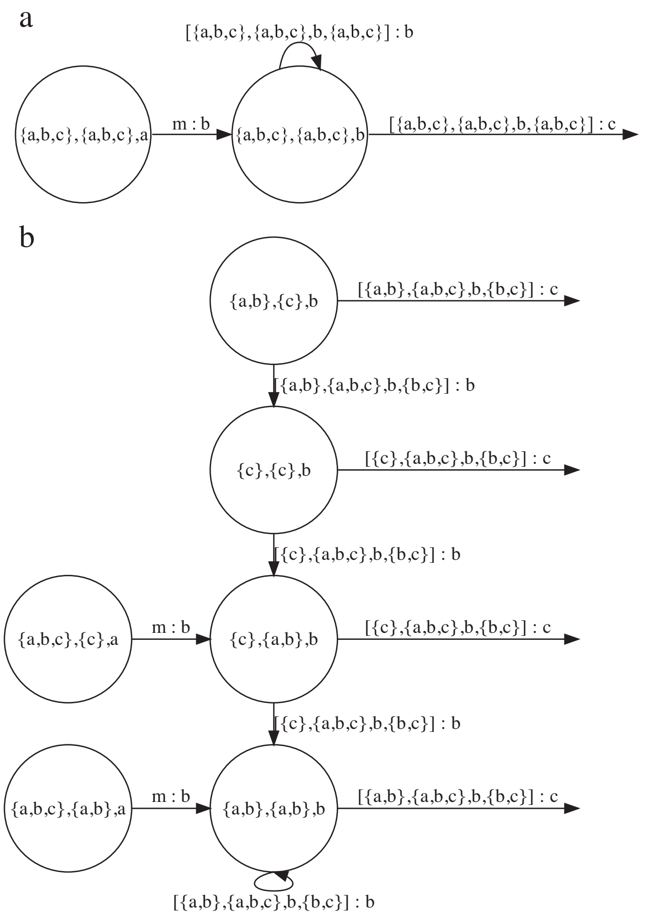
a
{a,b,c},{a,b,c},a
m : b
{a,b,c},{a,b,c},b
[{a,b,c},{a,b,c},b,{a,b,c}] : b
[{a,b,c},{a,b,c},b,{a,b,c}] : c
b
{a,b},{c},b
[{a,b},{a,b,c},b,{b,c}] : c
[{a,b},{a,b,c},b,{b,c}] : b
{c},{c},b
[{c},{a,b,c},b,{b,c}] : c
[{c},{a,b,c},b,{b,c}] : b
{a,b,c},{c},a
m : b
{c},{a,b},b
[{c},{a,b,c},b,{b,c}] : c
[{c},{a,b,c},b,{b,c}] : b
{a,b,c},{a,b},a
m : b
{a,b},{a,b},b
[{a,b},{a,b,c},b,{b,c}] : c
[{a,b},{a,b,c},b,{b,c}] : b

Fig. 5. Part of a 4-phone transducer before (top) and after (bottom) splitting model $[\{a, b, c\}, \{a, b, c\}, b, \{a, b, c\}]$ at position −2 into $\{a, b\}$ and $\{c\}$. m is an arbitrary model for phone a.

For splits of the right context, the number of new states is always 0, because we only need to relabel arcs. For a hypothesized split of model m at (left) context position $-l$ we proceed as follows. Starting from the set of affected states $Q_l = Q(m)$, we compute the sets of affected predecessor states Q_i:

$$Q_i = \{q \in Q : \exists e \in E[q], \quad n[e] \in Q_{i+1}\} \quad \text{for} \quad i = l-1, \ldots, 1$$

Q_i contains all states with a transition to one of the states in Q_{i+1}. Then for each state q in these state sets, starting with Q_1, we compute the required split states q_1, q_2 by intersecting the state's history set with the phone property sets at the appropriate context position. If both q_1 and q_2 do not exist, the count is incremented. Due to loop transitions, q might occur in several sets. Therefore, we have to update all following sets Q_j, $j = i + 1, \ldots, l$ by removing q and adding q_1 and q_2: $Q_j = (Q_j \cup \{q_1, q_2\}) - \{q\}$.

Although we will not prove it here, each step in the construction produces a minimal deterministic automaton (i.e. when the input and output label pair is considered as a single label). As such, $S(t)$ is an intrinsic measure of the transduction.

4.4. Generalized features

The proposed framework allows us to incorporate more complex features for the model splitting than just the phone identity. Examples for these features are word boundary information, syllable identity or even speaker gender. By incorporating, for example, word boundary information into the phone models different phone models can be generated depending on whether a phone occurs at beginning, end, or inside of a word.

