Direct Construction of Compact Context-Dependency Transducers From Data

David Rybach[*]

Human Language Technology
and Pattern Recognition
Computer Science Department
RWTH Aachen University, Germany
rybach@cs.rwth-aachen.de

Michael Riley

Google Inc.
76 Ninth Avenue
New York, NY, USA
riley@google.com

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.

Index Terms: WFST, LVCSR

1. Introduction

Weighted finite state transducers (WFST) are widely used in speech recognition applications [1]. 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 G, L is a phone to word transducer derived from the pronunciation dictionary, and C encodes the context dependency of the acoustic models. These transducers are combined by the finite-state operation of composition as $C \circ L \circ G$ to form a very efficient recognition transducer.

This paper focuses on the construction of the context dependency transducer C. C rewrites context independent *(CI)* phone sequences to sequences of context dependent (*CD*) phone models. Because of the composition with 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 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 and p^3 transitions for triphones [1]. 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, if the training data is not sufficient to build all context dependent models [2]. The construction of C can account for these classes of equivalent contexts, yielding a more compact transducer [3]. 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 [4]. The authors of [5, 6] describe an efficient construction for high-order context-dependent models that builds a minimal C. [7] presents an on-demand transducer to represent 11-phone decision trees. An "arc minimization" technique is used in [8] 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 C transducer. In this paper a new construction method is presented that builds a compact 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.

Section 2 describes conventional phonetic decision tree methods and their relation to our proposed method. Section 3 describes the proposed, direct context-dependency transducer construction. Section 4 presents the experimental results on a large corpus of spoken queries. Section 5 discusses the methods and results.

2. Decision-Tree Construction

The proposed method generates both the 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 for a better comparison and describe which steps are shared with our new construction.

2.1. Phonetic Decision Trees

In many state-of-the-art LVCSR systems phonetic decision trees are used to tie the parameters of CD phone models. The decision tree determines the similarity of CD phones by successively splitting sets of CD phones. 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

[*] Work was performed at Google.

phone and HMM state. The splitting is performed based on phonetic properties (or questions) of the phones in a specific context position (left and right in the triphone case). A decision tree is trained according to a maximum likelihood criterion using a greedy optimization strategy. In each iteration of the optimization, the best split is chosen among all current models (leaf nodes), phonetic properties, and context positions. The phonetic properties can be defined as sets of phones, e.g. the set of vowels. 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.

2.2. 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. 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. We omit many familiar details not related to these differences [2].

In each iteration a *split* is chosen, which defines the model to split, the phonetic property (or phonetic question), and the context position. 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 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.

3. 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 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 C construction, we begin with a simple example of part of a C transducer for three phones a, b, c as it might appear during its construction, as shown in Figure 1.

An input label on a transition in Figure 1 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 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 5).

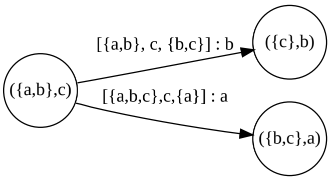

Figure 1: Example of a part of a C transducer for 3 phones. Two phone models are shown.

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

A state in Figure 1 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 C mentioned in the introduction, there would be a state for every $n - 1$ phones read. However, not all possible phone contexts have be to 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.

While the model construction deals with HMM state models, the 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. Splitting a state model therefore involves splitting all phone models sharing this state model. The splitting algorithm is applied for each of the split phone models.

3.1. Notation

A finite-state transducer $T = (\mathcal{A}, \mathcal{B}, Q, I, F, E)$ is specified by a finite input alphabet $\mathcal{A}$, a finite output alphabet $\mathcal{B}$, 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*) $E \subseteq Q \times (\mathcal{A} \cup \{\epsilon\}) \times (\mathcal{B} \cup \{\epsilon\}) \times Q$. Weights are trivial in the C transducer and hence omitted. $E[q]$ denotes the set of transitions leaving state $q \in Q$, and $I[q]$ the set of transitions to state q.

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 $E(a) = \{e : i[e] = a\}$ and the corresponding (previous) states as $Q(a) = \{p[e] : e \in E(a)\}$.

A state in C be represented by a tuple $(H_L, \ldots, H_1, \pi)$ with history sets $H_l \subseteq \mathcal{A}$, center phone $\pi \in \mathcal{A}$, and L the number of left contexts as discussed above.

