Joint Learning of Phonetic Units and Word Pronunciations for ASR

Chia-ying Lee, Yu Zhang, James Glass
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{chiaying,yzhang87,jrg}@csail.mit.edu

Abstract

The creation of a pronunciation lexicon remains the most inefficient process in developing an Automatic Speech Recognizer (ASR). In this paper, we propose an unsupervised alternative – requiring no language-specific knowledge – to the conventional manual approach for creating pronunciation dictionaries. We present a hierarchical Bayesian model, which jointly discovers the phonetic inventory and the Letter-to-Sound (L2S) mapping rules in a language using only transcribed data. When tested on a corpus of spontaneous queries, the results demonstrate the superiority of the proposed joint learning scheme over its sequential counterpart, in which the latent phonetic inventory and L2S mappings are learned separately. Furthermore, the recognizers built with the automatically induced lexicon consistently outperform grapheme-based recognizers and even approach the performance of recognition systems trained using conventional supervised procedures.

1 Introduction

Modern automatic speech recognizers require a few essential ingredients such as a signal representation of the speech signal, a search component, and typically a set of stochastic models that capture 1) the acoustic realizations of the basic sounds of a language, for example, phonemes, 2) the realization of words in terms of these sounds, and 3) how words are combined in spoken language. When creating a speech recognizer for a new language the usual requirements are: first, a large speech corpus with word-level annotations; second, a pronunciation dictionary that essentially defines a phonetic inventory for the language as well as word-level pronunciations, and third, optional additional text data that can be used to train the language model. Given these data and some decision about the signal representation, e.g., centi-second Mel-Frequency Cepstral Coefficients (MFCCs) (Davis and Mermelstein, 1980) with various derivatives, as well as the nature of the acoustic and language model such as 3-state HMMs and n-grams, iterative training methods can be used to effectively learn the model parameters for the acoustic and language models. Although the details of the components have changed through the years, this basic ASR formulation was well established by the late 1980's, and has not really changed much since then.

One of the interesting aspects of this formulation is the inherent dependence on the dictionary, which defines both the phonetic inventory of a language, and the pronunciations of all the words in the vocabulary. The dictionary is arguably the cornerstone of a speech recognizer as it provides the essential transduction from sounds to words. Unfortunately, the dependency on this resource is a significant impediment to the creation of speech recognizers for new languages, since they are typically created by experts, whereas annotated corpora can be relatively more easily created by native speakers of a language.

The existence of an expert-derived dictionary in the midst of stochastic speech recognition models is somewhat ironic, and it is natural to ask why it continues to receive special status after all these years. Why can we not learn the inventory of sounds of a language and associated word pronunciations automatically, much as we learn our acoustic model parameters? If successful, we would move one step forward towards breaking the language barrier that

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 182–192,
Seattle, Washington, USA, 18-21 October 2013.

limits us from having speech recognizers for all languages of the world, instead of the less than 2% that currently exist.

In this paper, we investigate the problem of inferring a pronunciation lexicon from an annotated corpus without exploiting any language-specific knowledge. We formulate our approach as a hierarchical Bayesian model, which jointly discovers the acoustic inventory and the latent encoding scheme between the letters and the sounds of a language. We evaluate the quality of the induced lexicon and acoustic model through a series of speech recognition experiments on a conversational weather query corpus (Zue et al., 2000). The results demonstrate that our model consistently generates close performance to recognizers that are trained with expert-defined phonetic inventory and lexicon. Compared to grapheme-based recognizers, our model is capable of improving the Word Error Rates (WERs) by at least 15.3%. Finally, the joint learning framework proposed in this paper is proven to be much more effective than modeling the acoustic units and the letter-to-sound mappings separately, as shown in a 45% WER deduction our model achieves compared to a sequential approach.

2 Related Work

Various algorithms for learning sub-word based pronunciations were proposed in (Lee et al., 1988; Fukada et al., 1996; Bacchiani and Ostendorf, 1999; Paliwal, 1990). In these previous approaches, spoken samples of a word are gathered, and usually only one single pronunciation for the word is derived based on the acoustic evidence observed in the spoken samples. The major difference between our work and these previous works is that our model learns word pronunciations in the context of letter sequences. More specifically, our model learns letter pronunciations first and then concatenates the pronunciation of each letter in a word to form the word pronunciation. The advantage of our approach is that pronunciation knowledge learned for a particular letter in some arbitrary word can subsequently be used to help learn the letter's pronunciation in other words. This property allows our model to potentially learn better pronunciations for less frequent words.

The more recent work by Garcia and Gish (2006) and Siu et al. (2013) has made extensive use of self-organizing units for keyword spotting and other tasks for languages with limited linguistic resources. Others who have more recently explored the unsupervised space include (Varadarajan et al., 2008; Jansen and Church, 2011; Lee and Glass, 2012). The latter work introduced a non-parametric Bayesian inference procedure for automatically learning acoustic units that is most similar to our current work except that our model also infers word pronunciations simultaneously.

The concept of creating a speech recognizer for a language with only orthographically annotated speech data has also been explored previously by means of graphemes. This approach has been shown to be effective for alphabetic languages with relatively straightforward grapheme to phoneme transformations and does not require any unsupervised learning of units or pronunciations (Killer et al., 2003; Stüker and Schultz, 2004). As we explain in later sections, grapheme-based systems can actually be regarded as a special case of our model; therefore, we expect our model to have greater flexibilities for capturing pronunciation rules of graphemes.

3 Model

The goal of our model is to induce a word pronunciation lexicon from spoken utterances and their corresponding word transcriptions. No other language-specific knowledge is assumed to be available, including the phonetic inventory of the language. To achieve the goal, our model needs to solve the following two tasks:

- Discover the phonetic inventory.
- Reveal the latent mapping between the letters and the discovered phonetic units.