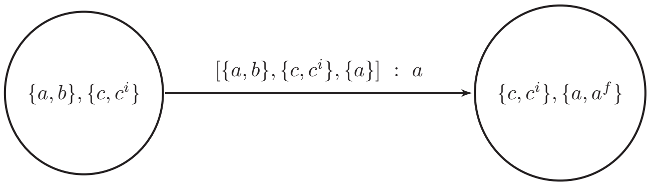

Fig. 6. Transition in a triphone transducer with word boundaries. c^i is an initial c phone and a^f is a final a phone.

A simple method to achieve word-boundary modeling is to introduce separate phone symbols for each of these phone variants and to modify the pronunciation dictionary, i.e. the $\mathcal{L}$ transducer, accordingly. In order to keep the number of phone models small, initial, final, and interior variants of a phone are assigned to the same phone model initially. New phone properties ("is initial phone", "is final phone", "is inside phone") are introduced to allow for splits separating these phone variants.

To keep the number of states in $\mathcal{C}$ small, different states for a phone variant are created only if they need to be distinguished, i.e. if different models exist for the different occurrences. That requires modifying the state representation to include a *center phone set* H_0, initialized with the phone variants, and allowing splits on these sets as well. The state splitting and counting are modified such that the center phone sets are considered when evaluating the validity of paths.

Fig. 6 shows an example for a triphone transducer with word boundaries. The model $[\{a, b\}, \left\{c, c^i\right\}, \{a\}]$ is a phone model for c both as initial and as interior phone.

The concept of center phone sets also allows us to tie parameters of models with different center phones. The construction could for example start with a single phone model shared among all phones and allow for splits using the center phone identity. This construction has not been evaluated in this work though.

Other phone features can be used in a similar way and in combination. As the number of such features is increased, the $\mathcal{C}$ transducer will grow very large in the conventional constructions, while our approach will control well for the number of states. However in this work we only did experiments with word boundary information.

4.5. Counting $\mathcal{C} \circ \mathcal{L}$ states

The impact of the size and structure of $\mathcal{C}$ on the final decoder graph also depends on the structure of the lexicon transducer $\mathcal{L}$. The frequency of n-phones and their positions in the $\mathcal{L}$ transducer change the number of states introduced by the composition with $\mathcal{C}$. For example, splitting an n-phone across words introduces more states in the composed transducer $\mathcal{C} \circ \mathcal{L}$ than a model occurring only within words. It might therefore be reasonable to incorporate the number of states in $\mathcal{C} \circ \mathcal{L}$ in the optimization procedure.

The state counting algorithm examines states in $\mathcal{C} \circ \mathcal{L}$ which are tuples (q_C, q_L) of states from $\mathcal{C}$ and $\mathcal{L}$ respectively. For each hypothesized split, first we perform the computation of required states in $\mathcal{C}$ as described in Section 4.3. Then, for each of the split states, we calculate how many states would be introduced in $\mathcal{C} \circ \mathcal{L}$. For a state q_C in $\mathcal{C}$ split into q'_C and q''_C, the set of involved composed states $q = (q_C, q_L)$ in $\mathcal{C} \circ \mathcal{L}$ is determined. If both new states $q' = (q'_C, q_L)$ and $q'' = (q''_C, q_L)$ are reachable in $\mathcal{C} \circ \mathcal{L}$, the state count is incremented. A state $q' = (q'_C, q_L)$ is reachable if a transition e exists with $p[e] = (p_C, p_L)$, $n[e] = q$ and the transition from p_C to q'_C is valid in $\mathcal{C}$ (new transitions in $\mathcal{C}$ are not generated during state counting).

The state counting on $\mathcal{C}$ can exploit that a state is defined by a unique tuple of history sets, whereas in $\mathcal{C} \circ \mathcal{L}$ the same tuple of history sets can occur for several states. The program has to consider that the examined states may be "virtual" states, that is already split states required for the same hypothesized model split. We need to integrate the split states with the actual transducer temporarily, in order to keep track of the predecessor state relationship. Furthermore, as described in Section 4.3, a state in $\mathcal{C}$ can be split several times for the same model split. The virtual states are also necessary to retrace these splits.

5. Construction of $\mathcal{C} \circ \mathcal{L}$

For the construction of the final decoder graph, the $\mathcal{C}$ transducer is composed either with the lexicon transducer $\mathcal{L}$ or with the composition of $\mathcal{L}$ with the language model transducer $\mathcal{G}$. In both cases, $\mathcal{C}$ is solely used to expand sequences of

CI phones to sequences of CD models. Therefore, instead of constructing $\mathcal{C}$ and applying composition, we can rather construct the expanded transducer directly.