3.2. Triphonic Contexts

We consider triphone models first and generalize the algorithm afterwards. Given a split of a phone model m to models m_1, m_2 at position c, the first step is to identify the affected states $q \in Q(m)$. To split the right context, only the outgoing transitions $e \in E[q]$ have to be relabeled:

$$i[e] \leftarrow \begin{cases} m_1 & \text{if } i[e] = m, o[e] \in C_1' \\ m_2 & \text{if } i[e] = m, o[e] \in C_1'' \\ i[e] & \text{if } i[e] \neq m \end{cases}$$

with $m_1 = [C_{-1}, \pi, C_1']$, $m_2 = [C_{-1}, \pi, C_1'']$.

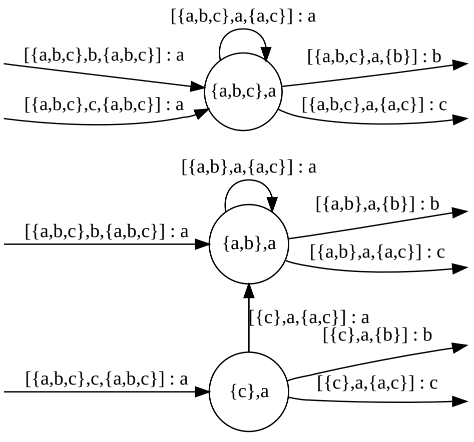

Figure 2: A state before and after splitting the model $[\{a,b,c\},a,\{a,c\}]$ at position -1 into $\{a,b\}$ and $\{c\}$.

If the split is performed on the left context with $m_1 = [C'_{-1}, \pi, C_1]$, $m_2 = [C''_{-1}, \pi, C_1]$, $q = (H, \pi)$, 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. An incoming transition $e \in I[q]$ is redirected to q_1 if $o[e] \in C'_{-1}$ or to q_2 otherwise. 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

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

Self-loop transitions require a special treatment which is omitted here, but straight-forward to implement. An example of a state split is shown in Figure 2.

The initial transducer has a state for each phone with $H_i = \Sigma$, and a transition $(\pi_1, [\Sigma, \pi_1, \Sigma], \pi_2, \pi_2)$ for every pair of phones π_1 and π_2 and labeled with a monophonic model.

3.3. Wider Contexts

For larger left contexts, splitting the history H_l of state $q = (H_L, \ldots, H_1, \pi)$ implies splits on the history H'_{l-1} of all predecessor states $p = (H'_L, \ldots, H'_1, \pi)$ with $\exists e \in E[p] : n[e] = q$. The splitting of predecessor states is done recursively on l down to 1. This ensures that only valid paths are included in the transducer.

3.4. Counting States

To count the number of new states required for a HMM state model split, $S(t)$ in the objective function, the affected states $Q(m)$ and their predecessors (cf. Section 3.3) have to be discovered. For each state q the required split states q_1, q_2 are computed by intersecting the state's history set with the phone property sets at the appropriate context position. If $q_i \notin Q$, the count is incremented. For splits of the right context, the number of new states is always 0.

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

3.5. 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.

A simple method to achieve word-boundary modeling is to introduce separate phones for each of these phone variants and to modify the pronunciation dictionary 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 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*, initialized with the phone variants, and allowing splits on these sets as well.

Other phone features can be used in a similar way and in combination. As number of such features is increased, the 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. Experimental Results

The proposed transducer and model construction method was evaluated using an LVCSR system on a 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.

4.1. Acoustic Modeling

All experiments use baseline acoustic models trained on 2100h of spoken queries. The acoustic models are retrained for each 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 dictionaries 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.

4.2. Recognition System

The decoder graph for the recognition system is constructed from C, lexicon L, and language model G as described in [9]. All experiments used a simple one-pass decoding strategy with a backoff 3-gram language model containing 14M n-grams for a vocabulary of 1M words. The test set contains 14.6K utterances with about 46K words in total.

4.3. Experiments

We evaluated decision tree-based C transducers and those constructed using the method proposed. The HMM state models were constrained to cover at least 20k observations. Table 1 shows the results grouped by n-phone order. For each n-phone order, the first row shows the results for the conventional C construction. In each case we show the number of states in the C transducer (p^{n-1}) and for the triphone case we show recognition results as well. We then follow with the sizes and recognitions results with the proposed method for various values of α. Note the number of C transitions is p times the number of C states throughout.

