Discriminative pronunciation modeling: A large-margin, feature-rich approach

Hao Tang, Joseph Keshet, and Karen Livescu
Toyota Technological Institute at Chicago
Chicago, IL USA
{haotang,jkeshet,klivescu}@ttic.edu

Abstract

We address the problem of learning the mapping between words and their possible pronunciations in terms of sub-word units. Most previous approaches have involved generative modeling of the distribution of pronunciations, usually trained to maximize likelihood. We propose a discriminative, feature-rich approach using large-margin learning. This approach allows us to optimize an objective closely related to a discriminative task, to incorporate a large number of complex features, and still do inference efficiently. We test the approach on the task of lexical access; that is, the prediction of a word given a phonetic transcription. In experiments on a subset of the Switchboard conversational speech corpus, our models thus far improve classification error rates from a previously published result of 29.1% to about 15%. We find that large-margin approaches outperform conditional random field learning, and that the Passive-Aggressive algorithm for large-margin learning is faster to converge than the Pegasos algorithm.

1 Introduction

One of the problems faced by automatic speech recognition, especially of conversational speech, is that of modeling the mapping between words and their possible pronunciations in terms of sub-word units such as phones. While pronouncing dictionaries provide each word's *canonical* pronunciation(s) in terms of phoneme strings, running speech often includes pronunciations that differ greatly from the dictionary. For example, some pronunciations of "probably" in the Switchboard conversational speech database are [p r aa b iy], [p r aa l iy], [p r ay], and [p ow ih] (Greenberg et al., 1996). While some words (e.g., common words) are more prone to such variation than others, the effect is extremely general: In the phonetically transcribed portion of Switchboard, fewer than half of the word tokens are pronounced canonically (Fosler-Lussier, 1999). In addition, pronunciation variants sometimes include sounds not present in the dictionary at all, such as nasalized vowels ("can't" → [k ae_n n t]) or fricatives introduced due to incomplete consonant closures ("legal" → [l iy g_fr ix l]).[1] This variation makes pronunciation modeling one of the major challenges facing speech recognition (McAllaster et al., 1998; Jurafsky et al., 2001; Saraçlar and Khudanpur, 2004; Bourlard et al., 1999). [2]

Most efforts to address the problem have involved either learning alternative pronunciations and/or their probabilities (Holter and Svendsen, 1999) or using phonetic transformation (substitution, insertion, and deletion) rules, which can come from linguistic knowledge or be learned from data (Riley et al., 1999; Hazen et al., 2005; Hutchinson and Droppo, 2011). These have produced some improvements in recognition performance. However, they also tend to cause additional confusability due to the introduction of additional homonyms (Fosler-

[1] We use the ARPAbet phonetic alphabet with additional diacritics, such as [_n] for nasalization and [_fr] for frication.

[2] This problem is separate from the grapheme-to-phoneme problem, in which pronunciations are predicted from a word's spelling; here, we assume the availability of a dictionary of canonical pronunciations as is usual in speech recognition.

Lussier et al., 2002). Some other alternatives are articulatory pronunciation models, in which words are represented as multiple parallel sequences of articulatory features rather than single sequences of phones, and which outperform phone-based models on some tasks (Livescu and Glass, 2004; Jyothi et al., 2011); and models for learning edit distances between dictionary and actual pronunciations (Ristad and Yianilos, 1998; Filali and Bilmes, 2005).

All of these approaches are generative—i.e., they provide distributions over possible pronunciations given the canonical one(s)—and they are typically trained by maximizing the likelihood over training data. In some recent work, discriminative approaches have been proposed, in which an objective more closely related to the task at hand is optimized. For example, (Vinyals et al., 2009; Korkmazskiy and Juang, 1997) optimize a minimum classification error (MCE) criterion to learn the weights (equivalently, probabilities) of alternative pronunciations for each word; (Schramm and Beyerlein, 2001) use a similar approach with discriminative model combination. In this work, the weighted alternatives are then used in a standard (generative) speech recognizer. In other words, these approaches optimize generative models using discriminative criteria.

We propose a general, flexible discriminative approach to pronunciation modeling, rather than discriminatively optimizing a generative model. We formulate a linear model with a large number of word-level and subword-level feature functions, whose weights are learned by optimizing a discriminative criterion. The approach is related to the recently proposed segmental conditional random field (SCRF) approach to speech recognition (Zweig et al., 2011). The main differences are that we optimize large-margin objective functions, which lead to sparser, faster, and better-performing models than conditional random field optimization in our experiments; and we use a large set of different feature functions tailored to pronunciation modeling.

In order to focus attention on the pronunciation model alone, our experiments focus on a task that measures only the mapping between words and subword units. Pronunciation models have in the past been tested using a variety of measures. For generative models, phonetic error rate of generated pronunciations (Venkataramani and Byrne, 2001) and phone- or frame-level perplexity (Riley et al., 1999; Jyothi et al., 2011) are appropriate measures. For our discriminative models, we consider the task of *lexical access*; that is, prediction of a single word given its pronunciation in terms of sub-word units (Fissore et al., 1989; Jyothi et al., 2011). This task is also sometimes referred to as "pronunciation recognition" (Ristad and Yianilos, 1998) or "pronunciation classification" (Filali and Bilmes, 2005).) As we show below, our approach outperforms both traditional phonetic rule-based models and the best previously published results on our data set obtained with generative articulatory approaches.