We propose a hierarchical Bayesian model for jointly discovering the two latent structures from an annotated speech corpus. Before presenting our model, we first describe the key latent and observed variables of the problem.

Letter (l_i^m) We use l_i^m to denote the i^{th} letter observed in the word transcription of the m^{th} training sample. To be sure, a training sample involves a speech utterance and its

corresponding text transcription. The letter sequence composed of l_i^m and its context, namely $l_{i-\kappa}^m, \cdots, l_{i-1}^m, l_i^m, l_{i+1}^m, \cdots, l_{i+\kappa}^m$, is denoted as $\vec{l}_{i,\kappa}^m$. Although l_i^m is referred to as a *letter* in this paper, it can represent any *character* observed in the text data, including space and symbols indicating sentence boundaries. The set of unique characters observed in the data set is denoted as G. For notation simplicity, we use $\mathcal{L}_\kappa$ to denote the set of letter sequences of length $2\kappa + 1$ that appear in the dataset and use $\vec{l}_\kappa$ to denote the elements in $\mathcal{L}_\kappa$. Finally, $\mathbb{P}(\vec{l}_\kappa)$ is used to represent the *parent* of $\vec{l}_\kappa$, which is a substring of $\vec{l}_\kappa$ with the first and the last characters truncated.

Number of Mapped Acoustic Units (n_i^m) Each letter l_i^m in the transcriptions is assumed to be mapped to a certain number of phonetic units. For example, the letter *x* in the word *fox* is mapped to 2 phonetic units /k/ and /s/, while the letter *e* in the word *lake* is mapped to 0 phonetic units. We denote this number as n_i^m and limit its value to be 0, 1 or 2 in our model. The value of n_i^m is always unobserved and needs to be inferred by the our model.

Identity of the Acoustic Unit ($c_{i,p}^m$) For each phonetic unit that l_i^m maps to, we use $c_{i,p}^m$, for $1 \leq p \leq n_i^m$, to denote the identity of the phonetic unit. Note that the phonetic inventory that describes the data set is unknown to our model, and the identities of the phonetic units are associated with the acoustic units discovered automatically by our model.

Speech Feature x_t^m The observed speech data in our problem are converted to a series of 25 ms 13-dimensional MFCCs (Davis and Mermelstein, 1980) and their first- and second-order time derivatives at a 10 ms analysis rate. We use $x_t^m \in \mathbb{R}^{39}$ to denote the t^{th} feature frame of the m^{th} utterance.

3.1 Generative Process

We present the generative process for a single training sample (i.e., a speech utterance and its corresponding text transcription); to keep notation simple, we discard the index variable m in this section.

For each l_i in the transcription, the model generates n_i, given $\vec{l}_{i,\kappa}$, from the 3-dimensional categorical distribution $\phi_{\vec{l}_{i,\kappa}}(n_i)$. Note that for every unique $\vec{l}_{i,\kappa}$ letter sequence, there is an associated $\phi_{\vec{l}_{i,\kappa}}(n_i)$ distribution, which captures the fact that the number of phonetic units a letter maps to may depend on its context. In our model, we impose a Dirichlet distribution prior $Dir(\eta)$ on $\phi_{\vec{l}_{i,\kappa}}(n_i)$.

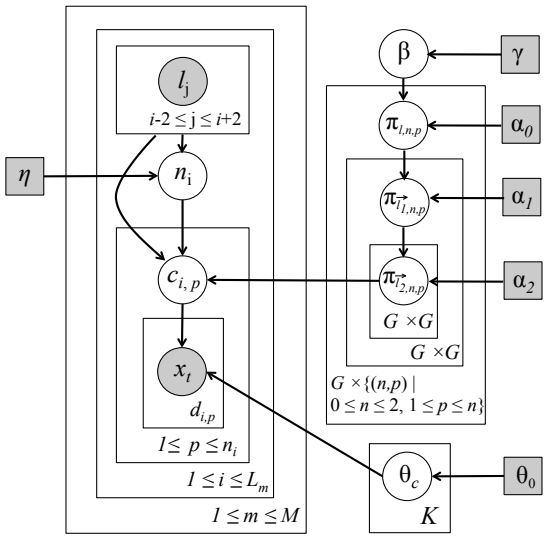

Figure 1: The graphical representation of the proposed hierarchical Bayesian model. The shaded circle denotes the observed text and speech data, and the squares denote the hyperparameters of the priors in our model. See Sec. 3 for a detailed explanation of the generative process of our model.

If $n_i = 0$, l_i is not mapped to any acoustic units and the generative process stops for l_i; otherwise, for $1 \leq p \leq n_i$, the model generates $c_{i,p}$ from:

$$c_{i,p} \sim \pi_{\vec{l}_{i,\kappa},n_i,p} \tag{1}$$

where $\pi_{\vec{l}_{i,\kappa},n_i,p}$ is a K-dimensional categorical distribution, whose outcomes correspond to the phonetic units discovered by the model from the given speech data. Eq. 1 shows that for each combination of $\vec{l}_{i,\kappa}$, n_i and p, there is an unique categorical distribution. An important property of these categorical distributions is that they are coupled together such that their outcomes point to a consistent set of phonetic units. In order to enforce the coupling, we construct $\pi_{\vec{l}_{i,\kappa},n_i,p}$ through a hierarchical process.

$$\beta \sim Dir(\gamma) \tag{2}$$

$$\pi_{\vec{l}_{i,\kappa},n_i,p} \sim Dir(\alpha_\kappa \beta) \text{ for } \kappa = 0 \tag{3}$$

$$\pi_{\vec{l}_{i,\kappa},n_i,p} \sim Dir(\alpha_\kappa \pi_{\vec{l}_{i,\kappa-1},n_i,p}) \text{ for } \kappa \geq 1 \tag{4}$$

