Overcoming Resource Limitations in the Processing of Unlimited Speech: Applications to Speaker and Language Recognition

by

Stephen H. Shum

B.S., University of California, Berkeley (2009)
S.M., Massachusetts Institute of Technology (2011)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

Author . . . .
Department of Electrical Engineering and Computer Science
May 20, 2016

Certified by . . . .
James R. Glass
Senior Research Scientist
Thesis Supervisor

Certified by . . . .
Najim Dehak
Assistant Professor, Johns Hopkins University
Thesis Supervisor

Accepted by . . . .
Professor Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Overcoming Resource Limitations in the Processing of Unlimited Speech: Applications to Speaker and Language Recognition

by

Stephen H. Shum

Submitted to the
Department of Electrical Engineering and Computer Science
on May 20, 2016,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

We live an era with almost unlimited access to data. Yet without their proper tagging and annotation, we often struggle to make effective use of most of it. And sometimes, the labels we have access to are not even the ones we really need for the task at hand. Asking human experts for input can be time-consuming and expensive, thus bringing to bear a need for better ways to handle and process unlabeled data.

In particular, successful methods in unsupervised domain adaptation can automatically recognize and adapt existing algorithms to systematic changes in the input. Furthermore, methods that can organize incoming streams of information can allow us to derive insights with minimal manual labeling effort – this is the notion of weakly supervised learning.

In this thesis, we explore these two themes in the context of speaker and language recognition. First, we consider the problem of adapting an existing algorithm for speaker recognition to a systematic change in our input domain. Then we undertake the scenario in which we start with only unlabeled data and are allowed to select a subset of examples to be labeled, with the goal of minimizing the number of actively labeled examples needed to achieve acceptable speaker recognition performance. Turning to language recognition, we aim to decrease our reliance on transcribed speech via the use of a large-scale model for discovering sub-word units from multilingual data in an unsupervised manner. In doing so, we observe the impact of even small bits of linguistic knowledge and use this as inspiration to improve our sub-word unit discovery methods via the use of weak, pronunciation-equivalent constraints.

Thesis Supervisor: James R. Glass
Title: Senior Research Scientist

Thesis Supervisor: Najim Dehak
Title: Assistant Professor, Johns Hopkins University

Acknowledgments

I've said this before, and I will say it again: When it comes to research advisers, I do not think I could have found a better combination than Jim and Najim. With Jim's calm, steady hand and Najim's explosive flurry of ideas, the two styles managed to regularize each other, and I think I hit the jackpot in terms of finding a balance between them. Jim, I hope we can find the time to ride bikes again soon. Najim, I look forward to catching up with you over some soft-shell crab in Baltimore.

Furthermore, I could not have done any of this without the help of a wonderful thesis committee. Victor, thank you for your continued encouragement, for helping me keep in mind the bigger picture, and for shaping so much of the identity of this thesis. Tommi, thank you for your endless insights into the rigors of machine learning; I really appreciate your helping me think outside of the box in all of the problems that I faced. Doug, your advice and perspective has proven invaluable through the years. It has been an absolute privilege working on a number of different projects with you; I have learned so much through your guidance and witnessing your pragmatic approach to research and development.

To past and present members of the Spoken Language Systems group – Ann, Carrie, Chen, Daniel, Dave, Eann, Ekapol, Felix, Henry, Hung-An, Ian, Jackie, Jen, Jingjing, Lee, Mandy, Marcia, Michael, Mitch, Mitra, Scott, Sree, Stephanie, Timo, Tuka, Wei-Ning, William, Xue, Yaodong, Yonatan, Yu, Yushi – thank you for being such a joy to work alongside. Special thanks to Daniel, Dave, and Tuka for bearing with the mess I make in our office. And of course, thank you, Marcia! Without you, I think a lot of our group's operations would grind to a halt. For all that you do to keep our worlds running smoothly – from birthdays to travel to correcting my poor grammar to scheduling to providing words of encouragement when we need it most – thank you so, so much.

Many thanks to the MIT Lincoln Laboratory for their continued support and sponsorship throughout the course of my Ph.D. I really appreciate the freedom you have given me to explore different research topics; it's been an extremely rewarding process. A massive thank you to Bill (and Doug R.) for a fantastic summer internship in 2012, as well as to Fred for all of his help in the development of the work in this thesis. Furthermore, it's been a pleasure working and interacting regularly with Doug S., Elliot, Joe, Pedro, and Tom.

Thank you to the JHU HLTCOE for an excellent SCALE 2013, especially Alan, Daniel, and Chuck for all of their help and guidance in shaping one of the major portions of this thesis. A shout-out as well to Colleen, Eli, Heather, Lisa, Sean, Walker, and everyone else for making an abnormally warm Baltimore summer that much more bearable.

I've had the pleasure of working and interacting with an amazing research group at the Brno University of Technology, including Honza, Lukas, Mehdi, Mireia, Olda, Ondrej, Pavel, among others. Thank you for your hospitality.

I owe much gratitude to Arlo, Elmar, Niko, Jason, and Sanjeev for being incredible research mentors since the day I began doing research in the field. Thank you, also, to Brian and Helen for hosting visits for me while I was in Hong Kong. Your insights and leadership will never be forgotten.

I've been fortunate enough to have been a part of a number of fantastic groups during my time here at MIT, most notably the MIT Cycling Team and the MIT Strong group, who made my first Boston Marathon experience so extremely meaningful and memorable. I'm also indebted to Abby, Jackie, and Michelle, for convincing me to dust off my cello and join them for a few semesters of chamber music; playing with you all was always an extremely welcome break from trying to finish this thesis!

I started graduate school here in 2009 and was lucky enough to make some amazing friends. Between graduations and relocations, the subset of us that are still in the area will always be evolving, but as a core, we'll always be growing. Thank you to Aaron, Angele, Bev, Chang, Christin, Dave S., Ian S., Janet, Jen W., Justin, Matthieu, Mehul, Michael P., Morgan, Nick H., Phil, Sam, Seoyoung, for always being a group I can always count on to put a smile on my face.

One of the first things I did upon my move to the Boston area was to find fellow Cal alumni with whom to connect and watch college football. While our team may still be in search of consistency on the field, I've always been able to count on Ben, Chad, Eliza, Geoff, Joey, Matt, Michael C., Sarah, Toshiki, Tristan, and everyone else in the Cal Alumni Club of New England. Go Bears!

Judy, William, and Carrie, thanks for being such a consistent source of friendship and fun; I truly struggle to recall instances without you in which I have laughed harder. Chian, I'm so glad we got to reconnect after so many years! Frances, thank you for always finding new ways to make fun of me; isn't life so much easier when we don't take it too seriously? Dave H., in addition to your incredible insight and understanding of all things research- and life-related, thank you for being a consistent source of deliciously brewed beverages – I'm not sure I would have made it this far without them. Ekapol, it's been a pleasure and an honor beginning and ending our respective Ph.D. experiences together here at SLS – I believe this makes us brothers in some sense! It'll be sad to see you move back to Thailand, but I look forward to visiting soon.

John and Michael A., I am honored to have the opportunity to graduate in the cap and gown worn by such esteemed predecessors. And I really look forward to a day in which we are all in the same geographic location.

A lot of my time as a graduate student was spent learning about myself through participation in a number of endurance sports, and I couldn't have done it without the camaraderie of, support from, and shared suffering with Jen S., Alicia, Devin, Jimmy, Michael A., Nick L., Pavel, and Scott. I look forward to continuing our journey together with more epic runs, rides, races, and explorations into the unknown.

Eli, words cannot express the beauty and joy you've added to my life. Thank you for helping me find myself, for guiding me through the darker times, and for urging me to the finish line. It's been an amazing ride, and I can't wait for our next adventure.

My final thanks goes to Mom and Dad, whose endless love, support, and encouragement I truly could not have made it this far without. Thank you for always being there for me; thank you for taking the time to attend my thesis defense; and thank you for being among the first to read through this thesis! For all that and much, much more, I dedicate this thesis to you. Thank you for everything.

Contents

List of Figures

List of Tables

List of Abbreviations

AC	Across Class
ACP	Average Cluster Purity
AD-LDA	Approximate Distributed Latent Dirichlet Allocation
AHC	Agglomerative Hierarchical Clustering
ASF	Average Speaker Fragmentation
ASR	Automatic Speech Recognition
AUD	Acoustic Unit Discovery
BN	Bottleneck
BUT	Brno University of Technology
DCF	Detection Cost Function
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
DP	Dirichlet Process
DTW	Dynamic Time Warp
EER	Equal Error Rate
EM	Expectation-Maximization
FA	False Alarm
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
IR	Information Retrieval

LDA	Linear Discriminant Analysis
LDC	Linguistic Data Consortium
LLR	Log Likelihood Ratio
LR	Logistic Regression
LRE	Language Recognition Evaluation
LRE11	2011 NIST LRE
LRE15	2015 NIST LRE
MAP	Maximium a Posteriori
MCL	Markov Clustering
MFCC	Mel-Frequency Cepstral Coefficient
ML	Maximum Likelihood
MMR	Maximal Marginal Relevance
NAP	Nuisance Attribute Projection
NIST	National Institute of Standards and Technology
NMI	Normalized Mutual Information
PCA	Principal Components Analysis
PE-UER	Pronunciation-Equivalent Unit Error Rate
PLDA	Probabilistic Linear Discriminant Analysis
PPR-LM	Parallel PR-LM
PR-LM	Phoneme Recognition followed by Language Modeling
SD	Speaker Dependent
SDC	Shifted Delta Cepstral
SI	Speaker Independent
SMM	Subspace Multinomial Model
SnGM	Subspace n-gram Model
SRE	Speaker Recognition Evaluation
SRE10	2010 NIST SRE
SVM	Support Vector Machine
SWB	Switchboard
UBM	Universal Background Model
UPGMA	Unweighted Pair Group Method with Arithmetic Mean

vMF	von Mises-Fisher (distribution)
WC	Within Class
WCCN	Within-Class Covariance Normalization
WER	Word Error Rate

Chapter 1

Introduction

With the ubiquity, connectivity, and expansive storage of data-recording devices (smart phones, embedded sensors, etc.), we live in an era with almost unlimited access to data. Nevertheless, we often struggle to make sense – much less make effective use – of most of it. One major reason for this is that a lot of the data that we have such convenient access to are unlabeled and, thus, useless in many of the traditionally supervised machine learning scenarios that require explicit labeled examples. Furthermore, sometimes the labels that we do have access to are not necessarily the labels that we really need for the task at hand. Because the process of using humans to tag and annotate can be expensive and time-consuming, we'd like to develop methods that utilize existing, previously labeled examples in ways that can make use of the many unlabeled examples at hand.