Even with the size-controlled construction described in the previous section, the number of arcs in the $\mathcal{C}$ transducer can get large, especially if wider context and complex features are used. A subset of the arcs and models in $\mathcal{C}$ accounts for contexts never encountered in any phone or word sequence of the used lexicon. By not explicitly constructing $\mathcal{C}$, this problem can be eluded.

In this section, we describe the direct construction of $\mathcal{C} \circ \mathcal{L}$ by iteratively changing the structure of $\mathcal{L}$ such that it complies with the context-dependency of the constructed phone models. The construction of $\mathcal{C} \circ \mathcal{L} \circ \mathcal{G}$ can be performed analogously. We assume a minimal $\mathcal{L}$ transducer, which can be obtained by applying transducer determinization and minimization (Mohri et al., 2008), disambiguation symbols, if required for determinization, are replaced by epsilon.

We start with a straight-forward approach and describe improvements later. To simplify matters, we only describe the construction for triphones in this section.

5.1. Straight forward construction

The construction is initialized by augmenting the input labels of $\mathcal{L}$, i.e. the CI phones, with the corresponding monophonic phone model. Thus, input labels are tuples (m, π) of a phone model m and a phone π. We define the right context of a state q as the set of phones occurring on its outgoing arcs:

$$H_1(q) = \{\pi : \exists e \in E[q], i[e] = (m, \pi)\}$$

The left context $H_{-1}(q)$ is defined accordingly for the incoming arcs $I[q]$.

When a new phone model $m = [C_{-1}, C_0, C_1]$ is created by splitting the right context we have to ensure for all arcs e with $i[e] = (\pi, m)$ that state $n[e]$ enforces the new contexts: $H_1(n[e]) \subseteq C_1$. And likewise for splits on the left context: $H_{-1}(p[e]) \subseteq C_{-1}$. Splits on the center phone set (cf. Section 4.4) do not require new states.

A split of a model $m = [C_{-1}, C_0, C_1]$ on the right context affects all states with an incoming transition labeled with this model:

$$N(m) = \{n[e] : e \in E, i[e] = (m, \pi)\}$$

The reversed transducer is used for splits based on the left context. New states are required if the state context set $H_{-1/1}$ is not compatible with the contexts of the new models.

If the transducer contains (input) epsilon transitions, these epsilon paths have to be considered in the state context sets. We merge the context set of a state q with all states $p \in \epsilon[q]$ reachable from q via a path labeled with epsilon: $\tilde{H}_1(q) = \bigcup_{p \in \epsilon[q]} H_1(p)$. The left context set $\tilde{H}_{-1}$ is defined accordingly on the reversed transducer.

If we split a state q with outgoing epsilon transitions on the right context, we have to split all reachable states $p \in \epsilon[q]$ as well in order to enforce the correct right context. Left context splits are handled in the same way on the reversed transducer. The state counting has to consider all epsilon reachable states, too.

Fig. 7 shows an example of a split of a $\mathcal{C} \circ \mathcal{L}$ transducer. In this example, the right context set of state 3 before splitting is $H_1(3) = \{c, d\}$ and hence the state needs to be split. In the resulting transducer, shown in Fig. 7(b), state 3 has been split into states 6 and 7. The incoming arcs of state 3 not labeled with the split model are connected to both new states, which introduces nondeterminism at state 2. In general, if a split state has incoming arcs labeled with a model other than the currently split one, this construction generates non-deterministic arcs. Non-deterministic arcs are usually undesirable in the search graph as they may introduce redundant search hypotheses.

By keeping the original state and all of its incoming arcs, except for those labeled with the split model, the transducer remains deterministic (given a deterministic transducer $\mathcal{L}$). This construction is illustrated in Fig. 7(c). The transducer size increases by 2 states instead of 1 state. The size can be decreased by subsequent splits though. Consider a split of model m using the same context as for the previous split. The split state 3 can be merged with the existing states 6 and 7 as they originated from the same state and they enforce the same right context. Therefore, the construction and state counting algorithms have to keep track of sibling states, i.e. states generated from the same state in the $\mathcal{L}$ transducer, as well as their enforced left and right context.

Splits on left context do not introduce nondeterminism, because they divide the incoming arcs of a split state into disjunct sets according to the phonetic property used for the split.