To interpret Eq. 2 to Eq. 4, we envision that the observed speech data are generated by a K-component mixture model, of which the components correspond to the phonetic units in the language. As a result, β in Eq. 2 can be viewed as the mixture weight over the components, which indicates how likely we are to observe each acoustic unit in the data overall. By adopting this point of view, we can also regard the mapping between l_i and the phonetic units as a mixture model, and $\pi_{l_i,n_i,p}$[1] represents how probable l_i is mapped to each phonetic unit given n_i and p. We apply a Dirichlet distribution prior parametrized by $\alpha_0\beta$ to $\pi_{l_i,n_i,p}$ as shown in Eq. 3. With this parameterization, the mean of $\pi_{l_i,n_i,p}$ is the global mixture weight β, and α_0 controls how similar $\pi_{l_i,n_i,p}$ is to the mean. More specifically, for large $\alpha_0 \gg K$, the Dirichlet distribution is highly peaked around the mean; on the contrary, for $\alpha_0 \ll K$, the mean lies in a valley. The parameters of a Dirichlet distribution can also be viewed as pseudo-counts for each category. Eq. 4 shows that the prior for $\pi_{\vec{l}_{i,\kappa},n_i,p}$ is seeded by pseudo-counts that are proportional to the mapping weights over the phonetic units of l_i in a shorter context. In other words, the mapping distribution of l_i in a shorter context can be thought of as a back-off distribution of l_i's mapping weights in a longer context.

Each component of the K-dimensional mixture model is linked to a 3-state Hidden Markov Model (HMM). These K HMMs are used to model the phonetic units in the language (Jelinek, 1976). The emission probability of each HMM state is modeled by a diagonal Gaussian Mixture Model (GMM). We use θ_c to represent the set of parameters that define the c^{th} HMM, which includes the state transition probability and the GMM parameters of each state emission distribution. The conjugate prior of θ_c is denoted as $H(\theta_0)$[2].

Finally, to finish the generative process, for each $c_{i,p}$ we use the corresponding HMM $\theta_{c_{i,p}}$ to generate the observed speech data x_t, and the generative process of the HMM determines the duration, $d_{i,p}$, of the speech segment. The complete generative model, with κ set to 2, is depicted in Fig. 1; M is the total number of transcribed utterances in the corpus, and L_m is the number of letters in utterance m. The shaded circles denote the observed data, and the squares denote the hyperparameters of the priors used in our model. Lastly, the unshaded circles denote the latent variables of our model, for which we derive inference algorithms in the next section.

[1] An abbreviation of $\pi_{\vec{l}_{i,0},n_i,p}$

[2] $H(\theta_0)$ includes a Dirichlet prior for the transition probability of each state, and a Dirichlet prior for each mixture weight of the three GMMs, and a normal-Gamma distribution for the mean and precision of each Gaussian mixture in the 3-state HMM.

4 Inference

We employ Gibbs sampling (Gelman et al., 2004) to approximate the posterior distribution of the latent variables in our model. In the following sections, we first present a message-passing algorithm for block-sampling n_i and $c_{i,p}$, and then describe how we leverage acoustic cues to accelerate the computation of the message-passing algorithm. Note that the block-sampling algorithm for n_i and $c_{i,p}$ can be parallelized across utterances. Finally, we briefly discuss the inference procedures for $\phi_{\vec{l}_\kappa}$, $\pi_{\vec{l}_\kappa,n,p}$, β, θ_c.

4.1 Block-sampling n_i and $c_{i,p}$

To understand the message-passing algorithm in this study, it is helpful to think of our model as a simplified Hidden Semi-Markov Model (HSMM), in which the letters represent the states and the speech features are the observations. However, unlike in a regular HSMM, where the state sequence is hidden, in our case, the state sequence is fixed to be the given letter sequence. With this point of view, we can modify the message-passing algorithms of Murphy (2002) and Johnson and Willsky (2013) to compute the posterior information required for block-sampling n_i and $c_{i,p}$.

Let $\mathbb{L}(x_t)$ be a function that returns the index of the letter from which x_t is generated; also, let $F_t = 1$ be a tag indicating that a new phone segment starts at $t+1$. Given the constraint that $0 \leq n_i \leq 2$, for $0 \leq i \leq L_m$ and $0 \leq t \leq T_m$, the backwards messages $B_t(i)$ and $B^*_t(i)$ for the m^{th} training sample can be defined and computed as in Eq. 5 and Eq. 7. Note that for clarity we discard the index variable m in the derivation of the algorithm.

$$
\begin{aligned}
B_t(i) &\triangleq p(x_{t+1:T}|\mathbb{L}(x_t) = i, F_t = 1) \\
&= \sum_{j=i+1}^{\min\{L,i+1+U\}} B_t^*(j) \prod_{k=i+1}^{j-1} p(n_k = 0|\vec{l}_{i,\kappa}) \\
&= \sum_{j=i+1}^{\min\{L,i+1+U\}} B_t^*(j) \prod_{k=i+1}^{j-1} \phi_{\vec{l}_{i,\kappa}}(0) \qquad (5)
\end{aligned}
$$

$$
\begin{aligned}
&B_t^*(i) \triangleq p(x_{t+1:T}|\mathbb{L}(x_{t+1}) = i, F_t = 1) \\
&= \sum_{d=1}^{T-t} p(x_{t+1:t+d}|\vec{l}_{i,\kappa}) B_{t+d}(i) \qquad (6) \\
&= \sum_{d=1}^{T-t} \{ \sum_{c_{i,1}=1}^{K} \phi_{\vec{l}_{i,\kappa}}(1) \pi_{\vec{l}_{i,\kappa},1,1}(c_{i,1}) p(x_{t+1:t+d}|\theta_{c_{i,1}}) \\
&+ \sum_{v=1}^{d-1} \sum_{c_{i,1}}^{K} \sum_{c_{i,2}}^{K} \phi_{\vec{l}_{i,\kappa}}(2) \pi_{\vec{l}_{i,\kappa},2,1}(c_{i,1}) \pi_{\vec{l}_{i,\kappa},2,2}(c_{i,2}) \\
&\times p(x_{t+1:t+v}|\theta_{c_{i,1}}) p(x_{t+v+1:t+d}|\theta_{c_{i,2}}) \} B_{t+d}(i) \qquad (7)
\end{aligned}
$$