First observe that the number of states in the conventional tree-based method is less than in the proposed method even with

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 C transducer, and the achieved word error rate. The first row for each n-phone order is for the conventional decision-tree-based construction.

		acoustic model		C	
n	α	HMM	dist.	states	WER
3	-	17,066	6,623	1,849	21.2
	0	17,086	6,623	1,056	21.3
	100	16,953	6,623	1,032	21.1
	10^3	16,280	6,619	938	21.2
	10^5	6,846	6,543	722	21.4
4	-	-	-	79,507	-
	0	51,782	8,273	18,951	21.6
	100	43,890	8,257	9,803	21.6
	10^3	33,124	8,263	6,302	21.6
	10^5	9,681	8,144	5,728	21.7
5	-	-	-	3.4M	-
	10^3	22,857	7,000	1,453	

Table 2: Results for models with incorporated word boundary information.

n	α	dist.	states	WER
3	0	7,052	12.3k	20.5
	100	6,146	3.0k	20.5
	1k	7,061	2.9k	20.6

$\alpha = 0$ since the latter builds C as a minimal automaton. Both methods given the same recognition accuracy when $\alpha = 0$[1] 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 C (e.g., 3× reduction in size for 4-grams with $\alpha = 1000$). There are larger values of α that give even greater reductions in the size of C with only a small impact of word error rate.

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 [10].

Table 2 shows the results obtained by incorporating word boundary information. Using this additional information does improve the acoustic models. The size of the C transducer can be reduced by 75% without losing accuracy.

5. 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.

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 C transitions would grow given how we have chosen to encode the features. This raises the possiblity of changing the objective function to count the number of transitions in C instead and to create new phone labels only as needed.

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) [11]. It would be interesting to explore the possibility of building the true minimal transducer at each step in the iteration.

6. References

[1] M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," in *Handbook of Speech Processing*, J. Benesty, M. Sondhi, and Y. Huang, Eds. Springer, 2008, ch. 28, pp. 559–582.

[2] S. Young, J. Odell, and P. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *ARPA Spoken Language Technology Workshop*, Plainsboro, NJ, USA, Mar. 1994, pp. 405–410.

[3] M. Riley, F. Pereira, and M. Mohri, "Transducer composition for context-dependent network expansion," in *EUROSPEECH*, Rhodes, Greece, Sep. 1997, pp. 1427–1430.

[4] R. Sproat and M. Riley, "Compilation of weighted finite-state transducers from decision trees," in *ACL*, Santa Cruz, CA, USA, Jun. 1996, pp. 215–222.

[5] M. Schuster and T. Hori, "Construction of weighted finite state transducers for very wide context-dependent acoustic models," in *ASRU*, San Juan, Puerto Rico, Nov. 2005, pp. 162–167.

[6] M. Schuster and T. Hori, "Efficient generation of high-order context-dependent weighted finite state transducers for speech recognition," in *ICASSP*, Philadelphia, PA, USA, Mar. 2005, pp. 201–204.

[7] S. F. Chen, "Compiling large-context phonetic decision trees into finite-state transducers," in *EUROSPEECH*, Geneva, Switzerland, Sep. 2003, pp. 1169–1172.

[8] F. Yvon, G. Zweig, and G. Saon, "Arc minimization in finite state decoding graphs with cross-word acoustic context," *Computer Speech and Language*, vol. 18, no. 4, pp. 397–415, 2004.

[9] C. Allauzen, M. Riley, and J. Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," in *INTERSPEECH*, Brighton, U.K., Sep. 2009, pp. 1203–1206.

[10] S. Chen, B. Kingsbury, L. Mangu, D. Povey, G. Saon, H. Soltau, and G. Zweig, "Advances in speech transcription at IBM under the DARPA EARS program," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 14, no. 5, pp. 1596–1608, 2006.

[11] M. Mohri, "Minimization algorithms for sequential transducers," *Theoretical Computer Science*, vol. 234, pp. 177–201, Mar. 2000.

[1] Due to software and memory limitations with our conventional tree-based system, we actually built C transducers only for triphones in that case.