2 Problem setting

We define a *pronunciation of a word* as a representation of the way it is produced by a speaker in terms of some set of linguistically meaningful sub-word units. A pronunciation can be, for example, a sequence of phones or multiple sequences of *articulatory features* such as nasality, voicing, and tongue and lip positions. For purposes of this paper, we will assume that a pronunciation is a single sequence of units, but the approach applies to other representations. We distinguish between two types of pronunciations of a word: (i) *canonical* pronunciations, the ones typically found in the dictionary, and (ii) *surface* pronunciations, the ways a speaker may actually produce the word. In the task of *lexical access* we are given a surface pronunciation of a word, and our goal is to predict the word.

Formally, we define a pronunciation as a sequence of sub-word units $\overline{p} = (p_1, p_2, \ldots, p_K)$, where $p_k \in \mathcal{P}$ for all $1 \leq k \leq K$ and $\mathcal{P}$ is the set of all sub-word units. The index k can represent either a fixed-length frame or a variable-length segment. $\mathcal{P}^\star$ denotes the set of all finite-length sequences over $\mathcal{P}$. We denote a word by $w \in \mathcal{V}$ where $\mathcal{V}$ is the vocabulary. Our goal is to find a function $f : \mathcal{P}^\star \rightarrow \mathcal{V}$ that takes as input a surface pronunciation and returns the word from the vocabulary that was spoken.

In this paper we propose a discriminative supervised learning approach for learning the function f from a training set of pairs $(\overline{p}, w)$. We aim to find a function f that performs well on the training set as well as on unseen examples. Let $\hat{w} = f(\overline{p})$ be the predicted word given the pronunciation $\overline{p}$. We assess the quality of the function f by the *zero-one loss*: if

$w \neq \hat{w}$ then the error is one, otherwise the error is zero. The goal of the learning process is to minimize the expected zero-one loss, where the expectation is taken with respect to a fixed but unknown distribution over words and surface pronunciations. In the next section we present a learning algorithm that aims to minimize the expected zero-one loss.

3 Algorithm

Similarly to previous work in structured prediction (Taskar et al., 2003; Tsochantaridis et al., 2005), we construct the function f from a predefined set of N *feature functions*, $\{\phi_j\}_{j=1}^N$, each of the form $\phi_j : \mathcal{P}^* \times \mathcal{V} \to \mathbb{R}$. Each feature function takes a surface pronunciation $\overline{p}$ and a proposed word w and returns a scalar which, intuitively, should be correlated with whether the pronunciation $\overline{p}$ corresponds to the word w. The feature functions map pronunciations of different lengths along with a proposed word to a vector of fixed dimension in $\mathbb{R}^N$. For example, one feature function might measure the Levenshtein distance between the pronunciation $\overline{p}$ and the canonical pronunciation of the word w. This feature function counts the minimum number of edit operations (insertions, deletions, and substitutions) that are needed to convert the surface pronunciation to the canonical pronunciation; it is low if the surface pronunciation is close to the canonical one and high otherwise.

The function f maximizes a score relating the word w to the pronunciation $\overline{p}$. We restrict ourselves to scores that are linear in the feature functions, where each ϕ_j is scaled by a weight θ_j:

$$\sum_{j=1}^{N} \theta_j \phi_j(\overline{p}, w) = \boldsymbol{\theta} \cdot \boldsymbol{\phi}(\overline{p}, w),$$

where we have used vector notation for the feature functions $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_N)$ and for the weights $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)$. Linearity is not a very strong restriction, since the feature functions can be arbitrarily non-linear. The function f is defined as the word w that maximizes the score,

$$f(\overline{p}) = \operatorname*{argmax}_{w \in \mathcal{V}} \boldsymbol{\theta} \cdot \boldsymbol{\phi}(\overline{p}, w).$$

Our goal in learning $\boldsymbol{\theta}$ is to minimize the expected zero-one loss:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbb{E}_{(\overline{p},w)\sim\rho} \left[\mathbb{1}_{w \neq f(\overline{p})} \right],$$

where $\mathbb{1}_\pi$ is 1 if predicate π holds and 0 otherwise, and where ρ is an (unknown) distribution from which the examples in our training set are sampled i.i.d. Let $S = \{(\overline{p}_1, w_1), \ldots, (\overline{p}_m, w_m)\}$ be the training set. Instead of working directly with the zero-one loss, which is non-smooth and non-convex, we use the surrogate hinge loss, which upper-bounds the zero-one loss:

$$L(\boldsymbol{\theta}, \overline{p}_i, w_i) = \max_{w \in \mathcal{V}} \Big[\mathbb{1}_{w_i \neq w} - \boldsymbol{\theta} \cdot \boldsymbol{\phi}(\overline{p}_i, w_i) + \boldsymbol{\theta} \cdot \boldsymbol{\phi}(\overline{p}_i, w) \Big]. \quad (1)$$

Finding the weight vector $\boldsymbol{\theta}$ that minimizes the ℓ^2-regularized average of this loss function is the structured support vector machine (SVM) problem (Taskar et al., 2003; Tsochantaridis et al., 2005):

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 + \frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{\theta}, \overline{p}_i, w_i), \quad (2)$$

where λ is a user-defined tuning parameter that balances between regularization and loss minimization.

