 Discriminative Training of WFST Factors with Application to Pronunciation Modeling

Preethi Jyothi\(^1\), Eric Fosler-Lussier\(^3\), Karen Livescu\(^2\)

\(^1\)Department of Computer Science and Engineering, The Ohio State University, USA
\(^2\)Toyota Technological Institute at Chicago, USA

Abstract
One of the most popular speech recognition architectures consists of multiple components (like the acoustic, pronunciation and language models) that are modeled as weighted finite state transducer (WFST) factors in a cascade. These factor WFSTs are typically trained in isolation and combined efficiently for decoding. Recent work has explored jointly estimating parameters for these models using considerable amounts of training data. We propose an alternative approach to selectively train factor WFSTs in such an architecture, while still leveraging information from the entire cascade. This technique allows us to effectively estimate parameters of a factor WFST using relatively small amounts of data, if the factor is small. Our approach involves an online training paradigm for linear models adapted for discriminatively training one or more WFSTs in a cascade. We apply this method to train a pronunciation model for recognition on conversational speech, resulting in significant improvements in recognition performance over the baseline model.

Index Terms: Pronunciation models, weighted finite state transducers, large-margin training.

1. Introduction

Weighted finite state transducers (WFSTs) provide a generic and efficient framework to represent the main components (acoustic, pronunciation and language models) of an automatic speech recognition (ASR) system [1]. These components (or factor FSTs, as we will refer to them henceforth) are composed into a decoding graph that can be efficiently searched during the ASR decoding task [2, 3].

The individual factor FSTs in this framework are typically trained in isolation. After they are composed together to obtain the decoding graph, the transition weights of this graph could be further optimized using discriminative training criteria [4, 5, 6]. Unfortunately, this graph typically runs into millions of arcs, especially for large vocabulary tasks, and hence requires large amounts of training data to effectively optimize weights on its arcs. We propose an alternative approach to selectively train a factor WFST, while still leveraging information from the entire ASR cascade. For small factor WFSTs, this technique allows us to effectively estimate parameters with limited amounts of training data.

We mainly borrow inspiration from previous work on discriminatively learning weights corresponding to the scores in a WFST by introducing a linear model on its arcs [7, 6, 8]. Our current approach builds on techniques introduced in [9] where a dynamic Bayesian network was converted to a WFST cascade and then discriminatively trained. The main difference in our approach, compared to prior work, is that we selectively learn weights on a single factor FST within a cascade.

We apply this technique to pronunciation models for recognizing conversational speech. This was done mainly for two reasons:

- Unlike acoustic and language models that are automatically learned from data, pronunciation models are typically derived from existing hand-designed dictionaries that map a word to one or more canonical pronunciations. There have been a number of previous attempts to stochastically learn weights for the pronunciations of a word [10, 11] and predict pronunciations of new words [12, 13]. There has also been recent work that introduces discriminatively trained phoneme confusion models to handle pronunciation variation [14]. Our approach is different in that we use a frame-level pronunciation model and train it in the context of other frame-level components in the ASR. This, in particular, allows our trained pronunciation model to capture information about phoneme durations. This is described in more detail in Section 3.

- Pronunciation models are typically much smaller than the acoustic and language models. This allows us to experiment with limited amounts of training data in order to discriminatively reweight the arcs of the pronunciation model.

We mainly demonstrate our technique with an isolated word recognition task on conversational speech from the Switchboard corpus [15] that shows significant performance improvements using our discriminatively trained pronunciation model. We also report preliminary results on a task using continuous word sequences from Switchboard that suggest that this approach is promising. Sections 3 and 4 elaborate on our experimental setup and results.