For many years now, the speech research community has dreamt of organic spoken language systems that grow and improve without continued expert supervision [6]. While such systems are closer than ever to becoming a reality, it is still common for users to simply learn to live with a system's errors. That is, despite being deployed in a dynamic environment, the underlying models of a spoken language system are often fixed [7]; each interface knows what it knows and doesn't know what it doesn't [6]. As such, when recording conditions change from a quiet meeting room to a bustling street corner, the typical system not only continues to operate under the former recording condition but also fails to recognize that its resulting outputs are subject to corruption from the additional noise. Such lack of awareness and inability to adapt to ever-changing scenarios are what prevent us from building machines that can truly hear [8]. Instead what we really need is the ability to build systems that behave like living organisms that can learn, grow, reconfigure, and repair themselves as necessary [6].

As one possible scenario, we envision a robot deployed into the real world equipped with the necessary computational capabilities and an ability to perceive audio, but not necessarily the training to fully understand it. We task the system with learning how to make sense of its environment and allow the robot to roam around freely and collect information in some reasonable fashion. In the beginning, most of its assimilation will come in an unsupervised manner – that is, no one will be there to label out what it is the robot is hearing for every moment of its existence – although later we can perhaps incorporate some interaction and allow the system to ask some limited number of questions as a way for it to improve its understanding about the world it inhabits. The ultimate result would be a robot that learns to make sense of its audio environment without constant supervision, realizes the aspects of sound that it knows and doesn't know, and is able to ask for additional information and clarification in a sensible manner. This is the sort of system that we strive for, a true specimen of artificial intelligence that exists and operates in organic fashion.

1.1 Tasks

Our aim in this thesis is to develop methods for speaker and language recognition that better utilize unlabeled data. The objective in *speaker verification* is to determine whether or not a given test utterance originates from a hypothesized speaker, while the goal of *language identification* is to determine, from a known set of target languages, the spoken language of a given test utterance. In contrast to spoken content (i.e., speech recognition), which varies far more rapidly over time, we consider both speaker and language identity to be more *persistent* properties of the speech signal. As we will see in Chapter 2, state-of-the-art approaches to both problems involve a very similar representation of the audio.

1.2 Themes

In the context of speaker and language recognition, we explore two main themes: *unsupervised domain adaptation* and learning from *weak supervision.*

1.2.1 Domain adaptation

To illustrate the need for domain adaptation, we imagine one or more students who have learned a lot from school and are looking to continue their education. To do so, they are tasked with performing well on some sort of a college entrance exam (e.g., Scholastic Aptitude Test (SAT) or any of the Advanced Placement (AP) exams, et cetera). One naive, baseline strategy is for these students, armed only with their respective knowledge obtained from school, to simply show up at the test center and take the exam without any exam-specific preparation. Another strategy that tends to yield better results is for each student to augment his or her school-based knowledge and prepare using an exam-specific study guide and maybe a set of practice tests. By working through practice problems that are similar to those found in the exam, our students might become more familiar with the testing environment and transfer their respective school-based knowledge into a form that also improves exam performance. This sort of transfer learning is what we call domain adaptation.

While effective, however, this strategy is not entirely scalable. Exams, such as the SAT, change regularly. And every time there is such an update, our students will need a new, corresponding study guide – this gets expensive! In this analogy, our students should be thought of as a set of one or more algorithms trained initially on knowledge from school and then tasked with performing well on a separate exam. By using a study guide and practice tests corresponding to the most updated exam, they are undergoing a process known as *supervised* domain adaptation.

Scalability is extremely important in many practical engineering scenarios. In this world, we have over 7000 spoken languages, 400 of which have at least one million speakers.[1] Even under a generous estimate, we have developed reasonable speech technologies for fewer than 100 languages. (This thesis involves no more than 30 different languages.) So we have a lot of work to do!

In this thesis, we are interested in the problem of *unsupervised* domain adaptation, which can be thought of as access to an unlimited number of test preparation resources, but **no** access to their corresponding answer keys. In this sense, our students can take as many practice tests as they would like, but they will not be provided any feedback on their performance. We are interested in this problem because tests and evaluation conditions change regularly, and we'd like to be able to adapt to these changes without needing a new study guide and corresponding answer key every time.

More formally, we would like to learn from some source data distribution (e.g., school) a model that can perform well on a different, but related, target data distribution (e.g., SAT) [9]. In the unsupervised case, we assume that our learning sample contains labeled source examples (e.g., textbooks) and unlabeled target examples (e.g., practice tests without answers). In Chapter 3, we outline a task along these lines in the context of speaker verification, while Chapter 5 entertains a variant of this scenario for language identification.

[1] Taken from https://www.ethnologue.com/statistics/size.

1.2.2 Weak supervision

Learning from weak supervision touches upon the broad range of methods that are neither fully supervised or fully unsupervised [10]. In this thesis, we specifically consider two such aspects of this paradigm: *active learning* and *top-down equivalence constraints.*

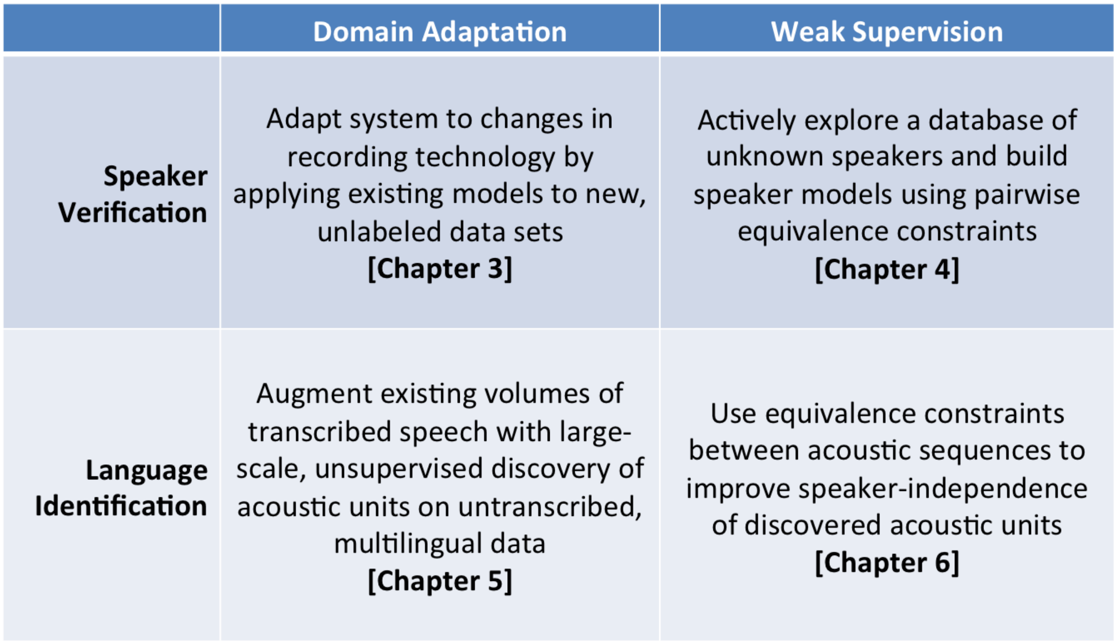
Active learning

Taken directly from [11], the key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. In situations where unlabeled data is abundant but manual labeling is expensive, an active learning system will choose what queries to pose, usually in the form of unlabeled data instances to be labeled by some oracle (e.g., a human annotator). Using these newly labeled instances, the system builds a model and, using this model, either applies it to the task at hand or goes back to the set of unlabeled data to find more examples that it might want to have labeled. In Chapter 4, we develop strategies to actively explore a database of unknown speakers and build speaker models for speaker verification using queries between pairs of utterances.

Top-down equivalence constraints

In the section above, we pose pairwise queries in the following way: "Are utterances A and B spoken by the same speaker?" The answer to such a query yields either a positive or negative equivalence constraint. While having these sorts of constraints are not quite the same as having fully labeled data, having enough of these constraints can still provide key bits of information that we can exploit to build our models for speaker recognition. The motivation for work along this front is in an effort to alleviate the need for expensive, expert-level knowledge. That is, instead of having to ask linguistic experts to provide for us a pronunciation dictionary for a new language, suppose our systems could simply organize the data in such a way that allows us to ask informative questions about the language to fluent, but non-linguist, speakers. The ability to involve a significantly broader population would be a huge step towards scaling speech technology to all the languages of the world. In addition to the pairwise equivalence constraints utilized in Chapter 4, we explore in Chapter 6 the use of equivalence constraints between acoustic sequences to improve the speaker-independence of discovered acoustic units.

1.3 Overview of contributions



	Domain Adaptation	Weak Supervision
Speaker Verification	Adapt system to changes in recording technology by applying existing models to new, unlabeled data sets **[Chapter 3]**	Actively explore a database of unknown speakers and build speaker models using pairwise equivalence constraints **[Chapter 4]**
Language Identification	Augment existing volumes of transcribed speech with large-scale, unsupervised discovery of acoustic units on untranscribed, multilingual data **[Chapter 5]**	Use equivalence constraints between acoustic sequences to improve speaker-independence of discovered acoustic units **[Chapter 6]**

Figure 1-1: *Overview of thesis contributions. We cover domain adaptation for speaker verification (top left) in Chapter 3, while Chapter 4 considers the problem of active learning and speaker recognition (top right). We apply large-scale unsupervised acoustic unit discovery to the language identification problem (bottom left) in Chapter 5, while Chapter 6 investigates methods to improve acoustic unit discovery using equivalence constraints (bottom right).*

Figure 1-1 provides an overview of our contributions in this thesis, which we also summarize in the following.