We use $x_{t_1:t_2}$ to denote the segment consisting of $x_{t_1}, \cdots, x_{t_2}$. Our inference algorithm only allows up to U letters to emit 0 acoustic units in a row. The value of U is set to 2 for our experiments. $B_t(i)$ represents the total probability of all possible alignments between $x_{t+1:T}$ and $l_{i+1:L}$. $B_t^*(i)$ contains the probability of all the alignments between $x_{t+1:T}$ and $l_{i+1:L}$ that map x_{t+1} to l_i particularly. This alignment constraint between x_{t+1} and l_i is explicitly shown in the first term of Eq. 6, which represents how likely the speech segment $x_{t+1:t+d}$ is generated by l_i given l_i's context. This likelihood is simply the marginal probability of $p(x_{t+1:t+d}, n_i, c_{i,p}|\vec{l}_{i,\kappa})$ with n_i and $c_{i,p}$ integrated out, which can be expanded and computed as shown in the last three rows of Eq. 7. The index v specifies where the phone boundary is between the two acoustic units that l_i is aligned with when $n_i = 2$. Eq. 8 to Eq. 10 are the boundary conditions of the message passing algorithm. $B_0(0)$ carries the total probably of all possible alignments between $l_{1:L}$ and $x_{1:T}$. Eq. 9 specifies that at most U letters at the end of an sentence can be left unaligned with any speech features, while Eq. 10 indicates that all of the speech features in an utterance must be assigned to a letter.

Algorithm 1 Block-sample n_i and $c_{i,p}$ from $B_t(i)$ and $B_t^*(i)$

```
1:  i ← 0
2:  t ← 0
3:  while i < L ∧ t < T do
4:      next_i ← SampleFromB_t(i)
5:      if next_i > i + 1 then
6:          for k = i + 1 to k = next_i − 1 do
7:              n_k ← 0
8:          end for
9:      end if
10:     d, n_i, ⟨c_{i,p}⟩, v ← SampleFromB*_t(next_i)
11:     t ← t + d
12:     i ← next_i
13: end while
```

$$
B_0(0) = \sum_{j=1}^{\min\{L,U+1\}} B_0^*(j) \prod_{k=1}^{j-1} \phi_{\vec{l}_{i,\kappa}}(0) \qquad (8)
$$

$$
B_T(i) \triangleq \begin{cases} 1 & \text{if } i = L \\ \prod_{j=i+1}^{L} \phi_{\vec{l}_{i,\kappa}}(0) & \text{if } L - U \leq i < L \\ 0 & \text{if } i < L - U \end{cases} \qquad (9)
$$

$$
B_t(L) \triangleq \begin{cases} 1 & \text{if } t = T \\ 0 & \text{otherwise} \end{cases} \qquad (10)
$$

Given $B_t(i)$ and $B_t^*(i)$, n_i and $c_{i,p}$ for each letter in the utterance can be sampled using Alg. 1. The $SampleFromB_t(i)$ function in line 4 returns a random sample from the relative probability distribution composed by entries of the summation in Eq. 5. Line 5 to line 9 check whether l_i (and maybe l_{i+1}) is mapped to zero phonetic units. $next_i$ points to the letter that needs to be aligned with 1 or 2 phone segments starting from x_t. The number of phonetic units that l_{next_i} maps to and the identities of the units are sampled in $SampleFromB_t^*(i)$. This subroutine generates a tuple of d, n_i, $\langle c_{i,p} \rangle$ as well as v (if $n_i = 2$) from all the entries of the summation shown in Eq. 7[3].

[3]We use $\langle c_{i,p} \rangle$ to denote that $\langle c_{i,p} \rangle$ may consist of two numbers, $c_{i,1}$ and $c_{i,2}$, when $n_i = 2$.

4.2 Heuristic Phone Boundary Elimination

The variables d and v in Eq. 7 enumerate through every frame index in a sentence, treating each feature frame as a potential boundary between acoustic units. However, it is possible to exploit acoustic cues to avoid checking feature frames that are unlikely to be phonetic boundaries. We follow the presegmentation method described in Glass (2003) to skip roughly 80% of the feature frames and greatly speed up the computation of $B_t^*(i)$.

Another heuristic applied to our algorithm to reduce the search space for d and v is based on the observation that the average duration of phonetic units is usually no longer than 300 ms. Therefore, when computing $B_t^*(i)$, we only consider speech segments that are shorter than 300 ms to avoid aligning letters to speech segments that are too long to be phonetic units.