2. Methodology

Following notation first introduced in [1], the decoding graph in an ASR system can be written as the composition of the following components (or factor FSTs):

\[
D = H \odot C \odot L \odot G
\]  

(1)

where \(H\) represents Hidden Markov Models (HMMs) trained for context-dependent phonemes in the acoustic model, \(C\) is a context-dependency transducer mapping context-dependent phonemes to monophones, \(L\) is the pronunciation (or lexicon) model and \(G\) is the language (or grammar) model. \(D\) transduces state sequences of the HMMs to word sequences. Often, \(C\) and \(L\) are unweighted and \(H\) and \(G\) are probabilistic, with the costs on their arcs being negative log-likelihood values. We shall start from such a baseline system, and discriminatively reweight \(L\).
For a given input utterance \( X \), the decoder’s function is to search for a path of arcs in \( D \) that minimizes the cost of a sequence of words best matching the acoustic input signal (HMM state sequence on \( D \)). This can be written as:

\[
a^* = \arg\min_{a \in D} w_\alpha(X, a)
\]

(2)

where \( a \) is a path in the decoding graph \( D \) and \( w \) is a scoring function that assigns a score to such a path; \( w \) is parameterized by \( \alpha \) (which we shall train). Since \( D = H \circ L \circ G \), this scoring function is given by

\[
w_\alpha(X, a) = \min_{(a^H, a^L, a^G)} \left( w^H_{a^H}(X, a^H) + w^L_{a^L}(X, a^L) + w^G_{a^G}(X, a^G) \right)
\]

(3)

where \((a^H, a^L, a^G)\) is a tuple of paths in the factor FSTs. Such a tuple is said to be consistent with \( \alpha \) if the sequences of input and output labels of \( a \) match those of the paths in the first and last factors (i.e., \( [a] = [a^H], \text{out}[a] = [a^L] \) and the sequence of output labels of the path in each factor FST matches the input labels of the path in the next factor (i.e., \( \text{out}[a^L] = [a^G] \)) and so on. Also, \( w^H(X, a^H) \) is the score assigned to the path \( a^H \) in \( H \), and so on. Although in Equation 3 we have parameterized the weights functions of all the factor FSTs with \( \alpha \), we can choose to parameterize only those of the factors we intend to train. We will now formally describe our training algorithm and some implementation details of this algorithm using WFSTs.

### 2.1. Training Paradigm

For a training utterance \( X \) with output word sequence \( W \), we first define \( f_{X,a}(\alpha) = w_\alpha(X, a) \) where \( f \) is a scoring function that linearly combines feature values associated with arcs in the path \( a \).

1. For the linear combination, \( \alpha \), are the parameters of our model. We define this function as:

\[
f_{X,a}(\alpha) = \alpha \cdot \phi(X, a) + \psi(X, a) \tag{4}
\]

where \( a \) is a path in the decoding graph \( D \), \( X \) corresponds to the acoustic features for the given utterance, \( \phi(X, a) \) refers to feature values on arcs of the factor FSTs that are trained and \( \psi(X, a) \) refers to arc weights in the factor FSTs that are not trained.

We also define the following function:

\[
g_{X,W}(\alpha) = \min_{a, \text{st.out}[a]=\text{st.out}[W]} f_{X,a}(\alpha) \tag{5}
\]

\( g_{X,W}(\alpha) \) corresponds to the best path in \( D \) for \( X \) such that the output sequence of \( a \) is the same as the reference word sequence \( W \). We note here that \( g_{X,W}(\alpha) \) is a concave function because it is a min over a set of affine functions \( (f_{X,a}(\alpha)) \) in \( \alpha \). We also define \( \mathcal{F}_{X,W}(\alpha) \):

\[
\mathcal{F}_{X,W}(\alpha) = \min_{a, \text{st.out}[a]=\text{st.out}[W]} f_{X,a}(\alpha) \tag{6}
\]

which corresponds to the best path in the FST for \( X \), with an output sequence other than \( W \).

As an optimization problem, we would like to minimize the expected zero-one loss over the training set:

\[
\alpha^* = \arg\min_{\alpha} \mathbb{E}_{(X,W)} \left[ 1_{W \neq \arg\min_{W'}} g_{X,W'}(\alpha) \right] \tag{7}
\]

In this section, in order to represent the final objective function as a difference of concave functions, it helps to define \( f \) and \( g \) as functions of \( \alpha \).