- In Chapter 3, we motivate and define a task for domain adaptation in speaker recognition, in which both in-domain and out-of-domain data are available, but labels are provided only for the mismatched, out-of-domain data. Our proposed approach, which utilizes both agglomerative and graph clustering techniques, achieves performance within 15% of a system that has access to all speaker labels. Along the way, we observe that both an imperfect clustering and an imprecise estimate of the number of speakers are forgivable in the presence of adaptation between hyper-parameters derived from in-domain and out-of-domain data.

- We subsequently consider a different form of resource constraints in Chapter 4, where we have matched, in-domain training data but no labels of any sort. In this exploration, we develop a greedy algorithm based on active learning that obtains speaker labels in the form of binary pairwise comparisons. We find that the actual number of pairwise comparisons needed to obtain state-of-the-art results is a mere fraction of the queries required to fully label an entire training set of utterances and is in line with the sample complexity guarantees provided by a more formal analysis of active learning.

- In Chapter 5, we return to the theme of domain adaptation and address the problem of language identification. Previously, state-of-the-art results leaned heavily on the use of transcribed speech in the languages we hope to recognize. We remove the need for such transcriptions via the use of an unsupervised method for discovering acoustic units. To do so, we modify an existing, small-scale Bayesian nonparametric model in ways that allow it to scale computationally to hundreds of hours of multilingual data. The resulting acoustic units can then be used to obtain neural network-based features for language recognition.

- In validating the ability of our acoustic unit discovery to generalize to different datasets and languages, we observe the continued importance of both language specificity and an improved acoustic representation of speech. Not surprisingly, we find that restricting our unit discovery to a particular family of languages – e.g., the various dialects of Arabic – yields a system that performs best on that specific subset of languages. We also find that supervised knowledge of even just one language can help substantially in discovering linguistically meaningful acoustic units and improving language recognition performance.

- In light of these observations, we explore ways to improve our acoustic unit discovery methods in Chapter 6 by incorporating weak pronunciation-equivalent constraints. While we still have no access to explicit textual transcriptions, we assume that we are given subsets of utterances that contain equivalent pronunciations of the same sentence (or word, or phrase) spoken by various speakers. To this end, we hope to use these constraints to encourage our model to learn similar unit sequences within these sets of utterances and, thus, arrive at a segmentation and clustering at the sub-word unit level that is more linguistically meaningful.

Chapter 2

i-vectors for Speaker and Language Recognition

A speech signal conveys information at many levels. In addition to the spoken content itself, we can also ascertain the language spoken, the speaker's identity, and even the speaker's emotional state. Both speaker and language recognition can serve as a front-end that improves the performance of an automatic speech recognizer; in particular, we must know what language whose speech we aim to recognize, and knowledge of the speaker's identity can help the recognizer adapt its models accordingly.

This thesis explores both speaker verification and language identification. While we loosely refer to these as "recognition" tasks throughout this thesis and corresponding literature, they are slightly different and we formally define them here. In *speaker verification*, we are given two speech utterances and asked to determine whether or not they were spoken by the same speaker [2]. *Language identification* is the task of automatically determining, from some list of known target languages, the language of a spoken utterance [3]. We should note that the former is an open-set verification problem in which our answer is either "yes" or "no." On the other hand, the latter is a closed-set identification problem in which we assume that the language of the test utterance must belong to one of a set of K known languages.

At the heart of our approach to both problems is the i-vector, a low-dimensional, vector-based representation of audio that easily allows the use of large amounts of previously collected and labeled audio to characterize and exploit both desired and nuisance directions of variability. Originally developed for speaker verification [12], the representation also obtained state-of-the-art results on language identification [13]. In this chapter, we provide an overview of how an i-vector can be extracted from a given utterance and used to represent the speaker or language within. Subsequently, we will compare and contrast techniques for both speaker and language recognition and provide some insights into what makes our approaches to each problem unique. This chapter will be by no means a comprehensive review of all of the techniques that have been explored, as the usage of i-vectors has grown enormously since their inception. Instead, we hope that it can serve as a primer for reference throughout the thesis.

2.1 Overview

In short, an i-vector can be seen as a low-dimensional summary of a particular utterance's distribution of acoustic features with respect to some existing background model [14]. Such a representation is useful because of its ability to transform a variable-length sequence of feature vectors into a single, fixed-length vector. We should note, of course, that this is useful only for properties of a signal that are assumed to be persistent throughout, such as speaker identity and language spoken. The more transient characteristics of the signal, such as the identity and order of phonemes spoken, are much more difficult to ascertain from an i-vector.

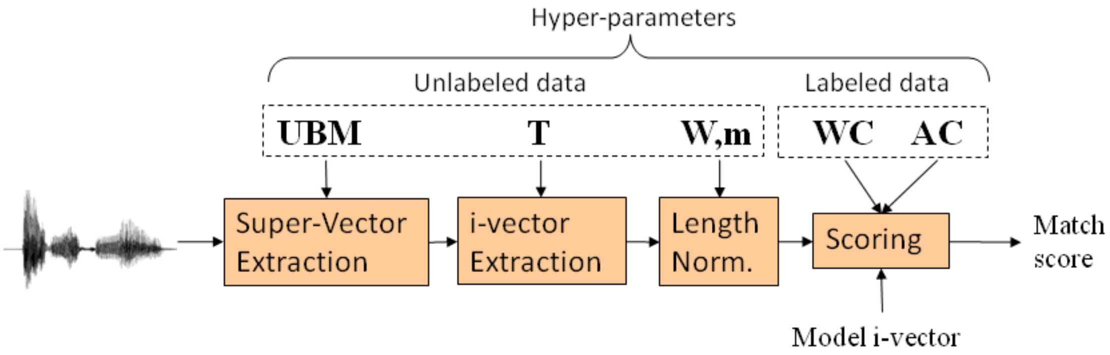


Figure 2-1: *High-level diagram of i-vector system for speaker recognition showing all hyper-parameters and which, respectively, require labeled and unlabeled data to train. (Taken from [1].)*

Figure 2-1 shows a high-level diagram of an i-vector system for speaker recognition; a system for language recognition follows the same general flow but has a few differences that we will highlight later in this chapter. In general, our goal is to ascertain how likely some given test utterance is to have originated from some model. In speaker recognition, the model might represent some target speaker whose identity we'd like to verify; in language recognition, we would have a model for each language we would like to identify. In the i-vector approach, both the model and the test utterance are represented as fixed-dimensional i-vectors.

2.2 Spectral Features

In this section, we provide an overview of the spectral features that are typically used for speaker and language recognition. A number of alternatives exist, many of which are beyond the scope of this thesis but are reviewed in [15]. Where appropriate, we will point out these variations throughout the text.

To obtain features for speech processing, we assume that the sampled speech waveform is approximately stationary over short intervals of approximately 10-30 ms. Taken from [2], Figure 2-2 provides an overview of the process, which involves a sliding analysis window (typically 25 ms in length) along the speech signal and computing a set of features every 10ms. For each window placement (typically Hamming), the higher frequences of the speech speech signal are amplified via a linear "pre-emphasis" filter and the discrete spectrum is computed using the fast Fourier transform (FFT).

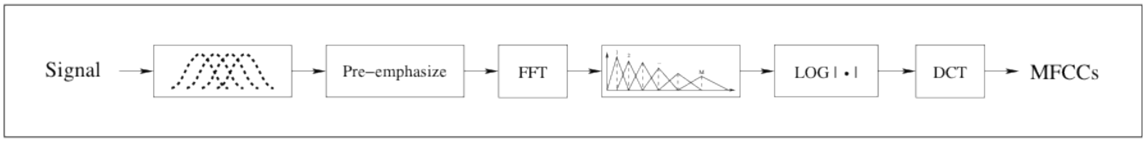


Figure 2-2: *Overview of MFCC extraction. (Taken from [2].)*

Introduced in the early 1980s for speech recognition, the so-called mel-frequency cepstral coefficients (MFCCs) are popular features in speech and audio processing [15]. MFCCs are computed from a spectral representation of the signal with the aid of a psychoacoustically motivated filterbank, followed by a logarithmic compression, which mimics a human's non-linear frequency resolution of loudness [3], and discrete cosine transform (DCT). Each of the filters (typically 24 for telephone speech) in the filterbank is triangular and computes the energy average around the center frequency of each triangle. The center frequences are linearly spaced on a mel-frequency scale, which was designed to approximate the behavior of the human auditory system. Finally, the DCT is applied to reduce the correlation between the filters. For speaker recognition, we use coefficients 1-19 as well as the log of the energy of the currently windowed signal (in lieu of the 0th coefficient), giving us a 20-dimensional feature vector. Finally, we typically incorporate some temporal information to the features through estimates of the first-and second-order temporal derivatives, known as delta and double-delta coefficients, respectively. This yields a final feature vector containing 60 dimensions for speaker recognition [2].

For language recognition, we derive a slightly different feature representation from MFCCs [3]. Instead of using first- and second-order temporal derivatives, another way to obtain information about temporal evolution is via shifted delta cepstral (SDC) features, which involve computing delta cepstra across multiple frames. SDC features are specified by four parameters: N, d, P, and k, where N is the number of cepstral coefficients computed at each frame, d defines the time shift between each delta computation, P is the time shift between the respective centers of consecutive blocks, and k is the number of blocks whose coefficients are stacked to form the final feature vector. A typical N-d-P-k configuration is 7-1-3-7, and Figure 2-3 illustrates the process for one cepstral coefficient. In particular, this one coefficient will yield $k = 7$ delta computations, each delta involves the coefficients $d = 1$ frame in front and behind some center frame, and each center frame is spaced $P = 3$ frames apart. In language recognition, we typically use coefficients 0-6 of the MFCCs, thus yielding a 49-dimensional feature vector [3].