4.3 Sampling $\phi_{\vec{l}_\kappa}$, $\pi_{\vec{l}_\kappa,n_i,p}$, β and θ_c

Sampling $\phi_{\vec{l}_\kappa}$ To compute the posterior distribution of $\phi_{\vec{l}_\kappa}$, we count how many times $\vec{l}_\kappa$ is mapped to 0, 1 and 2 phonetic units from n_i^m. More specifically, we define $\mathcal{N}_{\vec{l}_\kappa}(j)$ for $0 \leq j \leq 2$ as follows:

$$\mathcal{N}_{\vec{l}_\kappa}(j) = \sum_{m=1}^{M}\sum_{i=1}^{L_m} \delta(n_i^m, j)\delta(\vec{l}_{i,\kappa}^m, \vec{l}_\kappa)$$

where we use $\delta(\cdot)$ to denote the discrete Kronecker delta. With $\mathcal{N}_{\vec{l}_\kappa}$, we can simply sample a new value for $\phi_{\vec{l}_\kappa}$ from the following distribution:

$$\phi_{\vec{l}_\kappa} \sim Dir(\eta + \mathcal{N}_{\vec{l}_\kappa})$$

Sampling $\pi_{\vec{l}_\kappa,n,p}$ and β The posterior distributions of $\pi_{\vec{l}_\kappa,n,p}$ and β are constructed recursively due to the hierarchical structure imposed on $\pi_{\vec{l}_\kappa,n,p}$ and β. We start with gathering counts for updating the π variables at the lowest level, i.e., $\pi_{\vec{l}_2,n,p}$ given that κ is set to 2 in our model implementation, and then sample pseudo-counts for the π variables at higher hierarchies as well as β. With the pseudo-counts, a new β can be generated, which allows $\pi_{\vec{l}_\kappa,n,p}$ to be re-sampled sequentially.

More specifically, we define $\mathcal{C}_{\vec{l}_2,n,p}(k)$ to be the number of times that $\vec{l}_2$ is mapped to n units and the unit in position p is the k^{th} phonetic unit. This value can be counted from the current values of $c_{i,p}^m$ as follows.

$$\mathcal{C}_{\vec{l}_2,n,p}(k) = \sum_{m=1}^{M}\sum_{i=1}^{L_m} \delta(\vec{l}_{i,2}, \vec{l}_2)\delta(n_i^m, n)\delta(c_{i,p}^m, k)$$

To derive the posterior distribution of $\pi_{\vec{l}_1,n,p}$ analytically, we need to sample pseudo-counts $\mathcal{C}_{\vec{l}_1,n,p}$, which is defined as follows.

$$\mathcal{C}_{\vec{l}_1,n,p}(k) = \sum_{\vec{l}_2 \in \mathcal{U}_{\vec{l}_1}} \sum_{i=1}^{\mathcal{C}_{\vec{l}_2,n,p}(k)} \mathbb{I}[\nu_i < \frac{\alpha_2 \pi_{\vec{l}_1,n,p}(k)}{i + \alpha_2 \pi_{\vec{l}_1,n,p}(k)}] \tag{11}$$

We use $\mathcal{U}_{\vec{l}_1} = \{\vec{l}_2 | \mathbb{P}(\vec{l}_2) = \vec{l}_1\}$ to denote the set of $\vec{l}_2$ whose parent is $\vec{l}_1$ and ν_i to represent random variables sampled from a uniform distribution between 0 and 1. Eq. 11 can be applied recursively to compute $\mathcal{C}_{\vec{l}_0,n,p}(k)$ and $\mathcal{C}_{\_,n,p}(k)$, the pseudo-counts that are applied to the conjugate priors of $\pi_{\vec{l}_0,n,p}$ and β. With the pseudo-count variables computed, new values for β and $\pi_{\vec{l}_\kappa,n,p}$ can be sampled sequentially as shown in Eq. 12 to Eq. 14.

$$\beta \sim Dir(\gamma + \mathcal{C}_{\_,n,p}) \tag{12}$$

$$\pi_{\vec{l}_\kappa,n,p} \sim Dir(\alpha_\kappa \beta + \mathcal{C}_{\vec{l}_\kappa,n,p}) \text{ for } \kappa = 0 \tag{13}$$

$$\pi_{\vec{l}_\kappa,n,p} \sim Dir(\alpha_\kappa \pi_{\vec{l}_{\kappa-1},n,p} + \mathcal{C}_{\vec{l}_\kappa,n,p}) \text{ for } \kappa \geq 1 \tag{14}$$

5 Experimental Setup

To test the effectiveness of our model for joint learning phonetic units and word pronunciations from an annotated speech corpus, we construct speech recognizers out of the training results of our model. The performance of the recognizers is evaluated and compared against three baselines: first, a grapheme-based speech recognizer; second, a recognizer built by using an expert-crafted lexicon, which is referred to as an expert lexicon in the rest of the paper for simplicity; and third, a recognizer built by discovering the phonetic units and L2S pronunciation rules *sequentially* without using a lexicon. In this section, we provide a detailed description of the experimental setup.

η	γ	α_0	α_1	α_2	θ_0	κ	K
$\langle 0.1 \rangle_3$	$\langle 10 \rangle_{100}$	1	0.1	0.2	*	2	100

Table 1: The values of the hyperparameters of our model. We use $\langle a \rangle_D$ to denote a D-dimensional vector with all entries being a. *We follow the procedure reported in (Lee and Glass, 2012) to set up the HMM prior θ_0.

5.1 Dataset

All the speech recognition experiments reported in this paper are performed on a weather query dataset, which consists of narrow-band, conversational telephone speech (Zue et al., 2000). We follow the experimental setup of McGraw et al. (2013) and split the corpus into a training set of 87,351 utterances, a dev set of 1,179 utterances and a test set of 3,497 utterances. A subset of 10,000 utterances is randomly selected from the training set. We use this subset of data for training our model to demonstrate that our model is able to discover the phonetic composition and the pronunciation rules of a language even from just a few hours of data.