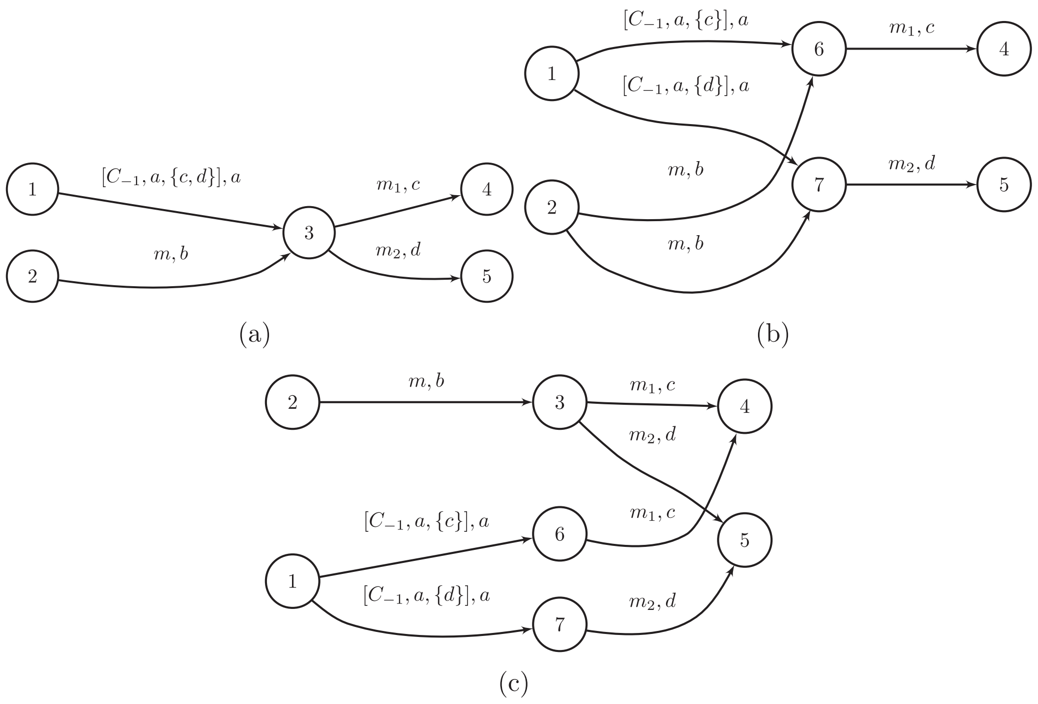

Fig. 7. Part of a $\mathcal{C} \circ \mathcal{L}$ transducer before and after splitting model $[C_{-1}, a, \{c, d\}]$ at position 1 into $\{c\}$ and $\{d\}$. (a) Before splitting, (b) after splitting and (c) after splitting ensuring deterministic arcs. m, m_i are arbitrary phone models. Output labels are omitted.

The construction described so far generates a $\mathcal{C} \circ \mathcal{L}$ transducer structured in the same way as when using a compact non-deterministic $\mathcal{C}$ transducer (cf. Section 3.2). In a compact (minimal as automaton) non-deterministic $\mathcal{C}$ transducer for triphones, a state represents the set of valid left context phones of all models on the state's outgoing arcs, as well as the set of valid successor phones. The set of valid successor phones is required to ensure the correct right context of models on the incoming arcs of a state. In contrast, a deterministic $\mathcal{C}$ (with shifted input labels), as constructed in Section 4.1, encodes only information about predecessor phones in states, the disambiguation for right contexts is carried out solely by arcs. This structural difference affects the structure and size of $\mathcal{C} \circ \mathcal{L}$. In the composition with a non-deterministic $\mathcal{C}$, states (s_C, s_L) are generated for a state s_L depending on both left and right context. Whereas in the composition with a deterministic (shifted) $\mathcal{C}$, the number of states generated for s_L only depends on its left context. In a common minimal $\mathcal{L}$ transducer most states have a higher outdegree than indegree, with the start state being a notable exception. Therefore, the composition with a non-deterministic $\mathcal{C}$ has more states than with a deterministic $\mathcal{C}$.

5.2. *Shifted labels*

To construct a shifted $\mathcal{C} \circ \mathcal{L}$ transducer, $\mathcal{L}$ is composed with a monophonic $\mathcal{C}$ transducer (cf. Section 4.1) with modified input labels. The transducer has for every pair of phones π and σ a transition $(\pi, (m_\pi, \sigma), \sigma, \sigma)$, which adds shifted monophonic models m_π to the input labels. The composition splits states having incoming arcs with different input (phone) labels, in particular the initial state. Afterwards all incoming arcs of a state have the same input label (or are members of the same center context set). Note this is the smallest $\mathcal{C} \circ \mathcal{L}$ obtainable with the $\mathcal{C}$ construction described in Section 4.1, as it is equivalent (except for the input labels) to the composition of $\mathcal{L}$ with the initial, not yet split $\mathcal{C}$ transducer.