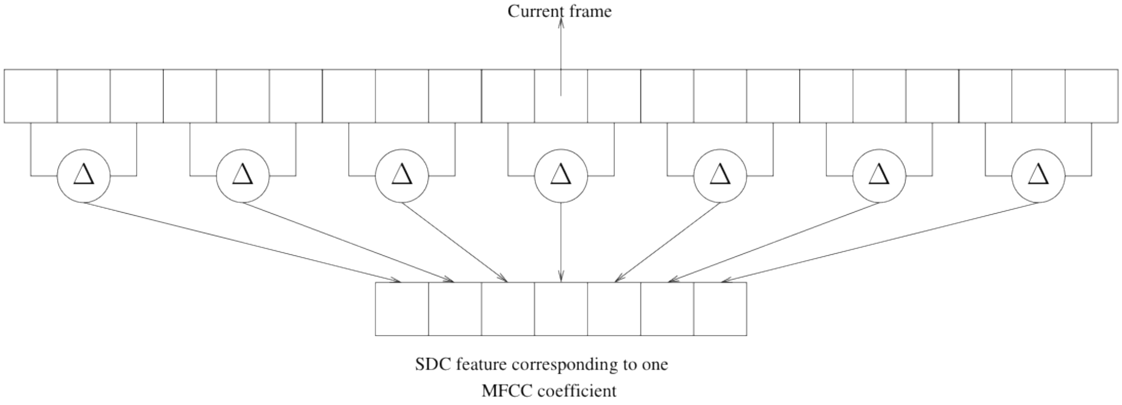


Figure 2-3: *An illustration of how we compute shifted delta cepstral (SDC) features for a single MFCC coefficient. (Taken from [3].)*

2.3 GMM-UBM-MAP

We begin by modeling feature vectors – either MFCC or SDC features for speaker or language recognition, respectively – as they relate to a Universal Background Model (UBM), which is typically a Gaussian Mixture Model (GMM) built to represent a speaker- and language-independent distribution of features [4]. The data used to build this UBM will be task-dependent; in general, using data that matches as closely as possible to the test data leads to better performance.[1] For example, in the ideal speaker recognition scenario, we would have data from thousands of speakers

[1] Of course, handling data mismatch is what this thesis is supposed to be all about!

each making over 10 calls from at least two different handsets [1]. And in the ideal language recognition scenario, we would have data from every language that we would ultimately want to be able to recognize.

A GMM is a generative model and can be seen as a semi-parametric probabilistic method that, given appropriate front-end features, adequately represents a speech signal and its variabilities [2]. Given a GMM, θ, consisting of C components and F-dimensional feature vectors, the likelihood of observing a given feature vector y is computed as

$$p(y|\theta) = \sum_{c=1}^{C} \pi_c \mathcal{N}_c(y|\mu_c, \Sigma_c) \tag{2.1}$$

where the sum of the mixture weights $\sum_c \pi_c = 1$, and $\mathcal{N}_c(y|\mu_c, \Sigma_c)$ is a multivariate Gaussian with F-dimensional mean vector μ_c, $F \times F$ covariance matrix Σ_c, and probability distribution function

$$\mathcal{N}_c(y|\mu_c, \Sigma_c) = \frac{1}{(2\pi)^{F/2}|\Sigma_c|^{1/2}} \exp\{-\frac{1}{2}(y-\mu_c)'\Sigma_c^{-1}(y-\mu_c)\}. \tag{2.2}$$

Though it is possible to use full covariances in the implementation, it is a bit more computationally efficient to use only diagonal covariances. The density modeling of a C-th order full covariance GMM can equally well be achieved using a diagonal covariance GMM of higher order (i.e. $C' > C$) [4]. Finally, to preserve our previously defined notation, we can denote $\theta = \{\theta_1, ..., \theta_C\}$, where $\theta_c = \{\pi_c, \mu_c, \Sigma_c\}$.

Given a sequence of feature vectors $u = \{y_1, y_2, ..., y_L\}$, we make the assumption that each observation vector is independent of the other observations [2]. As such, the likelihood of a given utterance u given the model θ is simply the product of the likelihood of each of the L frames. Because the multiplication of many probabilities on a machine can potentially lead to computational underflow, we instead use the log-likelihood in our computation as follows:

$$\log p(u|\theta) = \sum_{t=1}^{L} \log p(y_t|\theta) \tag{2.3}$$

Given a collection of feature vectors, the parameters of a GMM can be estimated via the Expectation-Maximization (EM) algorithm [16], which iteratively refines the GMM parameters to monotonically increase the likelihood of the estimated model for the observed feature vectors. Given a UBM, we can model a test utterance via a *maximum a posteriori* (MAP) adaptation of the original UBM and obtain a new set of parameters that characterizes the distribution of features in the test utterance. A more thorough explanation of the MAP adaptation process can be found in [17], but Figure 2-4 provides an illustration of adapting the mean and covariance parameters of the observed Gaussians. In practice, only the mean vectors μ_c, $c = 1, ..., C$ are adapted; updating the weights and covariance matrices does not significantly affect system performance [2].

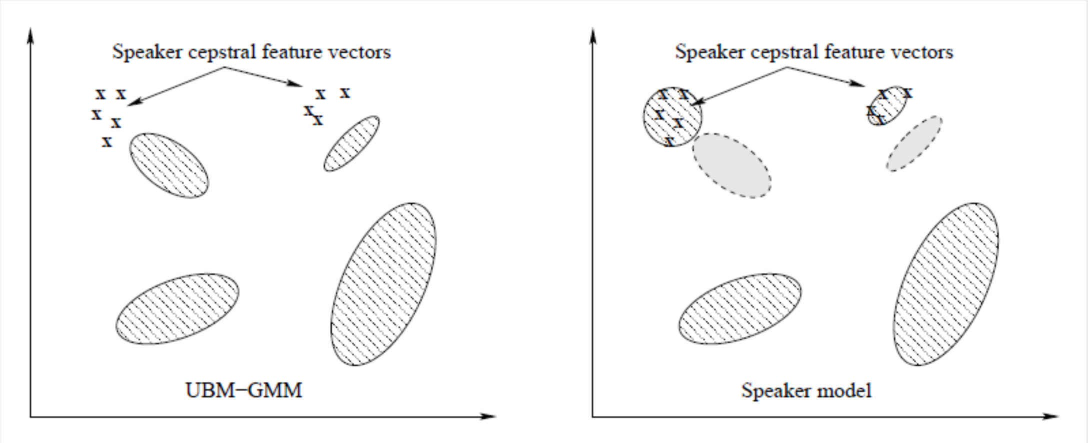

Speaker cepstral feature vectors
UBM−GMM
Speaker cepstral feature vectors
Speaker model

Figure 2-4: *Depiction of maximum a posteriori (MAP) adaptation, taken from [4, 2].*

2.4 Towards Factor Analysis

One way to parameterize a UBM with C mixture components in a feature space of dimension F is to view it as a single *supervector*, m, of dimension $C \cdot F$ along with a diagonal "super"-covariance matrix of dimension $CF \times CF$. This mean supervector is generated by concatenating all the Gaussian component means, while the covariance matrix is generated by respectively concatenating (along its diagonal) all the diagonal covariances of each mixture.

The idea behind the i-vector approach is that most of the variability between different supervectors can be captured within a subspace that is of far lower dimensionality than that of the original supervector, which can be on the order of 120,000 ($C = 2048$, $F \approx 60$). Intuitively, we can imagine applying principal components analysis (PCA) to the supervectors obtained from the MAP adaptation of the UBM to many different utterances; in practice, we use factor analysis and derive an EM algorithm to estimate the subspace, T, and residual, Σ. More formally, we represent our utterance-dependent supervector, M, as

$$M = m + Tw, \tag{2.4}$$

where m is the utterance-independent supervector taken from the UBM, T is a rectangular matrix of low rank, R, that defines our i-vector subspace, and w is an R-dimensional random vector with a standard normal prior distribution, $\mathcal{N}(0, I)$. We then define as our i-vector the MAP estimate of w given an observed utterance u. To compute this quantity, we estimate the posterior distribution, $w|u$, and use the mean of that resulting Gaussian distribution as the utterance's i-vector. Depending on its intended application, the rank of T and corresponding dimensionality of w typically ranges between 100 and 600; i.e., $R \in [100, 600]$.

2.4.1 The Posterior Distribution of w

A full explanation of the EM algorithm for training T can be found in [17]; here, we simply state how the posterior distribution, $w|u$, is estimated using Baum-Welch statistics from the UBM [12]. Assuming, as before, that our given utterance u is represented as a sequence of L frames $u = \{y_1, y_2, ..., y_L\}$, then the zeroth and first order Baum-Welch statistics are

$$N_c\left(u\right) = \sum_{t=1}^{L} P\left(c|y_t, \theta_{\text{UBM}}\right) = \sum_t \gamma_t(c) \tag{2.5}$$

$$F_c\left(u\right) = \sum_{t=1}^{L} P\left(c|y_t, \theta_{\text{UBM}}\right) y_t = \sum_t \gamma_t(c) \cdot y_t \tag{2.6}$$

where $c = 1, ..., C$ is the index of the corresponding Gaussian component and $\gamma_t(c) = P\left(c|y_t, \theta_{\text{UBM}}\right)$ corresponds to the posterior probability of mixture component c from our UBM generating the frame y_t.

To make our subsequent notation a bit simpler, let us define the centralized first order Baum-Welch statistics, $\tilde{F}_c(u)$, as

$$\tilde{F}_c(u) = F_c\left(u\right) - N_c\left(u\right) m_c = \sum_t \gamma_t(c) \cdot (y_t - m_c), \tag{2.7}$$

where m_c is the subvector corresponding to mixture component c of our supervector m. Lastly, let $N(u)$ be the $CF \times CF$ diagonal matrix whose diagonal blocks are $N_c(u) \cdot I$ $(c = 1, ..., C)$, and let $\tilde{F}(u)$ be the $CF \times 1$ supervector obtained by concatenating $\tilde{F}_c(u)$ $(c = 1, ..., C)$.

