# Discriminative Training of WFST Factors with Application to Pronunciation Modeling

*Preethi Jyothi*[1], *Eric Fosler-Lussier*[1], *Karen Livescu*[2]

[1]Department of Computer Science and Engineering, The Ohio State University, USA
[2]Toyota Technological Institute at Chicago, USA

jyothi,fosler@cse.ohio-state.edu, klivescu@ttic.edu

## 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):

$$\mathcal{D} = \mathcal{H} \circ \mathcal{C} \circ \mathcal{L} \circ \mathcal{G} \qquad (1)$$

where $\mathcal{H}$ represents Hidden Markov Models (HMMs) trained for context-dependent phonemes in the acoustic model, $\mathcal{C}$ is a context-dependency transducer mapping context-dependent phonemes to monophones, $\mathcal{L}$ is the pronunciation (or lexicon) model and $\mathcal{G}$ is the language (or grammar) model. $\mathcal{D}$ transduces state sequences of the HMMs to word sequences. Often, $\mathcal{C}$ and $\mathcal{L}$ are unweighted and $\mathcal{H}$ and $\mathcal{G}$ are probabilistic, with the costs on their arcs being negative log-likelihood values. We shall start from such a baseline system, and discriminatively reweight $\mathcal{L}$.

For a given input utterance $\mathbf{X}$, the decoder's function is to search for a path of arcs in $\mathcal{D}$ that minimizes the cost of a sequence of words best matching the acoustic input signal (HMM state sequence on $\mathcal{D}$). This can be written as:

$$\boldsymbol{a}^* = \operatorname*{argmin}_{\boldsymbol{a}\in\mathcal{D}} w_{\boldsymbol{\alpha}}(\mathbf{X}, \boldsymbol{a}) \tag{2}$$

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

$$w_{\boldsymbol{\alpha}}(\mathbf{X}, \boldsymbol{a}) = \min_{\substack{(\boldsymbol{a}^{\mathcal{H}}, \boldsymbol{a}^{\mathcal{C}}, \boldsymbol{a}^{\mathcal{L}}, \boldsymbol{a}^{\mathcal{G}}) \\ \text{consistent with } \boldsymbol{a}}} w^{\mathcal{H}}_{\boldsymbol{\alpha}}(\mathbf{X}, \boldsymbol{a}^{\mathcal{H}}) + w^{\mathcal{C}}_{\boldsymbol{\alpha}}(\mathbf{X}, \boldsymbol{a}^{\mathcal{C}}) + w^{\mathcal{L}}_{\boldsymbol{\alpha}}(\mathbf{X}, \boldsymbol{a}^{\mathcal{L}}) + w^{\mathcal{G}}_{\boldsymbol{\alpha}}(\mathbf{X}, \boldsymbol{a}^{\mathcal{G}}) \tag{3}$$

where $(\boldsymbol{a}_{\mathcal{H}}, \boldsymbol{a}_{\mathcal{C}}, \boldsymbol{a}_{\mathcal{L}}, \boldsymbol{a}_{\mathcal{G}})$ is a tuple of paths in the factor FSTs. Such a tuple is said to be *consistent* with $\boldsymbol{a}$ if the sequences of input and output labels of $\boldsymbol{a}$ match those of the paths in the first and last factors (i.e., $\text{in}\,[\boldsymbol{a}] = \text{in}\,[\boldsymbol{a}_{\mathcal{H}}]$, $\text{out}\,[\boldsymbol{a}] = \text{out}\,[\boldsymbol{a}_{\mathcal{G}}]$) 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., $\text{out}\,[\boldsymbol{a}_{\mathcal{H}}] = \text{in}\,[\boldsymbol{a}_{\mathcal{C}}]$ and so on). Also, $w^{\mathcal{H}}(\mathbf{X}, \boldsymbol{a}_{\mathcal{H}})$ is the score assigned to the path $\boldsymbol{a}_H$ in $\mathcal{H}$, and so on. Although in Equation 3 we have parameterized the weight functions of all the factor FSTs with $\boldsymbol{\alpha}$, 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 $\mathbf{X}$ with output word sequence $\mathbf{W}$, we first define $f_{\mathbf{X},\boldsymbol{a}}(\boldsymbol{\alpha}) = w_{\boldsymbol{\alpha}}(\mathbf{X}, \boldsymbol{a})$ where $f$ is a scoring function that linearly combines feature values associated with arcs in the path $\boldsymbol{a}$.[1] The weights for this linear combination, $\boldsymbol{\alpha}$, are the parameters of our model. We define this function as:

$$f_{\mathbf{X},\boldsymbol{a}}(\boldsymbol{\alpha}) = \boldsymbol{\alpha} \cdot \boldsymbol{\phi}(\mathbf{X}, \boldsymbol{a}) + \psi(\mathbf{X}, \boldsymbol{a}) \tag{4}$$