In practice, we have found that solving the quadratic optimization problem given in Eq. (2) converges very slowly using standard methods such as stochastic gradient descent (Shalev-Shwartz et al., 2007). We use a slightly different algorithm, the Passive-Aggressive (PA) algorithm (Crammer et al., 2006), whose average loss is comparable to that of the structured SVM solution (Keshet et al., 2007).

The Passive-Aggressive algorithm is an efficient online algorithm that, under some conditions, can be viewed as a dual-coordinate ascent minimizer of Eq. (2) (The connection to dual-coordinate ascent can be found in (Hsieh et al., 2008)). The algorithm begins by setting $\boldsymbol{\theta} = \mathbf{0}$ and proceeds in rounds. In the t-th round the algorithm picks an example $(\overline{p}_i, w_i)$ from S at random uniformly without replacement. Denote by $\boldsymbol{\theta}^{t-1}$ the value of the weight vector before the t-th round. Let $\hat{w}_i^t$ denote the predicted word for the i-th example according to $\boldsymbol{\theta}^{t-1}$:

$$\hat{w}_i^t = \operatorname*{argmax}_{w \in \mathcal{V}} \boldsymbol{\theta}^{t-1} \cdot \boldsymbol{\phi}(\overline{p}_i, w) + \mathbb{1}_{w_i \neq w}.$$

Let $\Delta\boldsymbol{\phi}_i^t = \boldsymbol{\phi}(\overline{p}_i, w_i) - \boldsymbol{\phi}(\overline{p}_i, \hat{w}_i^t)$. Then the algorithm updates the weight vector $\boldsymbol{\theta}^t$ as follows:

$$\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1} + \alpha_i^t \Delta\boldsymbol{\phi}_i^t \quad (3)$$

where

$$\alpha_i^t = \min\left\{\frac{1}{\lambda m}, \frac{\mathbb{1}_{w_i \neq \hat{w}_i^t} - \boldsymbol{\theta}\cdot\Delta\boldsymbol{\phi}_i^t}{\|\Delta\boldsymbol{\phi}_i^t\|}\right\}.$$

In practice we iterate over the m examples in the training set several times; each such iteration is an *epoch*. The final weight vector is set to the average over all weight vectors during training.

An alternative loss function that is often used to solve structured prediction problems is the *log-loss*:

$$L(\boldsymbol{\theta}, \overline{p}_i, w_i) = -\log P_{\boldsymbol{\theta}}(w_i|\overline{p}_i) \tag{4}$$

where the probability is defined as

$$P_{\boldsymbol{\theta}}(w_i|\overline{p}_i) = \frac{e^{\boldsymbol{\theta}\cdot\boldsymbol{\phi}(\overline{p}_i, w_i)}}{\sum_{w\in\mathcal{V}} e^{\boldsymbol{\theta}\cdot\boldsymbol{\phi}(\overline{p}, w)}}.$$

Minimization of Eq. (2) under the log-loss results in a probabilistic model commonly known as a conditional random field (CRF) (Lafferty et al., 2001). By taking the sub-gradient of Eq. (4), we can obtain an update rule similar to the one shown in Eq. (3).

4 Feature functions

Before defining the feature functions, we define some notation. Suppose $\overline{p} \in \mathcal{P}^*$ is a sequence of sub-word units. We use $\overline{p}_{1:n}$ to denote the n-gram substring $p_1 \dots p_n$. The two substrings $\overline{a}$ and $\overline{b}$ are said to be equal if they have the same length and $a_i = b_i$ for $1 \leq i \leq n$. For a given sub-word unit n-gram $\overline{u} \in \mathcal{P}^n$, we use the shorthand $\overline{u} \in \overline{p}$ to mean that we can find $\overline{u}$ in $\overline{p}$; i.e., there exists an index i such that $\overline{p}_{i:i+n} = \overline{u}$. We use $|\overline{p}|$ to denote the length of the sequence $\overline{p}$.

We assume we have a pronunciation dictionary, which is a set of words and their baseforms. We access the dictionary through the function *pron*, which takes a word $w \in \mathcal{V}$ and returns a set of baseforms.

4.1 TF-IDF feature functions

Term frequency (TF) and inverse document frequency (IDF) are measures that have been heavily used in information retrieval to search for documents using word queries (Salton et al., 1975). Similarly to (Zweig et al., 2010), we adapt TF and IDF by treating a sequence of sub-word units as a "document" and n-gram sub-sequences as "words." In this analogy, we use sub-sequences in surface pronunciations to "search" for baseforms in the dictionary. These features measure the frequency of each n-gram in observed pronunciations of a given word in the training set, along with the discriminative power of the n-gram. These features are therefore only meaningful for words actually observed in training.

The term frequency of a sub-word unit n-gram $\overline{u} \in \mathcal{P}^n$ in a sequence $\overline{p}$ is the length-normalized frequency of the n-gram in the sequence:

$$\mathrm{TF}_{\overline{u}}(\overline{p}) = \frac{1}{|\overline{p}| - |\overline{u}| + 1} \sum_{i=1}^{|\overline{p}|-|\overline{u}|+1} \mathbb{1}_{\overline{u}=\overline{p}_{i:i+|\overline{u}|-1}}.$$

Next, define the set of words in the training set that contain the n-gram $\overline{u}$ as $\mathcal{V}_{\overline{u}} = \{w \in \mathcal{V} \mid (\overline{p}, w) \in S, \overline{u} \in \overline{p}\}$. The inverse document frequency (IDF) of an n-gram $\overline{u}$ is defined as

$$\mathrm{IDF}_{\overline{u}} = \log \frac{|\mathcal{V}|}{|\mathcal{V}_{\overline{u}}|}.$$

IDF represents the discriminative power of an n-gram: An n-gram that occurs in few words is better at word discrimination than a very common n-gram.