Then the posterior distribution, $w|u$, given the acoustic observations of an utterance u is Gaussian with mean $l^{-1}(u)T^{*}\Sigma^{-1}\tilde{F}(u)$ and covariance matrix $l^{-1}(u)$, where $l(u)$ is the $R \times R$ matrix

$$l(u) = I + T^{*}\Sigma^{-1}N(u)T, \tag{2.8}$$

and both T and Σ are estimated via the EM algorithm described in [17]. In running that EM algorithm, T is initialized randomly and Σ is initialized as the diagonal super-covariance matrix from the original UBM. Our i-vector, ultimately, is the mean of this posterior distribution; i.e., $\hat{w}_u = \mathbb{E}[w|u] = l^{-1}(u)T^*\Sigma^{-1}\tilde{F}(u)$. To simplify our notation throughout the rest of this chapter, we will use the term $w_u = \hat{w}_u$ to refer an i-vector – i.e., the MAP estimate – and not the prior or posterior distributions of w mentioned above.

2.5 Incorporating Labels and Scoring

As defined, the i-vector is simply a compressed representation of an utterance's distribution of feature vectors with respect to a UBM; as such, it contains information regarding the speaker, language, and recording environment. We can also see from Figure 2-1 that the process so far has not required any supervised knowledge regarding speaker or language labels pertaining to the utterances. This also implies that we can use as much data as we would like to build our UBM and train our i-vector subspace, T. But now the data has been projected into a lower dimensional space, standard pattern classification techniques can be applied to utilize existing speaker or language labels in a straightforward and computationally efficient manner [17].

Incorporating labels and computing scores is the step at which techniques for speaker and language recognition diverge somewhat. Recall that speaker recognition is the open-set verification task of determining whether two test utterances are spoken by the same speaker, which is a *two-class* recognition problem. Language recognition, on the other hand, is the closed-set identification problem of deciding, from some known list of targets, the language of a test utterance; this is inherently a *multi-class* recognition problem. While an exhaustive survey of the various backend recipes for both respective problems is beyond the scope of this overview, we outline in the following sub-sections the various approaches and evaluation metrics used in this thesis.

2.5.1 Speaker Verification

A common theme in i-vector-based approaches is that they tend to ignore the process by which i-vectors were extracted (i.e., the factor analysis) and instead pretend they were generated by some assumed generative model [18]. Under this paradigm, we further assume that an i-vector can be broken down into statistically independent speaker and channel[2] components, both of which are Gaussian distributions [18, 19]. While the latter assumption was shown to be untrue in [19], the work in [18] showed that whitening and then length-normalizing the entire set of i-vectors as a pre-processing step can help overcome this setback. Our explorations in Chapters 3 and 4 will adhere to this pre-processing step.

In Probabilistic Linear Discriminant Analysis (PLDA) [20], we assume that the i-vector, w_u of some utterance, u, can be decomposed as

$$w_u = \mu + Vx + \epsilon_u, \tag{2.9}$$

where $s = \mu + Vx$ is the speaker-specific part that reflects the identity of the speaker in u but does not depend on u, and ϵ_u is the utterance-dependent channel component.[3] Specifically, ϵ_u is assumed to be Gaussian with zero mean and full covariance W, while μ is a global offset (which should be zero after our pre-processing step), the columns of V provide a basis for a speaker-specific subspace, and x is a latent speaker identity vector that has a standard normal prior. The parameters $\{\mu, V, W\}$ can be estimated from a large collection of labeled training data in maximum likelihood fashion using an EM algorithm as described in [20]. Intuitively, we can view W as a within-speaker covariance matrix and $B = VV^*$ as the between-speaker covariance.

The task of determining whether two utterances – represented as i-vectors, w_1, w_2, – share the same latent speaker identity can be restated as a hypothesis test between H_S, where both w_1 and w_2 share the same speaker identity, and H_D, where the i-vectors were generated from different identities. This amounts to the computation of a log likelihood ratio (LLR), where

$$\text{LLR}(w_1, w_2) = \log \frac{P(w_1, w_2|H_S)}{P(w_1|H_D)P(w_2|H_D)} \tag{2.10}$$

Conveniently, our Gaussian PLDA formulation above yields a closed-form solution

[2] We let the term “channel” encompass all the other variabilities that we don’t wish to model, such as recording environment, speaker emotion, phonetic content, and even language identity. Conversely, in the language recognition setting, the “channel” component might include speaker identity.

[3] The version of PLDA presented in [20] assumes the channel component is $c = Uz + \epsilon_u$, where the columns of U provide a basis for a channel-specific subspace and ϵ_u has diagonal covariance; we follow the simplification of [18, 19] to make $c = \epsilon_u$, where ϵ_u is modeled with a full covariance, W.

for the computation of LLR's. In particular,

$$\mathrm{LLR}(w_1, w_2) = \log \mathcal{N}\left(\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}; \begin{bmatrix} \mu \\ \mu \end{bmatrix}, \begin{bmatrix} \Sigma & B \\ B & \Sigma \end{bmatrix}\right) - \log \mathcal{N}\left(\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}; \begin{bmatrix} \mu \\ \mu \end{bmatrix}, \begin{bmatrix} \Sigma & 0 \\ 0 & \Sigma \end{bmatrix}\right), \tag{2.11}$$

where $B = VV^*$ as above and $\Sigma = B + W$ is the total data covariance. If we set the global offset $\mu = 0$, we can arrive at

$$\mathrm{LLR}(w_1, w_2) = w_1^* Q w_1 + w_2^* Q w_2 + 2 w_1^* P w_2 + Z, \tag{2.12}$$

where Z is some constant independent of w_1, w_2, and

$$Q = \Sigma^{-1} - \left(\Sigma - B\Sigma^{-1}B\right)^{-1}, \tag{2.13}$$

$$P = \Sigma^{-1} B \left(\Sigma - B\Sigma^{-1}B\right)^{-1}. \tag{2.14}$$

The computation is straightforward and can be implemented even more quickly with additional assumptions [18, 21].

For a speaker recognition evaluation, we compute the LLR for every necessary pair of i-vectors and use these scores to measure system performance, which amounts to two types of errors [2]:

- False acceptance / alarm (FA) – incorrectly believing that two utterances come from the same speaker;
- False rejection, or Miss – incorrectly believing that two utterances come from two separate speakers.

The frequency of these errors depend on the decision threshold. Higher thresholds will yield fewer positive detections overall, thus reducing false alarms but also increasing the number of misses. Conversely, lower thresholds will allow for more positive detections, thus decreasing the number of misses while increasing the number of false alarms.

When the National Institute of Standards and Technology (NIST) holds speaker recognition evaluations (SREs), they define a Detection Cost Function (DCF) that is a weighted sum of these FA and Miss rates. These weights correspond to costs, C_{FA} and C_{Miss}, respectively, along with an *a priori* probability of same speaker trials, $P_{=}$, and different speaker trials, $P_{\neq}$, all of which are specified by NIST depending on their desired application context [2]. This yields a DCF defined as follows:

$$\mathrm{DCF} = C_{\mathrm{FA}} P_{\neq} \cdot R_{\mathrm{FA}} + C_{\mathrm{Miss}} P_{=} \cdot R_{\mathrm{Miss}}, \tag{2.15}$$

where R_{FA} and R_{Miss} are rates of FA and Miss errors, respectively, on the test set. While the value of the DCF depends on the value of the threshold, we typically sweep across all possible threshold and report the MinDCF, which is the minimum value of the DCF obtained across all thresholds. This value is the principal metric used in all NIST SRE's [22, 23].

Another summary metric to compare performance of speaker recognition systems is the Equal Error Rate (EER), which represents the operating point at which the FA rate is equal to the Miss rate. The EER tends to be slightly easier to understand than the MinDCF; as such, our speaker recognition results in Chapters 3 and 4 will be reported in terms of the EER.

2.5.2 Language Identification

In language recognition, we have been able to achieve consistent and robust performance using the more traditional Linear Discriminant Analysis (LDA) – instead of its probabilistic counterpart, PLDA, described above – followed by cosine scoring [13] and an appropriate backend.

In order to better discriminate between classes, LDA looks to define a new orthogonal basis (rotation) within the feature space. In this case, different languages correspond to different classes, and a new basis is sought to simultaneously maximize the between-language variability while minimizing the within-language variability. In the analogous explanation of PLDA for speaker recognition above, the between-language covariance resembles B, while the within-language covariance resembles W. The LDA basis is then defined by a projection matrix, A, composed of the eigenvectors corresponding to the highest eigenvalues of the general equation

$$B\nu = \lambda W\nu, \tag{2.16}$$

where λ is the diagonal matrix of eigenvalues. We train A using the training data from all K target languages, which, in the case of the NIST LRE, typically ranges between 20 and 24 different languages [24, 25].

We apply the LDA projection matrix, A, to our i-vector w and normalize the result to unit length, yielding[4]

$$\tilde{w} = \frac{A^{*}w}{\|A^{*}w\|}. \tag{2.17}$$

We obtain a model i-vector, $\bar{w}_k$ for each language $k = 1, ..., K$ by assuming that

[4] In some systems, an additional Within-Class Covariance Normalization (WCCN) is also used [13, 26], but because its effectiveness does not seem to generalize to all evaluations, our experiments will not include WCCN.

languages can be modeled by a von Mises-Fisher distribution (vMF), which can be seen as the analog of a Gaussian distribution on the unit sphere [27]. Sparing the details of this distribution, it can be shown that the maximum likelihood estimate of $\bar{w}_k$ is

$$\bar{w}_k = \frac{\sum_{i=1}^{N_k} \tilde{w}_i^{(k)}}{\|\sum_{j=1}^{N_k} \tilde{w}_j^{(k)}\|}, \tag{2.18}$$

where $\tilde{w}_i^{(k)}$ is the i^{th} i-vector in the k^{th} language and N_k is the number of utterances in that language. And finally, scoring, according to the von Mises-Fisher distribution, merely reduces to a dot product between a test i-vector, $\tilde{w}_\tau$ and the model:

$$\text{score}(\tau, k) = \tilde{w}_\tau^* \bar{w}_k. \tag{2.19}$$

At this point, we effectively have a K-dimensional score vector, s_τ, for each test utterance, τ. One naive approach to language identification would be to treat each component of the score vector as though it were the likelihood of that particular language and simply select the index of the largest component of s_τ as our classification decision. However, this fails to take into account prior probabilities of each language, calibration issues, and other various relationships that may occur within the scores themselves. Instead, these score vectors need to be calibrated via the use of another multi-class probabilistic classifier on an additional, labeled, previously unused, development set that can yield calibrated class likelihoods [3].