5.2 Building a Recognizer from Our Model

The values of the hyperparameters of our model are listed in Table 1. We run the inference procedure described in Sec. 4 for 10,000 times on the randomly selected 10,000 utterances. The samples of $\phi_{\vec{l}_\kappa}$ and $\pi_{\vec{l}_\kappa n,p}$ from the last iteration are used to decode n_i^m and $c_{i,p}^m$ for each sentence in the entire training set by following the block-sampling algorithm described in Sec. 4.1. Since $c_{i,p}^m$ is the phonetic mapping of l_i^m, by concatenating the phonetic mapping of every letter in a word, we can obtain a pronunciation of the word represented in the labels of discovered phonetic units. For example, assume that word w appears in sentence m and consists of $l_3 l_4 l_5$ (the sentence index m is ignored for simplicity). Also, assume that after decoding, $n_3 = 1$, $n_4 = 2$ and $n_5 = 1$. A pronunciation of w is then encoded by the sequence of phonetic labels $c_{3,1} c_{4,1} c_{4,2} c_{5,1}$. By repeating this process for each word in every sentence for the training set, a list of word pronunciations can be compiled and used as a stochastic lexicon to build a speech recognizer.

In theory, the HMMs inferred by our model can be directly used as the acoustic model of a monophone speech recognizer. However, if we regard the $c_{i,p}$ labels of each utterance as the phone transcription of the sentence, then a new acoustic model can be easily re-trained on the entire data set. More conveniently, the phone boundaries corresponding to the $c_{i,p}$ labels are the by-products of the block-sampling algorithm, which are indicated by the values of d and v in line 10 of Alg. 1 and can be easily saved during the sampling procedure. Since these data are readily available, we re-build a context-independent model on the entire data set. In this new acoustic model, a 3-state HMM is used to model each phonetic unit, and the emission probability of each state is modeled by a 32-mixture GMM.

Finally, a trigram language model is built by using the word transcriptions in the full training set. This language model is utilized in all speech recognition experiments reported in this paper. Finite State Transducers (FSTs) are used to build all the recognizers used in this study. With the language model, the lexicon and the context-independent acoustic model constructed by the methods described in this section, we can build a speech recognizer from the learning output of the proposed model without the need of a pre-defined phone inventory and any expert-crafted lexicons.

5.2.1 Pronunciation Mixture Model Retraining

McGraw et al. (2013) presented the Pronunciation Mixture Model (PMM) for composing stochastic lexicons that outperform pronunciation dictionaries created by experts. Although the PMM framework was designed to incorporate and augment expert lexicons, we found that it can be adapted to polish the pronunciation list generated by our model.

In particular, the training procedure for PMMs includes three steps. First, train a L2S model from a manually specified expert-pronunciation lexicon; second, generate a list of pronunciations for each word in the dataset using the L2S model; and finally, use an acoustic model to re-weight the pronunciations based on the acoustic scores of the spoken examples of each word.

To adapt this procedure for our purposes, we simply plug in the word pronunciations and the acoustic model generated by our model. Once we obtain the re-weighted lexicon, we re-generate forced

phone alignments and retrain the acoustic model, which can be utilized to repeat the PMM lexicon reweighting procedure. For our experiments, we iterate through this model refining process until the recognition performance converges.

5.2.2 Triphone Model

Conventionally, to train a context-dependent acoustic model, a list of questions based on the linguistic properties of phonetic units is required for growing decision tree classifiers (Young et al., 1994). However, such language-specific knowledge is not available for our training framework; therefore, our strategy is to compile a question list that treats each phonetic unit as a unique linguistic class. In other words, our approach to training a context-dependent acoustic model for the automatically discovered units is to let the decision trees grow fully based on acoustic evidence.

5.3 Baselines

We compare the recognizers trained by following the procedures described in Sec. 5.2 against three baselines. The first baseline is a grapheme-based speech recognizer. We follow the procedure described in Killer et al. (2003) and train a 3-state HMM for each grapheme, which we refer to as the *monophone grapheme* model. Furthermore, we create a *singleton question set* (Killer et al., 2003), in which each grapheme is listed as a question, to train a *triphone grapheme* model. Note that to enforce better initial alignments between the graphemes and the speech data, we use a pre-trained acoustic model to identify the non-speech segments at the beginning and the end of each utterance before starting training the monophone grapheme model.

Our model jointly discovers the phonetic inventory and the L2S mapping rules from a set of transcribed data. An alternative of our approach is to learn the two latent structures sequentially. We follow the training procedure of Lee and Glass (2012) to learn a set of acoustic models from the speech data and use these acoustic models to generate a phone transcription for each utterance. The phone transcriptions along with the corresponding word transcriptions are fed as inputs to the L2S model proposed in Bisani and Ney (2008). A stochastic lexicon can be learned by applying the L2S model and the discovered acoustic models to PMM. This two-stage approach for training a speech recognizer without an expert lexicon is referred to as the *sequential model* in this paper.

unit(%)	Monophone
Our model	17.0
Oracle	13.8
Grapheme	32.7
Sequential model	31.4

Table 2: Word error rates generated by the four monophone recognizers described in Sec. 5.2 and Sec. 5.3 on the weather query corpus.

Finally, we compare our system against a recognizer trained from an *oracle* recognition system. We build the oracle recognizer on the same weather query corpus by following the procedure presented in McGraw et al. (2013). This oracle recognizer is then applied to generate forced-aligned phone transcriptions for the training utterances, from which we can build both monophone and triphone acoustic models. The expert-crafted lexicon used in the oracle recognizer is also used in this baseline. Note that for training the triphone model, we compose a singleton question list (Killer et al., 2003) that has every expert-defined phonetic unit as a question. We use this singleton question list instead of a more sophisticated one to ensure that this baseline and our system differ only in the acoustic model and the lexicon used to generate the initial phone transcriptions. We call this baseline the *oracle* baseline.

6 Results and Analysis

6.1 Monophone Systems

Table 2 shows the WERs produced by the four monophone recognizers described in Sec. 5.2 and Sec. 5.3. It can be seen that our model outperforms the grapheme and the sequential model baselines significantly while approaching the performance of the supervised oracle baseline. The improvement over the sequential baseline demonstrates the strength of the proposed joint learning framework. More specifically, unlike the sequential baseline, in which the acoustic units are discovered independently from the text data, our model is able to exploit the L2S mapping constraints provided by the word transcriptions to cluster speech segments.