Finally, we define word-specific features using TF and IDF. Suppose the vocabulary is indexed: $\mathcal{V} = \{w_1, \dots, w_n\}$. Define $\overline{e}_w$ as a binary vector with elements

$$(\overline{e}_w)_i = \mathbb{1}_{w_i = w}.$$

We define the TF-IDF feature function of $\overline{u}$ as

$$\phi_{\overline{u}}(\overline{p}, w) = (\mathrm{TF}_{\overline{u}}(\overline{p}) \times \mathrm{IDF}_{\overline{u}}) \otimes \overline{e}_w,$$

where $\otimes : \mathbb{R}^{a\times b} \times \mathbb{R}^{c\times d} \to \mathbb{R}^{ac\times bd}$ is the tensor product. We therefore have as many TF-IDF feature functions as we have n-grams. In practice, we only consider n-grams of a certain order (e.g., bigrams).

The following toy example demonstrates how the TF-IDF features are computed. Suppose we have $\mathcal{V} = \{\text{problem}, \text{probably}\}$. The dictionary maps "problem" to /pcl p r aa bcl b l ax m/ and "probably" to /pcl p r aa bcl b l iy/, and our input is $(\overline{p}, w) = (\text{[p r aa b l iy]}, \text{problem})$. Then for the bigram /l iy/, we have $\mathrm{TF}_{\text{/l iy/}}(\overline{p}) = 1/5$ (one out of five bigrams in $\overline{p}$), and $\mathrm{IDF}_{\text{/l iy/}} = \log(2/1)$ (one word out of two in the dictionary). The indicator vector is $\overline{e}_{\text{problem}} = \begin{bmatrix}1 & 0\end{bmatrix}^\top$, so the final feature is

$$\phi_{\text{/l iy/}}(\overline{p}, w) = \begin{bmatrix}\frac{1}{5}\log\frac{2}{1} \\ 0\end{bmatrix}.$$

4.2 Length feature function

The length feature functions measure how the length of a word's surface form tends to deviate from the baseform. These functions are parameterized by a and b and are defined as

$$\phi_{a \le \Delta\ell < b}(\overline{p}, w) = \mathbb{1}_{a \le \Delta\ell < b} \otimes \overline{e}_w,$$

where $\Delta\ell = |\overline{p}| - |\overline{v}|$, for some baseform $\overline{v} \in \text{pron}(w)$. The parameters a and b can be either positive or negative, so the model can learn whether the surface pronunciations of a word tend to be longer or shorter than the baseform. Like the TF-IDF features, this feature is only meaningful for words actually observed in training.

As an example, suppose we have $\mathcal{V} = \{\text{problem}, \text{probably}\}$, and the word "probably" has two baseforms, /pcl p r aa bcl b l iy/ (of length eight) and /pcl p r aa bcl b ax bcl b l iy/ (of length eleven). If we are given an input $(\overline{p}, w) = (\text{[pcl p r aa bcl l ax m]}, \text{probably})$, whose length of the surface form is eight, then the length features for the ranges $0 \le \Delta\ell < 1$ and $-3 \le \Delta\ell < -2$ are

$$\phi_{0 \le \Delta\ell < 1}(\overline{p}, w) = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top,$$
$$\phi_{-3 \le \Delta\ell < -2}(\overline{p}, w) = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top,$$

respectively. Other length features are all zero.

4.3 Phonetic alignment feature functions

Beyond the length, we also measure specific phonetic deviations from the dictionary. We define phonetic alignment features that count the (normalized) frequencies of phonetic insertions, phonetic deletions, and substitutions of one surface phone for another baseform phone. Given $(\overline{p}, w)$, we use dynamic programming to align the surface form $\overline{p}$ with all of the baseforms of w. Following (Riley et al., 1999), we encode a phoneme/phone with a 4-tuple: consonant manner, consonant place, vowel manner, and vowel place. Let the dash symbol "$-$" be a gap in the alignment (corresponding to an insertion/deletion). Given $p, q \in \mathcal{P} \cup \{-\}$, we say that a pair (p, q) is a deletion if $p \in \mathcal{P}$ and $q = -$, is an insertion if $p = -$ and $q \in \mathcal{P}$, and is a substitution if both $p, q \in \mathcal{P}$. Given $p, q \in \mathcal{P} \cup \{-\}$, let (s_1, s_2, s_3, s_4) and (t_1, t_2, t_3, t_4) be the corresponding 4-tuple encoding of p and q, respectively. The similarity between p and q is defined as

$$s(p, q) = \begin{cases} 1, & \text{if } p = - \text{ or } q = -; \\ \sum_{i=1}^{4} \mathbb{1}_{s_i = t_i}, & \text{otherwise.} \end{cases}$$