Lengthy discussions of backend calibration for language recognition using both discriminative and generative techniques are given both in Chapter 5 of [3] and in Chapters 3 and 4 of [28]. For a Gaussian backend, it largely comes down to three main principles:

1. We assume that each score vector, s_τ, is drawn from a single multivariate Gaussian probability density corresponding to its target language.

2. During training, these Gaussian densities are estimated in maximum likelihood fashion subject to various constraints. For example, we might assume that all models in the score space share a common (possibly low-rank) within-class covariance [28]. Other approaches have found success training these Gaussian densities in discriminative fashion [13, 27].

3. During testing, the log-likelihood for each language is simply the log Gaussian probability density of s_τ, given the language. This yields a K-dimensional log-likelihood vector ℓ_τ for test utterance, τ.

As we will see in Chapter 5, language recognition systems achieve improved performance as a fused combination of multiple sub-systems. This fusion is typically done via multi-class logistic regression across all sub-systems [27]. Let there be G input recognizers such that the g^{th} recognizer outputs a log-likelihood vector $\ell_\tau^{(g)}$ for test utterance τ. Then the fused log-likelihood vector is

$$\ell'_\tau = \sum_{g=1}^{G} \alpha_g \ell_\tau^{(g)} + \beta, \tag{2.20}$$

where each α_g, $g = 1, ..., G$, is a scalar, and β is a K-dimensional bias vector [28]. These fusion coefficients $(\alpha_1, \alpha_2, ..., \alpha_G, \beta)$ can be estimated via the optimization of a regularized multi-class cross entropy loss function [3]; we refer the interested reader to [28] for a much more thorough exposition of this topic.

It is worth noting how far we have come since the i-vector representation of an utterance. From the acoustic frames of test utterance $\tau = \{y_1, y_2, ..., y_L\}$, we have obtained a raw i-vector w_τ, applied LDA and length normalization to obtain $\tilde{w}_\tau$, scored against all language model i-vectors to obtain a score vector s_τ, applied a Gaussian backend to obtain a log likelihood vector ℓ_τ, which is then potentially fused that with other sub-systems to obtain ℓ'_τ. And only now are we finally ready to evaluate the output of a language recognition system!

While it is most intuitive to refer to language recognition as a closed-set identification task, the evaluation metrics put forth by NIST treat the problem as a series of verification tasks, in which the fundamental question is, "does test utterance τ belong to target language k?" In this way, we use false alarm rate, R_{FA}, and miss rate, R_{Miss}, in a way similar to our evaluation of speaker recognition. The only difference is that we express false alarm rate in the form of a language pair, $R_{\text{FA}}(k_T, k_N)$, which is the rate at which some specific non-target language k_N is mistaken for the target language, k_T. Under this paradigm, we obtain an average cost, C_{avg}, as

$$C_{\text{avg}} = \frac{1}{K}\left(C_{\text{Miss}} P_{\text{target}} \cdot \sum_{k_T} R_{\text{Miss}}(k_T)\right. \tag{2.21}$$

$$\left. + \frac{1}{K-1}\left(C_{\text{FA}} P_{\text{non-target}} \cdot \sum_{k_T} \sum_{k_N \neq k_T} R_{\text{FA}}(k_T, k_N)\right)\right), \tag{2.22}$$

where K is the number of target languages, k_T and k_N denote target and non-target language, respectively. For the language recognition evaluations on which we report our results, NIST sets the application-dependent costs for miss and false alarm errors, respectively, to be $C_{\text{Miss}} = C_{\text{FA}} = 1$, and the probability of target and non-target trials, respectively, are set at $P_{\text{target}} = P_{\text{non-target}} = 0.5$ [24, 25]. Furthermore, we obtain our miss and false alarm rates for C_{avg} assuming a fixed detection threshold of 0 for all target languages; this is, in part, one reason why system calibration via an appropriate backend is necessary.

2.6 Summary

This chapter has provided an overview of i-vectors as they are used for speaker and language recognition. Broadly speaking, an utterance's i-vector is a low dimensional summary of that utterance's distribution of acoustic features with respect to some UBM. The estimation of hyper-parameters for i-vector extraction requires no speaker or language labels; instead, these labels are used only at the scoring stage of the process.

For speaker recognition, scoring is typically done using PLDA to compute a log likelihood ratio between (i) the hypothesis that our two test utterances originate from the same speaker and (ii) the hypothesis that our utterances come from different speakers. Evaluation of a speaker recognition system amounts to counting its respective rates of false alarm and missed detections, and our experiments in Chapters 3 and 4 will report the equal error rate (the operating point at which $R_{\text{FA}} = R_{\text{Miss}}$) as a summary metric.

In language recognition, our i-vectors undergo multiple transformations provided by LDA, length normalization, dot product scoring, a Gaussian backend, and, optionally, system fusion. We go to these lengths to ensure a properly calibrated multi-class classification system. Similar to speaker recognition, the evaluation of a language recognition system involves a weighted average of false alarm rate and missed detection rate for each target language. Our experiments in Chapter 5 will report this average cost, C_{avg}.

Chapter 3

Domain Adaptation for Speaker Recognition

In this chapter,[1] we consider the problem of domain adaptation for speaker recognition. While the speech signal in general conveys many levels of information – words (speech recognition), language (language identification), speaker identity (speaker recognition), and sentiment (emotion recognition), to name a few – the task of speaker recognition is to simply ascertain a speaker's identity independent of all that is being said, the language spoken, and the speaker's current emotional state [14]. Furthermore, because we operate under the assumption that every speaker has a unique vocal identity, this task is not limited to the discrete confines of phonetic classes, the set of the world's languages, or the human construct of various emotions. In this way, we reduce the problem of speaker modeling to that of modeling some continuous space of speaker identity amidst the noisy and nuisance-filled space of recorded speech.

Over the past five years, results from the Speaker Recognition Evaluations (SRE) put on by the National Institute of Standards and Technology (NIST) have progressed towards real-world robustness and applicability in impressive fashion. One of the keys to this success is the development of the i-vector approach discussed previously in Chapter 2, a low-dimensional, vector-based representation of audio that easily allows the use of large amounts of previously collected and labeled audio to characterize and exploit speaker and channel (i.e., all non-speaker) variabilities [12]. In the SRE scenario, data from thousands of speakers each making over 10 calls from at least 2 different handsets, collected in a consistent manner, has been readily available from previous years. However, it is unrealistic to expect access to such large sets of labeled data from matched conditions when deploying such a system to a new domain in the

[1] A portion of this work was previously published in [1]. Many thanks to my co-authors who made this all possible.

real world.

To this end, we are faced with the problem of adapting the system so that it can operate faithfully in this new domain. We further assume that some additional in-domain examples can be made available, but access to expert knowledge is limited. In this chapter of the thesis, we describe a challenge task using SRE data that demonstrates the effect of a subtle domain mismatch and design experiments that allow for an empirical exploration of unsupervised domain adaptation techniques on i-vector speaker recognition systems. The following section briefly reiterates the use of prior data in an i-vector system and, while Section 3.2 describes an experiment demonstrating the effect of a domain mismatch. We then outline a domain adaptation challenge task for exploring techniques to mitigate performance degradations due to such mismatch; this was one of the topics explored at the Johns Hopkins University (JHU), Center for Language and Speech Processing (CLSP) Summer Workshop 2013.[2] Section 3.3 presents some initial experiments that attempt to quantify, at least to some extent, the difference between the two domains in question and, to first order, shed some insight on how the domain adaptation problem can be approached. In Section 3.4, we propose our general experimental framework and describe how it fits into an i-vector speaker recognition system. Section 3.5 outlines the various clustering algorithms we explore, while in Section 3.6, we present our initial experiments, results, and analysis. Section 3.7 surveys the work that followed our investigations in [1], and finally, Section 3.8 summarizes this chapter and looks ahead to our subsequent exploration, which seeks to quantify exactly how many labels are necessary to achieve high performance in speaker recognition.

3.1 Preliminaries

While an overview of the i-vector system and theory was provided in Chapter 2, it is worth both reiterating which parts of the process require labeled and unlabeled data and re-establishing notation. Figure 3-1 shows, once again, a simplified block diagram of i-vector extraction and scoring. A speech utterance (e.g., one side of a telephone call in SREs) is first represented by how its acoustic features (MFCC+deltas) are distributed relative to a Universal Background Model (UBM), which is a Gaussian mixture model (GMM) characterizing speaker-independent speech feature distributions. This representation consists of the zeroth-order (counts) and first-order (means) sufficient statistics of the speech utterance. These sufficient statistics are then trans-

[2] In the literature, this task may also be referred to as the 2013 JHU CLSP Summer Workshop Challenge.