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

We also define the following function:

$$g_{\mathbf{X},\mathbf{W}}(\boldsymbol{\alpha}) = \min_{\boldsymbol{a},\ \text{s.t. out}[\boldsymbol{a}]=\mathbf{W}} f_{\mathbf{X},a}(\boldsymbol{\alpha}) \tag{5}$$

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

$$\overline{g}_{\mathbf{X},\mathbf{W}}(\boldsymbol{\alpha}) = \min_{\boldsymbol{a},\ \text{s.t. out}[\boldsymbol{a}]\neq\mathbf{W}} f_{\mathbf{X},a}(\boldsymbol{\alpha}) \tag{6}$$

which corresponds to the best path in the FST for $\mathbf{X}$, with an output sequence other than $\mathbf{W}$.

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

$$\boldsymbol{\alpha}^* = \operatorname*{argmin}_{\boldsymbol{\alpha}} \mathbb{E}_{(\mathbf{X},\mathbf{W})} \left[ 1_{\mathbf{W}\neq\operatorname{argmin}_{\mathbf{W}'} g_{\mathbf{X},\mathbf{W}'}(\boldsymbol{\alpha})} \right] \tag{7}$$

[1]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 $\boldsymbol{\alpha}$.

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

```
Require: Training samples $\mathcal{T} = \{\mathbf{X}_i, \mathbf{W}_i\}_{i=1}^{\mathcal{N}}$, factor FSTs $\mathcal{H}, \mathcal{C}, \mathcal{L}$ and $\mathcal{G}$, learning rate $\lambda$, number of epochs $E$.
 1: Initialize $\boldsymbol{\alpha}_0$            /*See Section 3.2*/
 2: $\mathcal{D} := \mathcal{H} \circ \mathcal{C} \circ \mathcal{L} \circ \mathcal{G}$
 3: for $e := 1$ to $E$ do
 4:     $\beta_0 := \boldsymbol{\alpha}_{e-1}$
 5:     for $i := 1$ to $\mathcal{N}$ do
 6:         $\overline{\boldsymbol{a}}_i = \operatorname*{argmin}_{\boldsymbol{a},\ \text{s.t. out}[\boldsymbol{a}]=\mathbf{W}_i} w_{\beta_0}(\mathbf{X}_i, \boldsymbol{a})$
 7:     end for
 8:     for $i := 1$ to $\mathcal{N}$ do
 9:         $\hat{\boldsymbol{a}}_i = \operatorname*{argmin}_{\boldsymbol{a},\ \text{s.t. out}[\boldsymbol{a}]\neq\mathbf{W}_i} w_{\beta_{i-1}}(\mathbf{X}_i, \boldsymbol{a})$
10:         $\Delta\boldsymbol{\phi} := \boldsymbol{\phi}(\mathbf{X}_i, \hat{\boldsymbol{a}}_i) - \boldsymbol{\phi}(\mathbf{X}_i, \overline{\boldsymbol{a}}_i)$
11:         $\Delta\psi := \psi(\mathbf{X}_i, \hat{\boldsymbol{a}}_i) - \psi(\mathbf{X}_i, \overline{\boldsymbol{a}}_i)$
12:         $\eta := \min\{\frac{1}{\lambda}, \frac{\text{hinge-loss}(\beta_{i-1}\cdot(\Delta\boldsymbol{\phi}+\Delta\psi))}{\|\Delta\boldsymbol{\phi}\|^2}\}$
13:         $\beta_i := \beta_{i-1} + \eta\Delta\boldsymbol{\phi}$
14:     end for
15:     $\boldsymbol{\alpha}_e := \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \beta_i$
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 ($\mathcal{N}$ instances) to give an upper bound to the function in (7):

$$\alpha^* = \operatorname*{argmin}_{\boldsymbol{\alpha}} \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \text{hinge-loss}(\text{margin}_i(\boldsymbol{\alpha})) \tag{8}$$

where $\text{margin}_i(\boldsymbol{\alpha}) = \overline{g}_{\mathbf{X}_i,\mathbf{W}_i}(\boldsymbol{\alpha}) - g_{\mathbf{X}_i,\mathbf{W}_i}(\boldsymbol{\alpha})$ and $\text{hinge-loss}(x) = \max(0, 1-x)$. Henceforth, we shall refer to $g_{\mathbf{X}_i,\mathbf{W}_i}(\boldsymbol{\alpha})$ and $\overline{g}_{\mathbf{X}_i,\mathbf{W}_i}(\boldsymbol{\alpha})$ as $g_i(\boldsymbol{\alpha})$ and $\overline{g}_i(\boldsymbol{\alpha})$, 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:

$$\alpha^* = \operatorname*{argmin}_{\boldsymbol{\alpha}} \frac{1}{\mathcal{N}} \left( \sum_{i=1}^{\mathcal{N}} g_i(\boldsymbol{\alpha}) - \sum_{i=1}^{\mathcal{N}} \min\{g_i(\boldsymbol{\alpha}), \overline{g}_i(\boldsymbol{\alpha}) - 1\} \right) \tag{9}$$