Consider aligning $\overline{p}$ with the $K_w = |\text{pron}(w)|$ baseforms of w. Define the length of the alignment with the k-th baseform as L_k, for $1 \le k \le K_w$. The resulting alignment is a sequence of pairs $(a_{k,1}, b_{k,1}), \ldots, (a_{k,L_k}, b_{k,L_k})$, where $a_{k,i}, b_{k,i} \in \mathcal{P} \cup \{-\}$ for $1 \le i \le L_k$. Now we define the alignment features, given $p, q \in \mathcal{P} \cup \{-\}$, as

$$\phi_{p \to q}(\overline{p}, w) = \frac{1}{Z_p} \sum_{k=1}^{K_w} \sum_{i=1}^{L_k} \mathbb{1}_{a_{k,i} = p,\, b_{k,i} = q},$$

where the normalization term is

$$Z_p = \begin{cases} \sum_{k=1}^{K_w} \sum_{i=1}^{L_k} \mathbb{1}_{a_{k,i} = p}, & \text{if } p \in P; \\ |\overline{p}| \cdot K_w & \text{if } p = -. \end{cases}$$

The normalization for insertions differs from the normalization for substitutions and deletions, so that the resulting values always lie between zero and one.

	pcl	p	r	aa	pcl	p	er	l	iy
	pcl	p	r	aa	bcl	b	$-$	l	iy
pcl	p	r	aa	pcl	p	er	$-$	$-$	l	iy
pcl	p	r	aa	bcl	b	ax	bcl	b	l	iy

Table 1: Possible alignments of [p r aa pcl p er l iy] with two baseforms of "probably" in the dictionary.

As an example, consider the input pair $(\overline{p}, w) = (\text{[p r aa pcl p er l iy]}, \text{probably})$ and suppose there are two baseforms of the word "probably" in the dictionary. Let one possible alignments be the one shown in Table 1. Since /p/ occurs four times in the alignments and two of them are aligned to [b], the feature for p $\to$ b is then $\phi_{\text{p} \to \text{b}}(\overline{p}, w) = 2/4$.

Unlike the TF-IDF feature functions and the length feature functions, the alignment feature functions can assign a non-zero score to words that are not seen at training time (but are in the dictionary), as long as there is a good alignment with their baseforms. The weights given to the alignment features are the analogue of substitution, insertion, and deletion rule probabilities in traditional phone-based pronunciation models such as (Riley et al., 1999); they can also be seen as a generalized version of the Levenshtein features of (Zweig et al., 2011).

4.4 Dictionary feature function

The dictionary feature is an indicator of whether a pronunciation is an exact match to a baseform, which also generalizes to words unseen in training. We define the dictionary feature as

$$\phi_{\text{dict}}(\overline{p}, w) = \mathbb{1}_{\overline{p} \in \text{pron}(w)}.$$

For example, assume there is a baseform /pcl p r aa bcl b l iy/ for the word "probably" in the dictionary, and $\overline{p} =$ /pcl p r aa bcl b l iy/. Then $\phi_{\text{dict}}(\overline{p}, \text{probably}) = 1$, while $\phi_{\text{dict}}(\overline{p}, \text{problem}) = 0$.

4.5 Articulatory feature functions

Articulatory models represented as dynamic Bayesian networks (DBNs) have been successful in the past on the lexical access task (Livescu and Glass, 2004; Jyothi et al., 2011). In such models, pronunciation variation is seen as the result of asynchrony between the articulators (lips, tongue, etc.) and deviations from the intended articulatory positions. Given a sequence $\overline{p}$ and a word w, we use the DBN to produce an alignment at the articulatory level, which is a sequence of 7-tuples, representing the articulatory variables[3] lip opening, tongue tip location and opening, tongue body location and opening, velum opening, and glottis opening. We extract three kinds of features from the output—substitutions, asynchrony, and log-likelihood.

The substitution features are similar to the phonetic alignment features in Section 4.3, except that the alignment is not a sequence of pairs but a sequence of 14-tuples (7 for the baseform and 7 for the surface form). The DBN model is based on articulatory phonology (Browman and Goldstein, 1992), in which there are no insertions and deletions, only substitutions (apparent insertions and deletions are accounted for by articulatory asynchrony). Formally, consider the seven sets of articulatory variable values $F_1, \ldots, F_7$. For example, F_1 could be all of the values of lip opening, $F_1 =$\{closed, critical, narrow, wide\}. Let $\mathcal{F} = \{F_1, \ldots, F_7\}$. Consider an articulatory variable $F \in \mathcal{F}$. Suppose the alignment for F is $(a_1, b_1), \ldots, (a_L, b_L)$, where L is the length of the alignment and $a_i, b_i \in F$, for $1 \leq i \leq L$. Here the a_i are the intended articulatory variable values according to the baseform, and the b_i are the corresponding realized values. For each $a, b \in F$ we define a substitution feature function:

$$\phi_{a \to b}(\overline{p}, w) = \frac{1}{L} \sum_{i=1}^{L} \mathbb{1}_{a_i = a,\, b_i = b}.$$

The asynchrony features are also extracted from the DBN alignments. Articulators are not always synchronized, which is one cause of pronunciation variation. We measure this by looking at the phones that two articulators are aiming to produce, and find the time difference between them. Formally, we consider two articulatory variables $F_h, F_k \in \mathcal{F}$. Let the alignment between the two variables be $(a_1, b_1), \ldots, (a_L, b_L)$, where now $a_i \in F_h$ and $b_i \in F_k$. Each a_i and b_i can be mapped back to the corresponding phone index $t_{h,i}$ and $t_{k,i}$, for $1 \leq i \leq L$. The average degree of asynchrony is then defined as

$$\text{async}(F_h, F_k) = \frac{1}{L} \sum_{i=1}^{L} \left(t_{h,i} - t_{k,i}\right).$$