A split of a model m based on the right context does not require new states, just a relabeling of the affected transitions $E(m)$. A split based on the left context introduces new states if the new models' contexts require distinguishing the paths reaching an affected state $q \in Q(m)$. An arc e with input label $i[e] = ([C_{-1}, C_0, C_1], \sigma)$ and $p[e] = q$ requires that

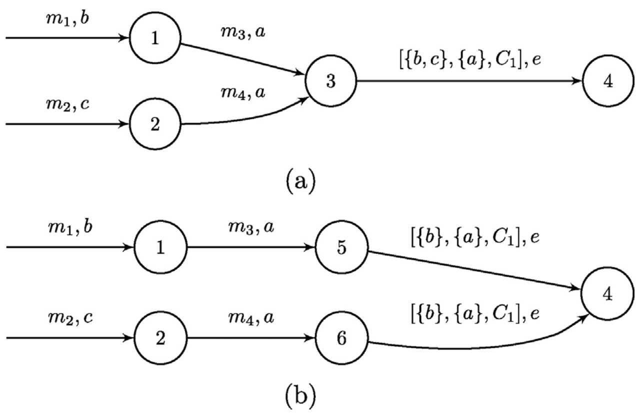

Fig. 8. Part of a $\mathcal{C} \circ \mathcal{L}$ transducer with shifted labels before and after splitting model $[\{b, c\}, a, C_1]$ (with $e \in C_1$) at position −1 into $\{b\}$ and $\{c\}$. (a) Before splitting, (b) after splitting. m_i are arbitrary phone models. Output labels are omitted.

all predecessor states $p \in P(q)$ with $P(q) = \{p[e] : e \in I[q]\}$ have a matching left context $H_{-1}(p) \subseteq C_{-1}$. Therefore, a new state is required if the intersection of the new context sets with the predecessor left context $H_{-2}(q)$ with

$$H_{-2}(q) = \bigcup_{p \in P(q)} H_{-1}(p)$$

are both not empty. In addition, states $p \in P(q)$ have to be split if $H_{-1}(p)$ is not compatible with the new contexts, for example for left context splits based on a word boundary property. For center phone set splits, states $q \in Q(m)$ are split according to their left context $H_{-1}(q)$. Epsilon transitions are handled as described in Section 5.1 by incorporating the epsilon closure of affected states.

Fig. 8 shows an example of a split based on the left context. Here, state 3 needs to be split, because $H_{-2}(3) = \{b, c\}$ which is not compatible with the new models. States 1 and 2 do not need to be split as their left contexts conforms with the new model contexts.

With the algorithm supporting epsilon transitions it is also possible to construct the context dependent models directly on the composition of $\mathcal{L}$ with the grammar transducer $\mathcal{G}$. However, this transducer is usually quite large in practice and the split optimization might exceed computation time requirements. Furthermore, constructing $\mathcal{C} \circ \mathcal{L}$ directly would allow us to efficiently integrate syllable or word identity as features in the model splitting by introducing new labels, as proposed in Liao et al. (2010), without having to build and to apply a complex $\mathcal{C}$ transducer.

6. Implementation

The iterative optimization described in Section 3.3 computes in each iteration the best split $\hat{t}$ according to the score $L(t)$:

$$\hat{t} = \operatorname*{argmax}_t L(t) = \operatorname*{argmax}_t \{G(t) - \alpha \cdot S(t)\}$$

where t defines an HMM state model, a context position, and a phonetic property. After $\hat{t}$ is computed, the split is applied to the set of models and to the transducer.

The computation of $S(t)$ is relatively costly, especially for larger contexts. It does not need to be computed for all split hypotheses t though. If the acoustic gain $G(t)$ is already lower than the anticipated best score, the incorporation of $S(t)$ can only decrease the score further and the computation for t can be waived. We can exploit this property by sorting the split hypotheses by decreasing acoustic gain. The incorporation of $S(t)$ is then a re-ranking of the hypotheses and the optimization loop can be stopped when a t with $L(t)$ lower than the current best score is discovered. For large α however, in particular if

$$L(\operatorname*{argmax}_t G(t)) \ll \max_t G(t) \ ,$$

the optimization still needs to consider nearly all hypotheses. A drawback of this method is the high memory usage required for generating and storing all split hypotheses.

If construction time is critical, the runtime can be decreased by limiting the considered hypotheses to the n best hypotheses (according to their acoustic gain) yielding an approximative solution. Furthermore, the computations of $S(t)$ for different t are independent of each other and can be performed in parallel. Similarly, the computation of acoustic likelihood scores can be parallelized.