By comparing our model to the grapheme baseline, we can see the advantage of modeling the pronunciations of a letter using a mixture model, especially for a language like English which has many pronunciation irregularities. However, even for languages with straightforward pronunciation rules, the concept of modeling letter pronunciations using mixture models still applies. The main difference is that the mixture weights for letters of languages with simple pronunciation rules will be sparser and spikier. In other words, in theory, our model should always perform comparable to, if not better than, grapheme recognizers.

Last but not least, the recognizer trained with the automatically induced lexicon performs similarly to the recognizer initialized by an oracle recognition system, which demonstrates the effectiveness of the proposed model for discovering the phonetic inventory and a pronunciation lexicon from an annotated corpus. In the next section, we provide some insights into the quality of the learned lexicon and into what could have caused the performance gap between our model and the conventionally trained recognizer.

6.2 Pronunciation Entropy

The major difference between the recognizer that is trained by using our model and the recognizer that is seeded by an oracle recognition system is that the former uses an automatically discovered lexicon, while the latter exploits an expert-defined pronunciation dictionary. In order to quantify, as well as to gain insights into, the difference between these two lexicons, we define the average pronunciation entropy, $\hat{H}$, of a lexicon as follows.

$$\hat{H} \equiv \frac{-1}{|V|} \sum_{w \in V} \sum_{b \in \mathcal{B}(w)} p(b) \log p(b) \qquad (15)$$

where V denotes the vocabulary of a lexicon, $\mathcal{B}(w)$ represents the set of pronunciations of a word w and $p(b)$ stands for the weight of a certain pronunciation b. Intuitively, we can regard $\hat{H}$ as an indicator of how much pronunciation variation that each word in a lexicon has on average. Table 3 shows that the $\hat{H}$ values of the lexicon induced by our model and the expert-defined lexicon as well as their respective PMM-refined versions[4]. In Table 3, we can see that the automatically-discovered lexicon and its PMM-reweighted versions have much higher $\hat{H}$ values than their expert-defined counterparts. These higher $\hat{H}$ values imply that the lexicon induced by our model contains more pronunciation variation than the expert-defined lexicon. Therefore, the lattices constructed during the decoding process for our recognizer tend to be larger than those constructed for the oracle baseline, which explains the performance gap between the two systems in Table 2 and Table 3.

Our model (Discovered lexicon)	*PMM iterations*		
	0	1	2
$\hat{H}$ (bit)	4.58	3.47	3.03
WER (%)	17.0	16.6	15.9
Oracle (Expert lexicon)	*PMM iterations*		
	0	1	2
$\hat{H}$ (bit)	0.69	0.90	0.92
WER (%)	13.8	12.8	12.4

Table 3: The upper-half of the table shows the average pronunciation entropies, $\hat{H}$, of the lexicons induced by our model and refined by PMM as well as the WERs of the monophone recognizers built with the corresponding lexicons for the weather query corpus. The definition of $\hat{H}$ can be found in Sec. 6.2. The first row of the lower-half of the table lists the average pronunciation entropies, $\hat{H}$, of the expert-defined lexicon and the lexicons generated and weighted by the L2P-PMM framework described in McGraw et al. (2013). The second row of the lower-half of the table shows the WERs of the recognizers that are trained with the expert-lexicon and its PMM-refined versions.

As shown in Table 3, even though the lexicon induced by our model is noisier than the expert-defined dictionary, the PMM retraining framework consistently refines the induced lexicon and improves the performance of the recognizers[5]. To the best of our knowledge, we are the first to apply PMM to lexicons that are created by a fully unsu-

[4]We build the PMM-refined version of the expert-defined lexicon by following the L2P-PMM framework described in McGraw et al. (2013).

[5]The recognition results all converge in $2 \sim 3$ PMM retraining iterations.

pronunciations	*pronunciation probabilities*		
	Our model	1 PMM	2 PMM
93 56 87 39 19	0.125	-	-
93 56 61 87 73 99	0.125	-	-
11 56 61 87 73 99	0.125	0.400	0.419
93 20 75 87 17 27 52	0.125	0.125	0.124
55 93 56 61 87 73 84 19	0.125	0.220	0.210
93 26 61 87 49	0.125	0.128	0.140
63 83 86 87 73 53 19	0.125	-	-
93 26 61 87 61	0.125	0.127	0.107

Table 4: Pronunciation lists of the word *Burma* produced by our model and refined by PMM after 1 and 2 iterations.

pervised method. Therefore, in this paper, we provide further analysis on how PMM helps enhance the performance of our model.

We compare the pronunciation lists for the word *Burma* generated by our model and refined iteratively by PMM in Table 4. The first column of Table 4 shows all the pronunciations of *Burma* discovered by our model, to which our model assigns equal probabilities to create a stochastic list[6]. As demonstrated in the third and the fourth columns of Table 4, the PMM framework is able to iteratively re-distribute the pronunciation weights and filter out less-likely pronunciations, which effectively reduces both the size and the entropy of the stochastic lexicon generated by our model. The benefits of using the PMM to refine the induced lexicon are twofold. First, the search space constructed during the recognition decoding process with the refined lexicon is more constrained, which is the main reason why the PMM is capable of improving the performance of the monophone recognizer that is trained with the output of our model. Secondly, and more importantly, the refined lexicon can greatly reduce the size of the FST built for the triphone recognizer of our model. These two observations illustrate why the PMM framework can be an useful tool for enhancing the lexicon discovered automatically by our model.