More generally, we compute the average asynchrony between any two sets of variables $\mathcal{F}_1, \mathcal{F}_2 \subset \mathcal{F}$ as

$$\text{async}(\mathcal{F}_1, \mathcal{F}_2) = \frac{1}{L} \sum_{i=1}^{L} \left[\frac{1}{|\mathcal{F}_1|} \sum_{F_h \in \mathcal{F}_1} t_{h,i} - \frac{1}{|\mathcal{F}_2|} \sum_{F_k \in \mathcal{F}_2} t_{k,i} \right].$$

We then define the asynchrony features as

$$\phi_{a \leq \text{async}(\mathcal{F}_1, \mathcal{F}_2) \leq b} = \mathbb{1}_{a \leq \text{async}(\mathcal{F}_1, \mathcal{F}_2) \leq b}.$$

Finally, the log-likelihood feature is the DBN alignment score, shifted and scaled so that the value lies between zero and one,

$$\phi_{\text{dbn-LL}}(\overline{p}, w) = \frac{\mathcal{L}(\overline{p}, w) - h}{c},$$

where $\mathcal{L}$ is the log-likelihood function of the DBN, h is the shift, and c is the scale.

Note that none of the DBN features are word-specific, so that they generalize to words in the dictionary that are unseen in the training set.

5 Experiments

All experiments are conducted on a subset of the Switchboard conversational speech corpus that has

[3] We use the term "articulatory variable" for the "articulatory features" of (Livescu and Glass, 2004; Jyothi et al., 2011), in order to avoid confusion with our feature functions.

been labeled at a fine phonetic level (Greenberg et al., 1996); these phonetic transcriptions are the input to our lexical access models. The data subset, phone set $\mathcal{P}$, and dictionary are the same as ones previously used in (Livescu and Glass, 2004; Jyothi et al., 2011). The dictionary contains 3328 words, consisting of the 5000 most frequent words in Switchboard, excluding ones with fewer than four phones in their baseforms. The baseforms use a similar, slightly smaller phone set (lacking, e.g., nasalization). We measure performance by error rate (ER), the proportion of test examples predicted incorrectly.

The TF-IDF features used in the experiments are based on phone bigrams. For all of the articulatory DBN features, we use the DBN from (Livescu, 2005) (the one in (Jyothi et al., 2011) is more sophisticated and may be used in future work). For the asynchrony features, the articulatory pairs are $(\mathcal{F}_1, \mathcal{F}_2) \in \{(\{\text{tongue tip}\}, \{\text{tongue body}\}), (\{\text{lip opening}\}, \{\text{tongue tip, tongue body}\}), \text{and } (\{\text{lip opening, tongue tip, tongue body}\}, \{\text{glottis, velum}\})\}$, as in (Livescu, 2005). The parameters (a, b) of the length and asynchrony features are drawn from $(a, b) \in \{(-3, -2), (-2, -1), \ldots (2, 3)\}$.

We compare the CRF[4], Passive-Aggressive (PA), and Pegasos learning algorithms. The regularization parameter λ is tuned on the development set. We run all three algorithms for multiple epochs and pick the best epoch based on development set performance.

For the first set of experiments, we use the same division of the corpus as in (Livescu and Glass, 2004; Jyothi et al., 2011) into a 2492-word training set, a 165-word development set, and a 236-word test set. To give a sense of the difficulty of the task, we test two simple baselines. One is a lexicon lookup: If the surface form is found in the dictionary, predict the corresponding word; otherwise, guess randomly. For a second baseline, we calculate the Levenshtein (0-1 edit) distance between the input pronunciation and each dictionary baseform, and predict the word corresponding to the baseform closest to the input. The results are shown in the first two rows of Table 2. We can see that, by adding just the Levenshtein distance, the error rate drops significantly. However, both baselines do quite poorly.

[4]We use the term "CRF" since the learning algorithm corresponds to CRF learning, although the task is multiclass classification rather than a sequence or structure prediction task.

Model	ER
lexicon lookup (from (Livescu, 2005))	59.3%
lexicon + Levenshtein distance	41.8%
(Jyothi et al., 2011)	29.1%
CRF/DP+	21.5%
PA/DP+	**15.2%**
Pegasos/DP+	**14.8%**
PA/ALL	**15.2%**

Table 2: Lexical access error rates (ER) on the same data split as in (Livescu and Glass, 2004; Jyothi et al., 2011). Models labeled X/Y use learning algorithm X and feature set Y. The feature set DP+ contains TF-IDF, DP alignment, dictionary, and length features. The set ALL contains DP+ and the articulatory DBN features. The best results are in bold; the differences among them are insignificant (according to McNemar's test with $p = .05$).

Table 2 shows the best previous result on this data set from the articulatory model of Jyothi et al., which greatly improves over our baselines as well as over a much more complex phone-based model (Jyothi et al., 2011). The remaining rows of Table 2 give results with our feature functions and various learning algorithms. The best result for PA/DP+ (the PA algorithm using all features besides the DBN features) on the development set is with $\lambda = 100$ and 5 epochs. Tested on the test set, this model improves over (Jyothi et al., 2011) by 13.9% absolute (47.8% relative). The best result for Pegasos with the same features on the development set is with $\lambda = 0.01$ and 10 epochs. On the test set, this model gives a 14.3% absolute improvement (49.1% relative). CRF learning with the same features performs about 6% worse than the corresponding PA and Pegasos models.