An implementation of the described context-dependency transducer construction and all variants can be found in the publicly available tool *TrainC* [2] which is based on OpenFst (Allauzen et al., 2007).

7. Experimental results

The proposed transducer and model construction method was evaluated using an LVCSR system on an in-house English spoken query task. The acoustic models and the recognition system are described in the following sections, followed by a presentation and discussion of the experiments.

7.1. Acoustic modeling

All experiments use baseline acoustic models trained on 2100h of spoken queries. The acoustic models are retrained for each $\mathcal{C}$ transducer using a bootstrap model. The acoustic front end consists of Perceptual Linear Prediction (PLP) cepstra. An LDA transform projects 9 consecutive 13-dimensional features to a 39-dimensional feature vector. The tied HMM state models consist of up to 128 Gaussian densities per mixture model with semi-tied covariances. The total number of densities in the acoustic model ranges between 400K and 500K depending on the parameters of the model construction. The pronunciation dictionary comprises 43 phones including pseudo phones for silence and noise. Further training steps for speaker normalization and discriminative training were omitted in favor of more experiments.

7.2. Recognition system

The decoder graph for the recognition system is constructed from $\mathcal{C}$, lexicon $\mathcal{L}$, and language model $\mathcal{G}$ as described in Allauzen et al. (2009). i.e. the individual transducers are compiled as $(\mathcal{C} \circ \det(L)) \circ \mathcal{G}$ using special composition filters for label and weight pushing. All experiments use a simple one-pass decoding strategy with a backoff 3-gram language model containing 14M n-gram for a vocabulary of 1M words. The test set contains 14.6K utterances with about 46K words in total.

7.3. Experiments

We evaluated decision tree-based $\mathcal{C}$ transducers and those constructed using the method proposed. The HMM state models were constrained to cover at least 20K observations. The algorithm was terminated when 7K models were generated, thereby we obtain comparable acoustic models among all experiments. For 5-phones we used only 25% of the training data and constrained the models to cover at least 5K observations, due to memory limitations.

Table 1 shows the results grouped by n-phone order. For each n-phone order, the first row shows the results for the conventional $\mathcal{C}$ construction (Mohri et al., 2008). In each case we show the number of states in the $\mathcal{C}$ transducer (p^{n-1}) and for the triphone case we show recognition results as well. We then follow with the sizes and recognition results with the proposed method for various values of α. Note the number of $\mathcal{C}$ transitions is p times the number of $\mathcal{C}$ states throughout.

First, observe that the number of states in the conventional tree-based method is greater than in the proposed method even with $\alpha = 0$ since the latter builds $\mathcal{C}$ as a minimal automaton. The conventionally constructed $\mathcal{C}$ shown here is

[2] http://trainc.googlecode.com.

Table 1
Construction of n-phone models with different values for α. The table shows the number of HMMs, the number of HMM state models (distributions), the size of the $\mathcal{C}$ transducer, and the achieved word error rate. The first row for each n-phone order is for the conventional decision tree-based construction.

n	α	$\mathcal{C}$ states	WER (%)
3	–	1226	20.5
	0	1152	20.8
	10^2	1122	20.8
	10^3	1040	20.8
	10^4	876	20.7
	10^5	739	21.1
4	–	79,507	–
	0	13,097	21.1
	10^2	6381	21.1
	10^3	2439	21.0
	10^5	376	21.1
5	–	3.4M	–
	10^3	2934	21.0

Table 2
Results for models with incorporated word boundary information.

n	α	$\mathcal{C}$ states	WER (%)
3	0	13,744	20.1
	10^2	9376	20.4
	10^3	2821	20.2
	10^4	1087	20.3
	10^5	268	20.7
4	10^5	221	21.0

not minimal.[3] Both methods give roughly the same recognition accuracy when $\alpha = 0$ (due to software and memory limitations with our conventional tree-based system, we actually built $\mathcal{C}$ transducers only for triphones in that case). Second observe that there are values of $\alpha > 0$ that cause no reduction in word-error rate but give a substantial reduction in the number of states in $\mathcal{C}$ (e.g. $3\times$ reduction in size for 4-gram with $\alpha = 1000$). There are larger values of α that give even greater reductions in the size of $\mathcal{C}$ with only a small impact on word error rate.

The performance of the tree-based system is slightly better as it generates trees with more than 7K models and applies subsequent pruning of leafs (Breiman et al., 1984).[4] In previous experiments (Rybach and Riley, 2010), where leaf pruning was not used for the decision trees, the accuracy of both systems was the same. The subsequent merging of models within the proposed construction framework should be investigated in future work.