### Algorithm 1 Algorithm using CCCP to optimize (9)

**Require:** Training samples \( T = \{X_i, W_i\}_{i=1}^N \), factor FSTs \( H, L, G \), learning rate \( \lambda \), number of epochs \( E \).

1. Initialize \( \alpha_0 \)
2. \( D := H \circ L \circ G \)
3. for \( e := 1 \) to \( E \) do
   4. \( \beta_0 := \alpha_{e-1} \)
   5. for \( i := 1 \) to \( N \) do
      6. \( \alpha_i = \arg\min_{a, \text{st.out}[a]=W_i} w_{\beta_{i-1}}(X_i, a) \)
      7. end for
   8. for \( i := 1 \) to \( N \) do
      9. \( \alpha_i = \arg\min_{a, \text{st.out}[a]=W_i} w_{\beta_{i-1}}(X_i, a) \)
      10. \( \Delta \phi := \phi(X_i, \alpha_i) - \phi(X_i, \alpha_i') \)
      11. \( \Delta \psi := \psi(X_i, \alpha_i) - \psi(X_i, \alpha_i') \)
      12. \( \eta := \min \left\{ \frac{1}{\eta}, \text{hinge-loss}(\beta_{i-1}((\Delta \phi, \Delta \psi))) \right\} \)
      13. \( \beta_i := \beta_{i-1} + \eta \Delta \phi \)
      14. end for
   15. \( \alpha_e := \frac{1}{N} \sum_{i=1}^N \beta_i \)
   16. end for

Instead, we replace the zero-one loss function with a smooth structural hinge-loss function and the expectation is replaced with the empirical average over the training data (\( N \) instances) to give an upper bound to the function in (7):

\[
\alpha^* = \arg\min_{\alpha} \frac{1}{N} \sum_{i=1}^N \text{hinge-loss}(	ext{margin}_{i}(\alpha)) \tag{8}
\]

where \( \text{margin}_{i}(\alpha) = \mathcal{F}_{X_i,W_i}(\alpha) - g_{X_i,W_i}(\alpha) \) and hinge-loss(\( \alpha \)) = \( \max(0, 1 - \alpha) \). Henceforth, we shall refer to \( g_{X_i,W_i}(\alpha) \) and \( \mathcal{F}_{X_i,W_i}(\alpha) \) as \( g_{i}(\alpha) \) and \( \mathcal{F}_{i}(\alpha) \), respectively. This objective function in Equation (8) is neither a concave nor a convex function; it can be expressed as a difference of two concave functions (or equivalently, difference of two convex functions) as follows:

\[
\alpha^* = \arg\min_{\alpha} \frac{1}{N} \left( \sum_{i=1}^N g_{i}(\alpha) - \sum_{i=1}^N \min\{g_{i}(\alpha), \mathcal{F}_{i}(\alpha) - 1\} \right) \tag{9}
\]

using the hinge-loss function that can be rewritten as:

\[
\text{hinge-loss}(	ext{margin}_{i}(\alpha)) = g_{i}(\alpha) - \min\{g_{i}(\alpha), \mathcal{F}_{i}(\alpha) - 1\}.
\]

Since the objective a difference of convex functions, we can use the concave-convex procedure (CCCP) [16] to solve for \( \alpha^* \) in Equation (9). The main idea in this optimization procedure is to linearize the concave part of the function (i.e. \( \sum_{i} g_{i}(\alpha) \) in (9)) around the solution \( \alpha \) obtained in the current iteration so that the objective function is now convex in \( \alpha \) and can be solved as a sequence of convex optimization problems (we use the online passive-aggressive algorithm [17] for the convex optimization problems). Our training algorithm, adapted from [18], is formally described in Algorithm 1.