The single-threaded running time for PA/DP+ and Pegasos/DP+ is about 40 minutes per epoch, measured on a dual-core AMD 2.4GHz CPU with 8GB of memory; for CRF, it takes about 100 minutes for each epoch, which is almost entirely because the weight vector $\boldsymbol{\theta}$ is less sparse with CRF learning. In the PA and Pegasos algorithms, we only update $\boldsymbol{\theta}$ for the most confusable word, while in CRF learning, we sum over all words. In our case, the number of non-zero entries in $\boldsymbol{\theta}$ for PA and Pegasos is around 800,000; for CRF, it is over 4,000,000. Though PA and Pegasos take roughly the same amount of time per epoch, Pegasos tends to require more epochs to

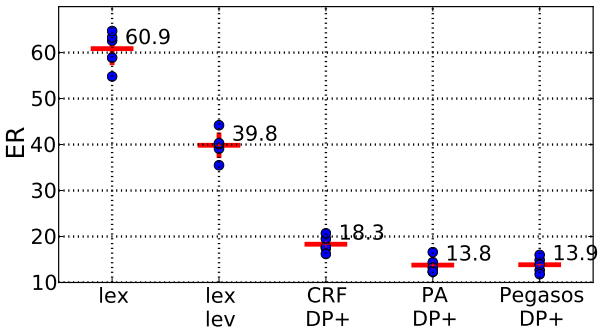

Figure 1: 5-fold cross validation (CV) results. The lexicon lookup baseline is labeled *lex*; *lex* + *lev* = lexicon lookup with Levenshtein distance. Each point corresponds to the test set error rate for one of the 5 data splits. The horizontal red line marks the mean of the results with means labeled, and the vertical red line indicates the mean plus and minus one standard deviation.

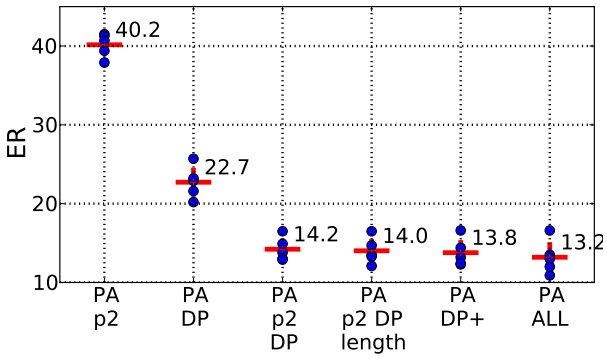

Figure 2: Feature selection results for five-fold cross validation. In the figure, phone bigram TF-IDF is labeled *p2*; phonetic alignment with dynamic programming is labeled *DP*. The dots and lines are as defined in Figure 1.

achieve the same performance as PA.

For the second experiment, we perform 5-fold cross-validation. We combine the training, development, and test sets from the previous experiment, and divide the data into five folds. We take three folds for training, one fold for tuning λ and the best epoch, and the remaining fold for testing. The results on the test fold are shown in Figure 1, which compares the learning algorithms, and Figure 2, which compares feature sets. Overall, the results are consistent with our first experiment. The feature selection experiments in Figure 2 shows that the TF-IDF features alone are quite weak, while the dynamic programming alignment features alone are quite good. Combining the two gives close to our best result. Although the marginal improvement gets smaller as we add more features, in general performance keeps improving the more features we add.

6 Discussion

The results in Section 5 are the best obtained thus far on the lexical access task on this conversational data set. Large-margin learning, using the Passive-Aggressive and Pegasos algorithms, has benefits over CRF learning for our task: It produces sparser models, is faster, and produces better lexical access results. In addition, the PA algorithm is faster than Pegasos on our task, as it requires fewer epochs.

Our ultimate goal is to incorporate such models into complete speech recognizers, that is to predict word sequences from acoustics. This requires (1) extension of the model and learning algorithm to word sequences and (2) feature functions that relate acoustic measurements to sub-word units. The extension to sequences can be done analogously to segmental conditional random fields (SCRFs). The main difference between SCRFs and our approach would be the large-margin learning, which can be straightforwardly applied to sequences. To incorporate acoustics, we can use feature functions based on classifiers of sub-word units, similarly to previous work on CRF-based speech recognition (Gunawardana et al., 2005; Morris and Fosler-Lussier, 2008; Prabhavalkar et al., 2011). Richer, longer-span (e.g., word-level) feature functions are also possible.

Thus far we have restricted the pronunciation-to-word score to linear combinations of feature functions. This can be extended to non-linear combinations using a kernel. This may be challenging in a high-dimensional feature space. One possibility is to approximate the kernels as in (Keshet et al., 2011). Additional extensions include new feature functions, such as context-sensitive alignment features, and joint inference and learning of the alignment models embedded in the feature functions.

Acknowledgments

We thank Raman Arora, Arild Næss, and the anonymous reviewers for helpful suggestions. This research was supported in part by NSF grant IIS-0905633. The opinions expressed in this work are those of the authors and do not necessarily reflect the views of the funding agency.