using the hinge-loss function that can be rewritten as: $\text{hinge-loss}(\text{margin}_i(\boldsymbol{\alpha})) = g_i(\boldsymbol{\alpha}) - \min(g_i(\boldsymbol{\alpha}), \overline{g}_i(\boldsymbol{\alpha}) - 1)$.

Since the objective a difference of convex functions, we can use the concave-convex procedure (CCCP) [16] to solve for $\boldsymbol{\alpha}^*$ in Equation (9). The main idea in this optimization procedure is to linearize the concave part of the function (i.e. $\sum_i g_i(\boldsymbol{\alpha})$ in (9)) around the solution ($\boldsymbol{\alpha}$) obtained in the current iteration so that the objective function is now convex in $\boldsymbol{\alpha}$ 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].

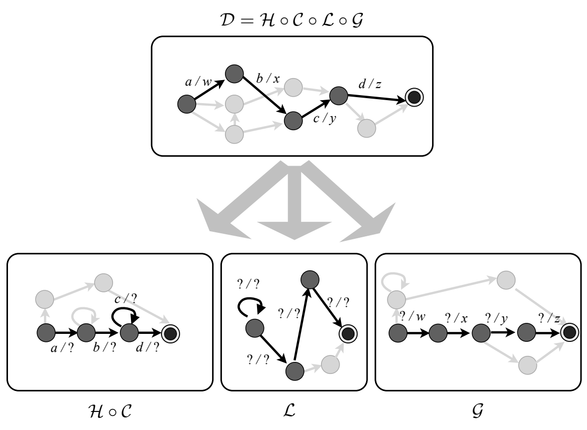
$\mathcal{D} = \mathcal{H} \circ \mathcal{C} \circ \mathcal{L} \circ \mathcal{G}$
a/w
b/x
c/y
d/z
c/?
a/?
b/?
d/?
?/?
?/w
?/x
?/y
?/z
$\mathcal{H} \circ \mathcal{C}$
$\mathcal{L}$
$\mathcal{G}$

Figure 1: Illustration of decomposition of a path in $\mathcal{D}$ into component paths in its factor FSTs. A path $\boldsymbol{a}$ of length 4 is shown in $\mathcal{D}$, with in $[\boldsymbol{a}] = abcd$ and with out $[\boldsymbol{a}] = wxyz$. This path is decomposed into paths $\boldsymbol{a}_{\mathcal{H}\circ\mathcal{C}}$, $\boldsymbol{a}_{\mathcal{L}}$ and $\boldsymbol{a}_{\mathcal{G}}$ in the factor FSTs, according to Equation (3). Given $\boldsymbol{a}$, the sequence of states in each of these paths is fixed, but the arcs (with labels indicated by "?") are solved for, subject to the constraints in $[\boldsymbol{a}_{\mathcal{H}\circ\mathcal{C}}] = abcd$, out $[\boldsymbol{a}_{\mathcal{H}\circ\mathcal{C}}] =$ in $[\boldsymbol{a}_{\mathcal{L}}]$, out $[\boldsymbol{a}_{L}] =$ in $[\boldsymbol{a}_{\mathcal{G}}]$ and out $[\boldsymbol{a}_{\mathcal{G}}] = wxyz$.

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

In order to compute $\boldsymbol{\phi}$, we need to decompose a path $\boldsymbol{a}$ in $\mathcal{D}$ into a set of paths $(\boldsymbol{a}_H, \boldsymbol{a}_C, \boldsymbol{a}_L, \boldsymbol{a}_G)$. 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 $\Delta\boldsymbol{\phi}$ by only considering arcs in the path $\boldsymbol{a}^{\mathcal{L}}$ from $\overline{\boldsymbol{a}}_i$ and $\hat{\boldsymbol{a}}_i$.

## 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[2]) 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."

[2]http://www1.icsi.berkeley.edu/Speech/stp/

| Model | Dev ER (%) | Test ER (%) |
|---|---|---|
| Baseline system | 46.1 | 50.4 |
| System with a discriminatively trained $\mathcal{L}$ | **38.8** | **41.1** |
| System with a discriminatively trained $\mathcal{H} \circ \mathcal{C}$ | 44.8 | 46.2 |
| System with a discriminatively trained $\mathcal{H} \circ \mathcal{C} \circ \mathcal{L}$ | 41.2 | 41.5 |

Table 1: Error rates for 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 $\mathcal{L}$. Since the WSFT updates are general, we can also update weights on the complete graph ($\mathcal{H} \circ \mathcal{C} \circ \mathcal{L}$), or leave the lexicon out of the discriminative training (updating $\mathcal{H} \circ \mathcal{C}$).

### 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. $\boldsymbol{\alpha}$ 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, $\mathcal{H} \circ \mathcal{C}$ 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 $\mathcal{L}$ 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

| Case | Baseline system (%) | Our system with trained $\mathcal{L}$ (%) |
|---|---|---|
| Words predicted correctly by both systems | 38.1 | 34.2 |