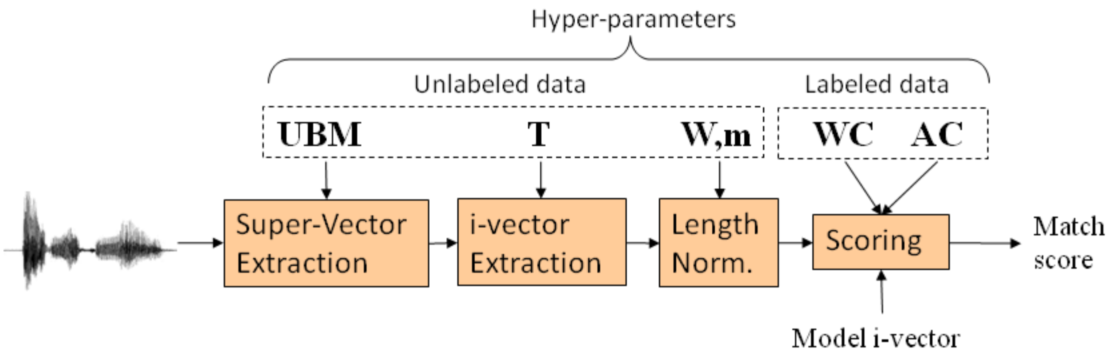


Figure 3-1: *High-level diagram of i-vector system showing all hyper-parameters and which, respectively, require labeled and unlabeled data to train. (Taken from [1].)*

formed into an i-vector, typically of 600 dimensions, using a total variability matrix T. The i-vector is then whitened by subtracting a global mean, m, and scaling by the inverse square root of a global covariance matrix, W, and then length-normalized to unit length [18]. Finally, a scoring function between a model and test i-vector is computed; this requires a within-class (WC) matrix, characterizing how i-vectors from a single speaker vary, and an across class (AC) matrix, characterizing how i-vectors between different speakers vary. The scoring function most often used is called Probablistic Linear Discriminant Analysis (PLDA) [18, 20], which we briefly covered in Chapter 2.

Collectively, the UBM, T, W, m, WC, and AC are known as the system's hyper-parameters and must be trained before a system can enroll and/or score any data. The UBM, T, W, and m represent general feature distributions and total variance of statistics and i-vectors, so they only require unlabeled data for training. The WC and AC matrices, however, require labeled data to learn within speaker (calls from the same speaker) and across speaker (calls from different speakers) variabilities. While all hyper-parameters are susceptible to mismatch, those requiring labeled data to train are more difficult to handle.

3.2 The Domain Adaptation Challenge Task

When porting a system to a new domain, we are faced with three options:

(1) Assume the new domain data is sufficiently close to the data used to train the hyper-parameters that the system will work well;

(2) Collect a large amount of unlabeled data from the new domain and adapt the hyper-parameters using unsupervised techniques;

(3) Collect and label sufficient amounts of new domain data to allow re-training or supervised adaptation of the hyper-parameters.

In this work, we will explore approaches to option (2). Another paper examines approaches to option (3) using limited labeled data in a similar context [29].

Using Linguistic Data Consortium (LDC) telephone corpora, we have designed an experiment that demonstrates the effect of data mismatch on hyper-parameters and defines the challenge task on which we are working. In this experiment, we use the telephone data from the 2010 NIST SRE (SRE10) as enroll and test sets. Specifically, we are using the one conversation (1c) telephone data enroll and test lists from condition 5 (normal vocal effort) [30, 22].

We designate two datasets to be used for hyper-parameter training:

- **SRE** – this consists of all telephone calls from all speakers taken from the SRE 04, 05, 06, and 08 collections. This will serve as the "matched" hyper-parameter training list or *in-domain* data.
- **SWB** – this consists of all telephone calls from all the speakers taken from switchboard-I and switchboard-II (all phases) corpora. This will serve as the "mismatched" hyper-parameter training list or *out-of-domain* data.

In accordance with the NIST SRE protocol, the speakers present in these prior data are explicitly disjoint from the speakers in SRE10 used to evaluate the system. Some key statistics of the two data sets are given in Table 3.1, while Figure 3-2 shows a histogram of the distribution of calls per speaker for both SWB and SRE.

Table 3.1: *Summary statistics for the SRE and SWB training lists.*

	SRE	**SWB**
# spkrs (m, f)	3790 (1115, 2675)	3114 (1461, 1653)
# calls	36470	33039
Avg. # calls/spkr	9.6	10.6
Avg. # phone-numbers/spkr	2.8	3.8

These two datasets appear very similar and the expectation is hyper-parameters trained from these should produce similar results. However, the resulting equal error rates (EER's) in Table 3.2 clearly show a gap in performance on the SRE10 enroll/test set when hyper-parameters are trained with the different sets.[3] Similar performance gaps were observed by other sites using independent i-vector implementations, indicating that the performance gap is not a function of particular implementation details (features, speech activity detection, hyper-parameter training algorithms, etc.).

[3] We provide our implementation details in Section 3.6.1.

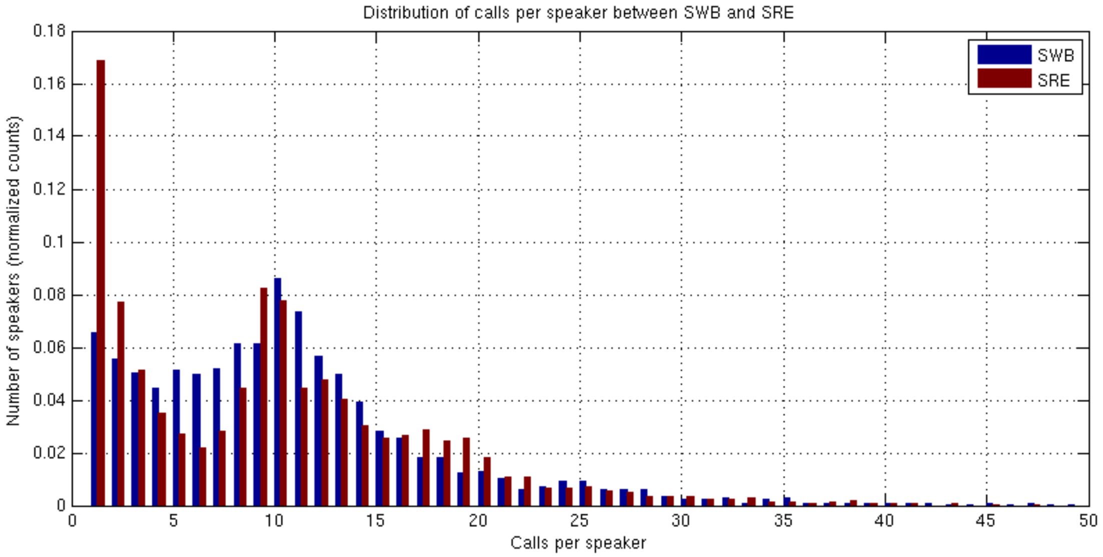


Figure 3-2: *Histogram comparing the respective distributions of calls per speaker in both the SWB and SRE corpora.*

Table 3.2: *EER's on SRE10 from hyper-parameters trained using the SWB or SRE datasets, as specified.*

#	UBM & T	W & m	WC & AC	1c EER (%)
1	SWB	SWB	SWB	6.92%
2	SWB	SRE	SWB	5.54%
3	SWB	SRE	SRE	**2.30**%
4	SRE	SRE	SRE	2.43%

In this work, we are primarily focused on how to effectively train or adapt the hyper-parameters that depend on labeled data (WC, AC) when only unlabeled data is available in the target domain. Of the hyper-parameters which do not depend on labeled data – UBM, T, W, and m – it was found on this challenge set that the difference in using SWB or SRE for UBM and T training was insignificant, but using SRE (in-domain) data for training the whitening, W and m, gave a significant improvement (compare rows 1 and 2 in Table 3.2) [29]. We will use the system specified in row 2 of Table 3.2 as our starting out-of-domain baseline and the result in row 3 as our desired in-domain benchmark. To avoid making this a data engineering exercise, we restrict our system to only use the labeled SWB data and unlabeled SRE data. The ultimate goal is to develop a recipe that can be applied in future situations where only unlabeled data from a new domain is available.

3.3 Exploring the Domain Mismatch

Before attempting to compensate for the mismatch in performance between the SWB and SRE corpora, we attempt to explain, anecdotally, some of the difference between the two datasets. An analysis of respective age distributions and languages spoken – SWB includes only English, while a small proportion of SRE also contains speech from over 20 different languages – yielded no fruitful insights. This came as a surprise; the work in [31] seemed to demonstrate that a discrepancy in languages spoken would introduce a dataset shift. However, we did notice that using different subsets of SWB produced different recognition results.

In particular, the entire SWB set can be broken down into six subsets that approximately correspond to their chronological release. Follow the labeling convention of [32], these subsets are: SWPH0 (1992), SWPH1 (1996), SWPH2 (1997), SWPH3 (1997), SWCELLP1 (1999), and SWCELLP2 (2000). We observed initially that, upon training a simple linear classifier to separate between the SRE and SWB i-vectors, the subsets of SWB that shared the most overlap with the SRE data were SWCELLP1 and SWCELLP2, while those that were easiest to separate were SWPH0, SWPH1, and SWPH2. Conversely, the 04, 05, 06, and 08 collections composing the SRE data seem to be more homogenous and do not exhibit similar trends. One possible explanation could be the difference between recorded cellular and landline telephone calls; perhaps the progression of technology caused a systematic shift in the data.

In light of this, we ran another set of baseline experiments using the various subsets of SWB in reverse chronological order. That is, we first use the labels from just SWCELLP2 and SWCELLP1, which are the two most recent subsets of SWB. Then we add in SWPH3, followed by SWPH2 and SWPH1, before finally including SWPH0. These results are shown in Table 3.3. For these experiments, we only varied the data used to train our WC & AC matrices; our UBM & T were always trained using the entirety of SWB (all subsets), while W & m were obtained using all of SRE. As such, row 4 of Table 3.3 displays exactly the same result as row 2 of Table 3.2, which was obtained using all of SWB.

From these results, it becomes clear that the SWB data is not homogenous and that there certainly exist subsets of our out-of-domain SWB data that are more suited to the in-domain SRE data. Similar findings were reported in [33], where the mismatch caused by different SWB subsets was compensated via a Nuisance Attribute Projection (NAP) before applying PLDA. These observations also seem to support the conjecture made above that the mismatch is, in part, driven by the progress in telephone technology moving from landline to cellular. A more detailed analysis of the meta-data, however, would be required before any more assertions can be made.