Using larger contexts did not yield recognition accuracy improvements for the spoken query task (see Table 1). However, phone models with larger context have improved recognition accuracy for other tasks and this new compact construction should apply similarly to other domains and languages (Chen et al., 2006).

Table 2 shows the results obtained by incorporating word boundary information. Using this additional information does improve the acoustic models. Without controlling the size of $\mathcal{C}$, the number of states increases significantly compared to the transducer without word boundary information. By using $\alpha > 0$ the size of $\mathcal{C}$ can be reduced by 75% with only a slight increase in word error rate.

[3] The size of $\mathcal{C}$ could be reduced by applying automaton minimization. However, with increasing context size or with the incorporation of word boundary information, the size of the intermediate transducer becomes unwieldy. Schuster and Hori (2005a) describe a direct construction of a minimal $\mathcal{C}$ viewed as an automaton. However this $\mathcal{C}$ is deterministic with respect to the input not the output. The latter is efficient for classical transducer composition whereas the former is not (Mohri et al., 2008).

[4] The mismatch w.r.t. leaf pruning was discovered after finishing the series of experiments, which is the reason for this slight inconsistency.

Table 3
Results achieved when optimizing the number of states in $\mathcal{C} \circ \mathcal{L}$, results obtained by optimizing the number of states in $\mathcal{C}$ are shown for comparison.

n	Count	α	$\mathcal{C}$ states	$\mathcal{C} \circ \mathcal{L}$		WER (%)
				States	Arcs	
3	$\mathcal{C} \circ \mathcal{L}$	0	1152	645.4K	2.38M	20.8
		1	1148	645.3K	2.38M	20.8
		10	1100	643.8K	2.36M	20.7
		100	962	639.1K	2.28M	20.8
	$\mathcal{C}$	10,000	876	637.6K	2.24M	20.7
4	$\mathcal{C} \circ \mathcal{L}$	0	13,097	780.2K	9.11M	21.1
		1	11,172	763.6K	7.93M	21.0
		10	5879	712.5K	5.03M	21.0
	$\mathcal{C}$	100	6,381	725,0K	5.37M	21.1

Results obtained by counting the states in $\mathcal{C} \circ \mathcal{L}$, as described in Section 4.5, are shown in Table 3. The $\mathcal{L}$ transducer used has 486.3K states and 1.64M arcs. For $n = 4$ we had to constrain the search space to 10K split hypotheses in order complete the experiments in reasonable time. As can be seen from the results, the optimization based on states in $\mathcal{C} \circ \mathcal{L}$ does not reduce the size of the composed transducer significantly compared to the optimization on $\mathcal{C}$. For triphonic models, the impact of the size of $\mathcal{C}$ on $\mathcal{C} \circ \mathcal{L}$ is very low. Even for 4-phones, the effect on $\mathcal{C} \circ \mathcal{L}$ is not substantial.

8. Discussion

The key ideas in this paper are that (a) you can build the context-dependency transducer directly from data without bothering to build an explicit decision tree in a separate step and (b) that in doing so it is easy to incorporate a regularization term that controls the size of the transducer in the greedy optimization without affecting accuracy in any significant way. Others have struggled with larger n-gram orders and generalized features precisely because the standard decision tree construction does not afford this direct size control and makes unfortunate splits that substantially increase the transducer size.

The presented construction method allows us to substantially reduce the size of $\mathcal{C}$ and $\mathcal{C} \circ \mathcal{L}$, especially for models other than triphonic. We expect that a small $\mathcal{C} \circ \mathcal{L}$ transducer is beneficial for the dynamic composition of $\mathcal{C} \circ \mathcal{L}$ with $\mathcal{G}$ during decoding (Allauzen et al., 2009). Not only the memory consumption may be reduced in this case, but potentially also the number of active hypotheses, because a lower number of states induces more path recombinations. The impact of the size of $\mathcal{C} \circ \mathcal{L}$ on the runtime performance of decoders using dynamic composition is not within the scope of this paper though.

If we were to try to use several generalized features, e.g. word boundary, syllable boundary, and gender, together in one model the number of states in a conventional construction would likely be very unwieldy. In our construction, the number of states would be well-controlled. However, the number of phone labels and hence $\mathcal{C}$ transitions would grow given how we have chosen to encode the features. This raises the possibility of changing the objective function to count the number of transitions in $\mathcal{C}$ instead and to create new phone labels only as needed. The direct construction of $\mathcal{C} \circ \mathcal{L}$, as described in Section 5, avoids dealing with a huge number of transitions in $\mathcal{C}$ by not building the transducer explicitly. Instead, only those transitions required for contexts actually existent in $\mathcal{L}$ need to be computed and are directly applied to the expanded lexicon transducer. The advantages of both the direct construction and the incorporation of the size of $\mathcal{C} \circ \mathcal{L}$ into the objective function could not be fully exploited with the word boundary feature used in this work though.