| All words | 49.9 | 46.8 |

Table 2: Phone alignment error rates on the evaluation set

$p < 0.001$ according to McNemar's test (we used the NIST SCTK toolkit [23] to perform this evaluation). These were tuned on the development set and the best results were obtained with $\lambda = 0.001$ and 8 training iterations. Table 1 shows that these are comparable to training weights on the entire decoding graph ($\mathcal{H} \circ \mathcal{C} \circ \mathcal{L}$). However, only training $\mathcal{H} \circ \mathcal{C}$ gives much lower improvement over the baseline, suggesting that most of the improvement that comes from discriminative training can be attributed to the pronunciation model. We suspect that this improvement is due in part to the nature of the task: there are many fewer data than one would find in a typical discriminative acoustic modeling condition. It is possible that discriminative acoustic modeling (training $\mathcal{H} \circ \mathcal{C}$) will be appropriate in larger training data situations, suggesting that discriminative lexicon training may be appropriate for low-resource conditions.

Our trained pronunciation model outperforms the baseline model beyond what the error rate improvements (Table 1) suggest. For the STP data set, we have manually transcribed utterances at the phonetic level that we can use as a reference. Since our pronunciation model is also at the frame level, we can produce phone labels for each frame; we measure phone alignment error rates computed by an edit distance between a frame-level reference phone sequence and our hypothesized phone sequence. We compute the minimum number of edits (insertions, deletions or substitutions) required to change our hypothesized phone sequence to the reference phone sequence at the frame level. Table 2 reports these phone alignment error rates on the evaluation set. The first case refers to utterances in the evaluation set that are correctly predicted by both systems and "All words" refers to using all of the evaluation data in the analysis. We highlight the fact that even in the cases where the baseline model gets the word right, our trained model outperforms the baseline on the frame level by an absolute 4%.

## 4. Pilot experiments: Continuous ASR

We also conduct preliminary experiments on a Switchboard [15] continuous word recognition task to demonstrate the feasibility of extending this approach to full recognition.

### 4.1. Experimental design

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 words, including disfluencies such as um, uh, and non-speech sounds such as noise and laughter. A trigram grammar was used to constrain decoding in the baseline system.

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]. Final decoding after discriminative training reintroduces the trigram grammar.

| Model | Dev WER (%) | Test WER (%) |
|---|---|---|
| Baseline system | 44.6 | 39.9 |
| Our system with a discriminatively trained $\mathcal{L}$ | 44.1 | 39.7 |

Table 3: Preliminary word error rates (WER) for the SWB continuous decoding task.

### 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-0905420. The opinions expressed in this work are those of the authors and do not necessarily reflect the views of the funding agency.

## 7. References

[1] M. Mohri, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[2] S. Kanthak, H. Ney, M. Riley, and M. Mohri, "A comparison of two LVCSR search optimization techniques," in *Proc. of ICSLP*, 2002.

[3] G. Saon, G. Zweig, B. Kingsbury, L. Mangu, and U. Chaudhari, "An architecture for rapid decoding of large vocabulary conversational speech," in *Proc. of Eurospeech*, 2003.