References

H. Bourlard, S. Furui, N. Morgan, and H. Strik. 1999. Special issue on modeling pronunciation variation for automatic speech recognition. *Speech Communication*, 29(2-4).

C. P. Browman and L. Goldstein. 1992. Articulatory phonology: an overview. *Phonetica*, 49(3-4).

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive aggressive algorithms. *Journal of Machine Learning Research*, 7.

K. Filali and J. Bilmes. 2005. A dynamic Bayesian framework to model context and memory in edit distance learning: An application to pronunciation classification. In *Proc. Association for Computational Linguistics (ACL)*.

L. Fissore, P. Laface, G. Micca, and R. Pieraccini. 1989. Lexical access to large vocabularies for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(8).

E. Fosler-Lussier, I. Amdal, and H.-K. J. Kuo. 2002. On the road to improved lexical confusability metrics. In *ISCA Tutorial and Research Workshop (ITRW) on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology*.

J. E. Fosler-Lussier. 1999. *Dynamic Pronunciation Models for Automatic Speech Recognition*. Ph.D. thesis, U. C. Berkeley.

S. Greenberg, J. Hollenback, and D. Ellis. 1996. Insights into spoken language gleaned from phonetic transcription of the Switchboard corpus. In *Proc. International Conference on Spoken Language Processing (ICSLP)*.

A. Gunawardana, M. Mahajan, A. Acero, and J. Platt. 2005. Hidden conditional random fields for phone classification. In *Proc. Interspeech*.

T. J. Hazen, I. L. Hetherington, H. Shu, and K. Livescu. 2005. Pronunciation modeling using a finite-state transducer representation. *Speech Communication*, 46(2).

T. Holter and T. Svendsen. 1999. Maximum likelihood modelling of pronunciation variation. *Speech Communication*.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear SVM. In *Proc. International Conference on Machine Learning (ICML)*.

B. Hutchinson and J. Droppo. 2011. Learning non-parametric models of pronunciation. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

D. Jurafsky, W. Ward, Z. Jianping, K. Herold, Y. Xiuyang, and Z. Sen. 2001. What kind of pronunciation variation is hard for triphones to model? In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

P. Jyothi, K. Livescu, and E. Fosler-Lussier. 2011. Lexical access experiments with context-dependent articulatory feature-based models. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

J. Keshet, S. Shalev-Shwartz, Y. Singer, and D. Chazan. 2007. A large margin algorithm for speech and audio segmentation. *IEEE Transactions on Acoustics, Speech, and Language Processing*, 15(8).

J. Keshet, D. McAllester, and T. Hazan. 2011. PAC-Bayesian approach for minimization of phoneme error rate. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

F. Korkmazskiy and B.-H. Juang. 1997. Discriminative training of the pronunciation networks. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. International Conference on Machine Learning (ICML)*.

K. Livescu and J. Glass. 2004. Feature-based pronunciation modeling with trainable asynchrony probabilities. In *Proc. International Conference on Spoken Language Processing (ICSLP)*.

K. Livescu. 2005. *Feature-based Pronunciation Modeling for Automatic Speech Recognition*. Ph.D. thesis, Massachusetts Institute of Technology.

D. McAllaster, L. Gillick, F. Scattone, and M. Newman. 1998. Fabricating conversational speech data with acoustic models : A program to examine model-data mismatch. In *Proc. International Conference on Spoken Language Processing (ICSLP)*.

J. Morris and E. Fosler-Lussier. 2008. Conditional random fields for integrating local discriminative classifiers. *IEEE Transactions on Acoustics, Speech, and Language Processing*, 16(3).

R. Prabhavalkar, E. Fosler-Lussier, and K. Livescu. 2011. A factored conditional random field model for articulatory feature forced transcription. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

M. Riley, W. Byrne, M. Finke, S. Khudanpur, A. Ljolje, J. McDonough, H. Nock, M. Saraclar, C. Wooters, and G. Zavaliagkos. 1999. Stochastic pronunciation modelling from hand-labelled phonetic corpora. *Speech Communication*, 29(2-4).

E. S. Ristad and P. N. Yianilos. 1998. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2).

G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18.

M. Saraçlar and S. Khudanpur. 2004. Pronunciation change in conversational speech and its implications for automatic speech recognition. *Computer Speech and Language*, 18(4).

H. Schramm and P. Beyerlein. 2001. Towards discriminative lexicon optimization. In *Proc. Eurospeech*.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. 2007. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proc. International Conference on Machine Learning (ICML)*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Advances in Neural Information Processing Systems (NIPS) 17*.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6.

V. Venkataramani and W. Byrne. 2001. MLLR adaptation techniques for pronunciation modeling. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

O. Vinyals, L. Deng, D. Yu, and A. Acero. 2009. Discriminative pronunciation learning using phonetic decoder and minimum-classification-error criterion. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

G. Zweig, P. Nguyen, and A. Acero. 2010. Continuous speech recognition with a TF-IDF acoustic model. In *Proc. Interspeech*.

G. Zweig, P. Nguyen, D. Van Compernolle, K. Demuynck, L. Atlas, P. Clark, G. Sell, M. Wang, F. Sha, H. Hermansky, D. Karakos, A. Jansen, S. Thomas, G.S.V.S. Sivaram, S. Bowman, and J. Kao. 2011. Speech recognition with segmental conditional random fields: A summary of the JHU CLSP 2010 summer workshop. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.