### 2.2. Implementation using WFSTs

This section describes some implementation details of Algorithm 1; we used the OpenFst libraries [19] to construct, combine and decode the WFSTs in this algorithm. We also used utilities from the Kaldi toolkit [20].
A dynamic Bayesian network was converted to a WFST cascade by introducing a linear model on its arcs [7, 6, 8]. Our training data allows us to effectively estimate parameters with limited amounts of training data, while still leveraging information from the entire factor WFSTs, as we will refer to them henceforth.

In step 6 of Algorithm 1, for a given training utterance \( X_i \), we need to compute the best path of arcs corresponding to its word sequence \( W_i \). In order to find this path, we first compose \( D \) with a finite state acceptor that takes as input the word labels in \( W_i \). Then, we search for a path in this decoding graph constrained by \( W_i \). The decoding step in step 9 of Algorithm 1 finds the best path in \( D \) other than the correct word sequence \( W_i \). For our isolated word task, we composed \( D \) with an acceptor that accepted every word other than the correct word.

In order to compute \( \phi \), we need to decompose a path \( a \) in \( D \) into a set of paths \( (a_H, a_C, a_L, a_G) \). Finding this best set of paths is an optimization problem that is modeled as a shortest path problem in an auxiliary network (as was previously done in our prior work [9]). This is illustrated in Figure 1. Once these paths are computed, we build the feature function \( \Delta \phi \) by only considering arcs in the path \( a^c \) from \( H \) and \( a_i \).

3. Experiments: Isolated Word Recognition

In order to evaluate the utility of discriminative lexical modeling approaches, it is useful to see how these models can improve word discriminability independent of a grammar. We validate our approach using an isolated word recognition task for conversational speech, where we may expect to see significant pronunciation variation.

3.1. Experimental Design

Our experiments were conducted on a subset of the Switchboard conversational speech corpus (called the Switchboard Transcription Project, STP) that is phonetically labeled at a fine-grained level [21]. Words are excised from continuous utterances in STP and are treated as isolated for this task. There are 3328 words in the dictionary for this model, with a total of 5560 pronunciations. We use 2942 words from the subsets 24–29 as training data to discriminatively train the pronunciation model; 165 words from set 20 and 236 words from sets 21–22 were used as the development set and evaluation set, respectively. The dataset is described in more detail in [22]. We will refer to this task as the “STP task.”

The baseline recognizer for this isolated word dataset is a standard, speaker-independent GMM-HMM speech recognizer that was built using the Kaldi toolkit [20]. We use an acoustic model that is trained on all of Switchboard-I (i.e., around 320 hours of data) [15]. We exclude the STP sentences that were used to construct the isolated word datasets from this training data. The acoustic features are Mel-frequency cepstral coefficients (MFCCs) with delta and double delta coefficients; linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT) feature-space transformations were applied for feature-space reduction. These acoustic features serve as the observations to model tied-state triphones using mixtures of Gaussians (with a maximum of 100000 Gaussians in total over all the tied-state triphones). Since this is an isolated word task, the language model is a simple acceptor for all the 3328 words in the vocabulary. The isolated nature of the task makes it harder than standard ASR since the acoustic-lexical models cannot rely on the prior of a strong language model.

Our approach adapts the weights of the lexicon transducer \( L \). Since the WSFT updates are general, we can also update weights on the complete graph \( (H \circ C \circ L) \), or leave the lexicon out of the discriminative training (upating \( H \circ C \)).

3.2. Results and Discussion

The only difference between the baseline recognizer and the systems used in our experiments is the pronunciation model. We start with an untrained pronunciation model used in the baseline recognizer and discriminatively train an indicator feature (that takes the value of 1 when the arc is traversed and 0 otherwise) on the arcs of this model. \( \alpha \) is initialized (step 1 in Algorithm 1) such that the starting point before discriminative training coincides with the baseline system. We note here that we used a frame-level pronunciation model that is trained in the context of other frame-level ASR components. Typically, the pronunciation model is at the phone level. This means that for every phone recognized, \( H \circ C \) outputs a single (non-epsilon) symbol. However, we observe that using a frame-level pronunciation model allows us to capture phoneme duration information and performs better empirically when compared to a phone-level pronunciation model for our limited data tasks.