[4] S. Lin and F. Yvon, "Discriminative training of finite state decoding graphs," in *Proc. of Interspeech*, 2005.

[5] H. Kuo, B. Kingsbury, and G. Zweig, "Discriminative training of decoding graphs for large vocabulary continuous speech recognition," in *Proc. of ICASSP*, 2007.

[6] S. Watanabe, T. Hori, and A. Nakamura, "Large vocabulary continuous speech recognition using WFST-based linear classifier for structured data," in *Proc. of Interspeech*, 2010.

[7] B. Roark, M. Saraçlar, M. Collins, and M. Johnson, "Discriminative language modeling with conditional random fields and the perceptron algorithm," in *Proc. of ACL*, 2004.

[8] M. Lehr and I. Shafran, "Learning a discriminative weighted finite-state transducerfor speech recognition," *IEEE Trans on Audio, Speech, and Language Processing*, vol. 19, no. 5, pp. 1360–1367, 2011.

[9] P. Jyothi, E. Fosler-Lussier, and K. Livescu, "Discriminatively learning factorized finite state pronunciation models from dynamic Bayesian networks," in *Proc. of Interspeech*, 2012.

[10] E. Fosler-Lussier, "Multi-level decision trees for static and dynamic pronunciation models," in *Proc. Eurospeech*, 1999.

[11] M. Riley, W. Byrne, M. Finke, S. Khudanpur, A. Ljolje, J. McDonough, H. Nock, M. Saraçlar, C. Wooters, and G. Zavaliagkos, "Stochastic pronunciation modelling from hand-labelled phonetic corpora," *Speech Communication*, vol. 29, no. 2-4, pp. 209–224, 1999.

[12] O. Vinyals, L. Deng, D. Yu, and A. Acero, "Discriminative pronounciation learning using phonetic decoder and minimum-classification-error criterion," in *Proc. of ICASSP*, 2009.

[13] I. Badr, I. McGraw, and J. Glass, "Pronunciation learning from continuous speech," in *Proc. of Interspeech*, 2011.

[14] P. Karanasou, L. Burget, D. Vergyri, M. Akbacak, and A. Mandal, "Discriminatively trained phoneme confusion model for keyword spotting," in *Proc. of Interspeech*, 2012.

[15] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Proc. of ICASSP*, 1992.

[16] A. L. Yuille and A. Rangarajan, "The concave-convex procedure," *Neural Computation*, vol. 15, no. 4, pp. 915–936, 2003.

[17] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *The Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[18] R. Prabhavalkar, J. Keshet, K. Livescu, and E. Fosler-Lussier, "Discriminative spoken term detection with limited data," in *Symposium on Machine Learning in Speech and Language Processing (MLSLP)*, 2012.

[19] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Proc. of CIAA*, 2007.

[20] D. Povey, A. Ghoshal *et al.*, "The Kaldi speech recognition toolkit," *ASRU*, 2011.

[21] S. Greenberg, J. Hollenback, and D. Ellis, "Insights into spoken language gleaned from phonetic transcription of the Switchboard corpus," in *Proc. ICSLP*, 1996.

[22] K. Livescu, "Feature-based Pronunciation Modeling for Automatic Speech Recognition," *PhD dissertation*, 2005.

[23] "The NIST Scoring Toolkit (SCTK)," ftp://jaguar.ncsl.nist.gov/current_docs/sctk/doc/sctk.htm.

[24] R. Schlüter, B. Müller, F. Wessel, and H. Ney, "Interdependence of language models and discriminative training," in *Proc. IEEE ASRU Workshop*, 1999.

[25] D. Povey, "Discriminative training for large vocabulary speech recognition," *Ph.D. dissertation*, 2003.

[26] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1352–1365, 2007.

[27] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiát, S. Kombrink, P. Motlicek, Y. Qian *et al.*, "Generating exact lattices in the WFST framework," in *Proc. of ICASSP*, 2012, pp. 4213–4216.