In this work we considered only a single phone right context. To handle an r-phone right context, we could initialize the optimization with the appropriate p^r state transducer with a fixed r-phone shift between input and output labels. However, this would make obvious that our construction builds a minimal automaton (i.e. the minimal transducer among all equivalent deterministic transducers with the same alignment between input and output labels) and not a minimal transducer (among all deterministic transducers irrespective of that alignment) (Mohri, 2000). It would be interesting to explore the possibility of building the true minimal transducer at each step in the iteration.

References

Allauzen, C., Riley, M., Schalkwyk, J., 2009. A generalized composition algorithm for weighted finite-state transducers. In: Proc. Conf. of the Int. Speech Communication Association (Interspeech), Brighton, UK, pp. 1203–1206.

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., Mohri, M., 2007. OpenFst: a general and efficient weighted finite-state transducer library. In: Proc. Int. Conf. on Implementation and Application of Automata, Prague, Czech Republic, pp. 11–23.

Bahl, L., de Souza, P., Gopalakrishnan, P., Nahamoo, D., Picheny, M., 1991. Context dependent modeling of phones in continuous speech using decision trees. In: Proc. DARPA Speech and Natural Language Workshop, Pacific Grove, CA, USA, pp. 264–269.

Beulen, K., Bransch, E., Ney, H., 1997. State-tying for context dependent phoneme models. In: Proc. European Conf. on Speech Communication and Technology (Eurospeech), Rhodes, Greece, pp. 1179–1182.

Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A., 1984. Classification and Regression Trees. Chapman & Hall CRC Press, Boca Raton, FL, USA.

Chen, S., Kingsbury, B., Mangu, L., Povey, D., Saon, G., Soltau, H., Zweig, G., 2006. Advances in speech transcription at IBM under the DARPA EARS program. IEEE Transactions on Audio Speech and Language Processing 14, 1596–1608.

Chen, S.F., 2003. Compiling large-context phonetic decision trees into finite-state transducers. In: Proc. European Conf. on Speech Communication and Technology (Eurospeech), Geneva, Switzerland, pp. 1169–1172.

Liao, H., Alberti, C., Bacchiani, M., Siohan, O., 2010. Revisiting decision tree state clustering: word and syllable features. In: Proc. Conf. of the Int. Speech Communication Association (Interspeech), Makuhari, Japan, pp. 2958–2961.

Mohri, M., 2000. Minimization algorithms for sequential transducers. Theoretical Computer Science 234, 177–201.

Mohri, M., Riley, M., 1999. Network optimizations for large-vocabulary speech recognition. Speech Communication 28, 1–12.

Mohri, M., Pereira, F., Riley, M., 2008. Speech recognition with weighted finite-state transducers. In: Benesty, J., Sondhi, M., Huang, Y. (Eds.), Handbook of Speech Processing. Springer, Berlin, Germany, pp. 559–582, Chapter 28.

Riley, M., Pereira, F., Mohri, M., 1997. Transducer composition for context-dependent network expansion. In: Proc. European Conf. on Speech Communication and Technology (Eurospeech), Rhodes, Greece, pp. 1427–1430.

Rybach, D., Riley, M., 2010. Direct construction of compact context-dependency transducers from data. In: Proc. Conf. of the Int. Speech Communication Association (Interspeech), Makuhari, Japan, pp. 218–221.

Schuster, M., Hori, T., 2005a. Construction of weighted finite state transducers for very wide context-dependent acoustic models. In: Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), San Juan, Puerto Rico, pp. 162–167.

Schuster, M., Hori, T., 2005b. Efficient generation of high-order context-dependent weighted finite state transducers for speech recognition. In: Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Philadelphia, PA, USA, pp. 201–204.

Sproat, R., Riley, M., 1996. Compilation of weighted finite-state transducers from decision trees. In: Proc. Annual Meeting of the Association for Computational Linguistics, Santa Cruz, CA, USA, pp. 215–222.

Stoimenov, E., McDonough, J., 2006. Modeling polyphone context with weighted finite-state transducers. In: Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Toulouse, France.

Young, S., Odell, J., Woodland, P., 1994. Tree-based state tying for high accuracy acoustic modelling. In: Proc. ARPA Spoken Language Technology Workshop, Plainsboro, NJ, USA, pp. 405–410.

Yvon, F., Zweig, G., Saon, G., 2004. Arc minimization in finite state decoding graphs with cross-word acoustic context. Computer Speech and Language 18, 397–415.