Table 1 shows error rates on the development set and test set for the STP task. We obtain our best results by using a discriminatively trained \( L \) model. Our proposed approach reduces error rate on the test set by an absolute 9% when compared to the baseline model. This difference is statistically significant at

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev ER (%)</th>
<th>Test ER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline system</td>
<td>46.1</td>
<td>50.4</td>
</tr>
<tr>
<td>System with a discriminatively trained ( L )</td>
<td>38.8</td>
<td>41.1</td>
</tr>
<tr>
<td>System with a discriminatively trained ( H \circ C )</td>
<td>44.8</td>
<td>46.2</td>
</tr>
<tr>
<td>System with a discriminatively trained ( H \circ C \circ L )</td>
<td>41.2</td>
<td>41.5</td>
</tr>
</tbody>
</table>

Table 1: Error rates for the STP task.
Based on the data split from the WS96 JHU workshop, we have 68719 sentences as training utterances, 1344 sentences as the development set and 409 sentences as the evaluation set. As in the STP task, the acoustic models are Gaussian mixture models for tied-state triphones using MFCC delta + double-delta features with LDA, MLLT feature-space transformations. The pronunciation model (with one pronunciation per word) has 30247 features with LDA, MLLT feature-space transformations. The pronunciation model is trained using tied-state triphones using MFCC delta + double-delta features with LDA, MLLT feature-space transformations.

In order to facilitate rapid-turnaround experiments that integrate the discriminative lexical model into a full recognition process, we conduct pilot studies utilizing a limited data setting: we discriminatively train our pronunciation model with 1/8th of the training data. We also remove “excess utterances,” where the same transcription has been seen 20 times already; this primarily removes single-word utterances.

For efficiency purposes, we generate decoding lattices using the baseline system and its trigram language model; these lattices are used to constrain our word hypotheses during discriminative training. The discriminative training requires multiple decoding passes; we use a unigram language model during these decodes to allow for more confusable words, similar to strategies used for discriminative acoustic modeling [24, 25].

4.2. Results

Table 3 shows preliminary results after 4 epochs of discriminatively training the frame-level pronunciation model. The pilot system improves word recognition for the development and evaluation sets, albeit by a margin that is not statistically significant ($p > 0.05$). We find the performance improvement encouraging: the pilot study suggests that discriminative lexical modeling may be helpful beyond the isolated word domain (where we saw significant success). We are investigating this setting further, including enlarging the training set beyond 1/8th of the data, to further improve recognition for continuous ASR.

5. Conclusions

This paper proposes a general approach to selectively train a factor FST within an ASR cascade using large-margin training. This technique allows us to use limited amounts of training data to effectively estimate parameters for small factor FSTs. We demonstrate our approach by training a phone-based pronunciation model for recognition on conversational speech. We show significant improvements in recognition performance (absolute 9% improvement) on an isolated word task using our trained pronunciation model. We also present preliminary results on a continuous word recognition task that suggest that this paradigm will prove useful in that setting as well.

For future work, we intend to explore this approach further by using more complex pronunciation models that model words as overlapping sequences of articulatory features [9]. Also, we only report preliminary results on the continuous word task and constrain our approach to use word lattices from the baseline recognizer. In order to use the entire decoding graph, a more streamlined decoding procedure, possibly along the lines of those in [26, 27], needs to be developed.

6. Acknowledgements

The authors thank Rohit Prabhavalkar and Joseph Keshet for introducing them to the idea of CCCP. This research was supported by the NSF grants IIS-0905633 and IIS-09005420. The opinions expressed in this work are those of the authors and do not necessarily reflect the views of the funding agency.
7. References