6.3 Triphone Systems

The best monophone systems of the grapheme baseline, the oracle baseline and our model are used to generate forced-aligned phone transcriptions, which are used to train the triphone models described in Sec. 5.2.2 and Sec. 5.3. Table 5 shows the WERs of the triphone recognition systems. Note that if a more conventional question list, for example, a list that contains rules to classify phones into different broad classes, is used to build the oracle triphone system, the WER can be reduced to 6.5%. However, as mentioned earlier, in order to gain insights into the quality of the induced lexicon and the discovered phonetic set, we compare our model against an oracle triphone system that is built by using a singleton question set.

Unit(%)	Triphone
Our model	13.4
Oracle	10.0
Grapheme	15.7

Table 5: Word error rates of the triphone recognizers. The triphone recognizers are all built by using the phone transcriptions generated by their best monohpone system. For the oracle initialized baseline and for our model, the PMM-refined lexicons are used to build the triphone recognizers.

By comparing Table 2 and Table 5, we can see that the grapheme triphone improves by a large margin compared to its monophone counterpart, which is consistent with the results reported in (Killer et al., 2003). However, even though the grapheme baseline achieves a great performance gain with context-dependent acoustic models, the recognizer trained using the lexicon learned by our model and subsequently refined by PMM still outperforms the grapheme baseline. The consistently better performance our model achieves over the grapheme baseline demonstrates the strength of modeling the pronunciation of each letter with a mixture model that is presented in this paper.

Last but not least, by comparing Table 2 and Table 5, it can be seen that the relative performance gain achieved by our model is similar to that obtained by the oracle baseline. Both Table 2 and Table 5 show that even without exploiting any language-specific knowledge during training, our recognizer is able to perform comparably with the recognizer trained using an expert lexicon. The ability of our model to obtain such similar performance

[6]It is also possible to assign probabilities proportional to the decoding scores of the word tokens.

further supports the effectiveness of the joint learning framework proposed in this paper for discovering the phonetic inventory and the word pronunciations from simply an annotated speech corpus.

7 Conclusion

We present a hierarchical Bayesian model for simultaneously discovering acoustic units and learning word pronunciations from transcribed spoken utterances. Both monophone and triphone recognizers can be built on the discovered acoustic units and the inferred lexicon. The recognizers trained with the proposed unsupervised method consistently outperforms grapheme-based recognizers and approach the performance of recognizers trained with expert-defined lexicons. In the future, we plan to apply this technology to develop ASRs for more languages.

Acknowledgements

The authors would like to thank Ian McGraw and Ekapol Chuangsuwanich for their advice on the PMM and recognition experiments presented in this paper. Thanks to the anonymous reviewers for helpful comments. Finally, the authors would like to thank Stephen Shum for proofreading and editing the early drafts of this paper.

References

Michiel Bacchiani and Mari Ostendorf. 1999. Joint lexicon, acoustic unit inventory and model design. *Speech Communication*, 29:99 – 114.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451, May.

Steven B. Davis and Paul Mermelstein. 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 28(4):357–366.

Toshiaki Fukada, Michiel Bacchiani, Kuldip Paliwal, and Yoshinori Sagisaka. 1996. Speech recognition based on acoustically derived segment units. In *Proceedings of ICSLP*, pages 1077 – 1080.

Alvin Garcia and Herbert Gish. 2006. Keyword spotting of arbitrary words using minimal speech resources. In *Proceedings of ICASSP*, pages 949–952.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2004. *Bayesian Data Analysis*. Texts in Statistical Science. Chapman & Hall/CRC, second edition.

James Glass. 2003. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17:137 – 152.

Aren Jansen and Kenneth Church. 2011. Towards unsupervised training of speaker independent acoustic models. In *Proceedings of INTERSPEECH*, pages 1693 – 1696.

Frederick Jelinek. 1976. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64:532 – 556.

Matthew J. Johnson and Alan S. Willsky. 2013. Bayesian nonparametric hidden semi-markov models. *Journal of Machine Learning Research*, 14:673–701, February.

Mirjam Killer, Sebastian Stüker, and Tanja Schultz. 2003. Grapheme based speech recognition. In *Proceeding of the Eurospeech*, pages 3141–3144.

Chia-ying Lee and James Glass. 2012. A nonparametric Bayesian approach to acoustic model discovery. In *Proceedings of ACL*, pages 40–49.

Chin-Hui Lee, Frank Soong, and Biing-Hwang Juang. 1988. A segment model based approach to speech recognition. In *Proceedings of ICASSP*, pages 501–504.

Ian McGraw, Ibrahim Badr, and James Glass. 2013. Learning lexicons from speech using a pronunciation mixture model. *IEEE Trans. on Speech and Audio Processing*, 21(2):357–366.

Kevin P. Murphy. 2002. Hidden semi-Markov models (hsmms). Technical report, University of British Columbia.

Kuldip Paliwal. 1990. Lexicon-building methods for an acoustic sub-word based speech recognizer. In *Proceedings of ICASSP*, pages 729–732.

Man-hung Siu, Herbert Gish, Arthur Chan, William Belfield, and Steve Lowe. 2013. Unsupervised training of an HMM-based self-organizing unit recgonizer with applications to topic classification and keyword discovery. *Computer, Speech, and Language*.

Sebastian Stüker and Tanja Schultz. 2004. A grapheme based speech recognition system for Russian. In *Proceedings of the 9th Conference Speech and Computer*.

Balakrishnan Varadarajan, Sanjeev Khudanpur, and Emmanuel Dupoux. 2008. Unsupervised learning of acoustic sub-word units. In *Proceedings of ACL-08: HLT, Short Papers*, pages 165–168.

Steve J. Young, J.J. Odell, and Philip C. Woodland. 1994. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of HLT*, pages 307–312.

Victor Zue, Stephanie Seneff, James Glass, Joseph Polifroni, Christine Pao, Timothy J. Hazen, and Lee Hetherington. 2000. Jupiter: A telephone-based conversational interface for weather information. *IEEE Trans. on Speech and Audio Processing*, 8:85–96.