Table 3.3: *EER's on SRE10 using various subsets of SWB to train the WC & AC hyper-parameters. Each of rows 2-4 implies the use of the SWB subset specified as well as all the data in the rows above. Per Section 3.3, the UBM & T were trained using all of SWB, while W & m were obtained from all of SRE.*

#	WC & AC	1c EER (%)
1	SWCELLP1/2	4.67%
2	+ SWPH3	**3.51**%
3	+ SWPH1/2	4.85%
4	+ SWPH0	5.54%

We ran two additional experiments to test whether the domain mismatch can be overcome simply by selecting a subset of the out-of-domain SWB i-vectors for WC & AC training in some clever way. This sort of a strategy is known more formally in the literature as *covariate shift adaptation* and revolves around techniques such as *importance sampling* or *weighting* [34, 35]. Our initial experiments described below are not as sophisticated or well-developed, but we would like to point out that an initial attempt at the covariate shift problem in the context of closed-set speaker identification (i.e., as opposed to our problem of open-set speaker verification) was done in [35] and demonstrated some improvement using the techniques developed by [36, 37]. Some possible reasons why the work in [35] did not seem to achieve more significant improvements, despite demonstrating significant gains on the tasks evaluated in [36, 37], may have been due to their evaluating on a closed-set speaker identification test set consisting of only ten speakers as well as their choice to identify speakers using a mere 1.5 seconds of observed speech, which is orders of magnitude less data than the 150 seconds of speech that we typically use to build speaker models for our task. In light of this, an investigation of these methods under a more appropriate context may be a fruitful avenue for further analysis. For example, it might be interesting to model the progression of such a shift over time across the various SWB subsets and try to extrapolate this effect on the SRE data.

Figure 3-3 shows the result of our first approach in blue, where the resulting SRE10 EER is plotted as a function of the proportion, x, of SWB i-vectors that were the closest in likelihood to the marginal distribution of the in-domain SRE i-vectors. When $x = 1.0$, we are using all of SWB, so the result is, correspondingly, the same as both row 4 of Table 3.3 and row 2 of Table 3.2. Similarly, in green is the set of results obtained using the proportion, x, of SWB i-vectors that were closest to the SRE i-vectors in a linear discriminant sense. That is, we trained a simple linear classifier between the SWB and SRE corpora and used the subset of SWB i-vectors whose scores were closest to those of the SRE i-vectors.

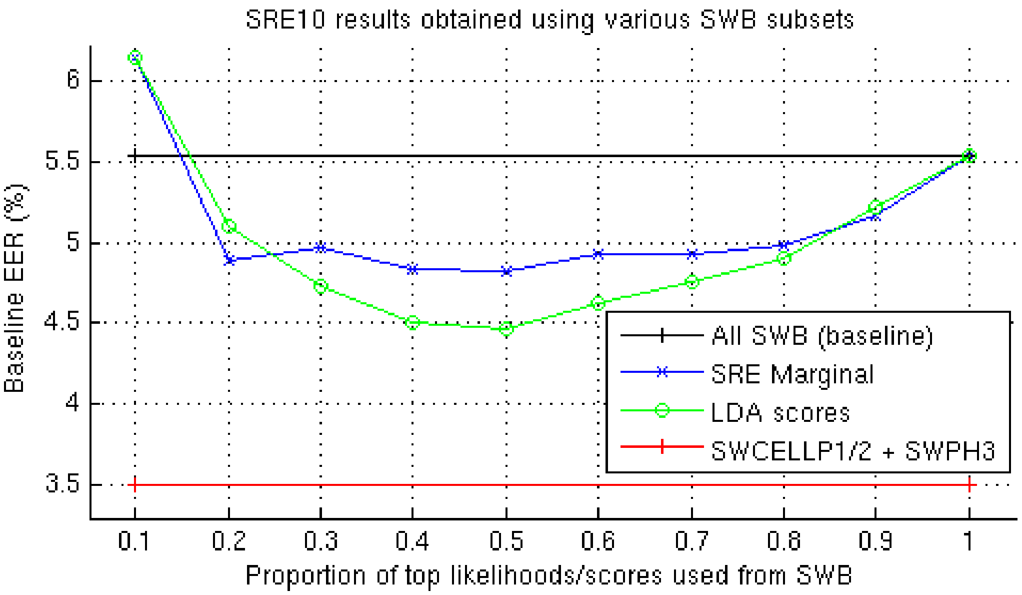


Figure 3-3: *SRE10 results obtained using various subsets of the SWB data for WC & AC.*

The results shown in Figure 3-3 suggest that there exist ways in which we can improve our baseline results by selecting, in unsupervised fashion, subsets of our out-of-domain data to match our in-domain data as closely as possible. However, our analysis is incomplete in two ways: (a) our methods for subset selection are still unable to attain performance comparable to the 3.51% EER obtained using SWCELLP1/2 + SWPH3 on row 2 of Table 3.3 and shown in red on Figure 3-3; and (b) we are still unable to explain why this aforementioned subset of SWB is able to obtain such an outstanding baseline result.

Upon implementing the domain adaptation framework outlined in Section 3.4, our experimental results still demonstrate that using *all* of the SWB data for WC & AC still provides the best speaker recognition performance. Indeed, despite the ability of subset selection to improve the initial baseline, there really is "nothing better than more data," at least in this context. Thus, the rest of our work in this chapter will not discern between different SWB subsets; we will use all of SWB as our labeled, out-of-domain data.

3.4 General Framework and Initial Setup

We begin our work assuming the existence of an initial set of hyper-parameters and a PLDA scoring function, as discussed in Chapter 2; exact implementational details are consistent with our parallel work in [38] and can be found in Section 3.6.1. For notational convenience, we will subsequently use Σ to refer to the WC matrix and Φ

to refer to the AC matrix. As such, our initial setup begins with Σ_{SWB} and Φ_{SWB}, which we train using the labeled SWB data that are provided.

We propose the following approach to the domain adaptation problem:

(a) Use Σ_{SWB} and Φ_{SWB} to compute a pairwise affinity matrix, A, on the unlabeled SRE data. Specifically, element A_{ij} is the log likelihood ratio between the hypothesis that i-vectors i and j are from the same speaker and the hypothesis that they come from different speakers.

(b) Use A to obtain a hypothesized speaker clustering of the SRE data. A discussion of different clustering algorithms will be covered in Section 3.5. These estimated speaker clusters can then be used to obtain estimates of Σ_{SRE} and Φ_{SRE}, which can be used in PLDA scoring for the final recognition task.

(c) However, instead of directly using Σ_{SRE} and Φ_{SRE} for recognition, we found success in linearly interpolating between the prior (SWB) and new (SRE) covariance matrices to obtain the final hyper-parameters, Σ_{F} and Φ_{F}, as follows:

$$\Sigma_{\text{F}} = \alpha_{\text{WC}} \cdot \Sigma_{\text{SRE}} + (1 - \alpha_{\text{WC}}) \cdot \Sigma_{\text{SWB}} \tag{3.1}$$

$$\Phi_{\text{F}} = \alpha_{\text{AC}} \cdot \Phi_{\text{SRE}} + (1 - \alpha_{\text{AC}}) \cdot \Phi_{\text{SWB}} \tag{3.2}$$

To simplify notation, we denote the set of parameters as $\alpha = \{\alpha_{\text{AC}}, \alpha_{\text{WC}}\}$. Note that setting $\alpha = \mathbf{1}$ corresponds to $\Sigma_{\text{F}} = \Sigma_{\text{SRE}}$ and $\Phi_{\text{F}} = \Phi_{\text{SRE}}$, or the hyper-parameters obtained using *only* the hypothesized speaker labels obtained from clustering the unlabeled SRE data. Conversely, setting $\alpha = \mathbf{0}$ is equivalent to not using any of the in-domain data; this yields the 5.54% EER shown on row 2 of Table 3.2.

Intuitively, we might expect that α_{AC} should reflect the relative proportion of the number of speakers between the SWB data and the subset of SRE data used, while α_{WC} should reflect the relative proportion of the number of utterances between the two datasets. However, we will see in both Sections 3.6.3 and 3.6.5 that an independent optimization of these parameters yield significantly better results (i.e., $\approx 10\%$ relative improvement) on the SRE10 evaluation. Section 3.6.3 will cover the results in more detail, while Section 3.6.5 will discuss potential ways to ascertain optimal values of α without the use of an oracle.

It should be noted that such an arbitrary weighting system may not be theoretically justified; the sum $\Sigma_{\text{F}} + \Phi_{\text{F}}$ should equal the total variability of our i-vectors, XX^T, where X is a $D \times N$ matrix consisting of all N of our D-dimensional i-vectors [39], but it does not. Nevertheless, in addition to observing improved performance, we can interpret this linear combination as a form of maximum a posteriori (MAP) adaptation with an associated set of relevance factors, $\alpha = \{\alpha_{\text{AC}}, \alpha_{\text{WC}}\}$ [4].

Lastly, another possibility is to iterate this procedure, where the Σ_{F} and Φ_{F} obtained in step (c) respectively replace the Σ_{SWB} and Φ_{SWB} of step (a) and the process is repeated until some form of convergence criterion is met, after which we proceed to the final recognition task. We observed from some pilot experiments that, assuming a reasonable choice of clustering algorithm in step (b), iterating can have a positive impact on both clustering and recognition performance, but its effect is quite small. Results tended to converge within just a few (≤ 5) iterations, so it is a relatively inexpensive modification that might be useful to explore in future work.

We should point out that this framework follows what [9] calls the "standard approach" to domain adaptation, where our out-of-domain data are treated as "prior knowledge" and maximum a posteriori values are estimated for the model parameters given the unlabeled in-domain data [40]. We initially coined this a "bootstrap" method based on the discussion in [41] on semi-supervised algorithms. While this approach has been successfully applied to language modeling, parsing and other problem domains in addition to ours [9],[4] we realize that there are other solutions to domain adaptation that can be applied in our scenario for potentially improved and more generalizable results [43, 44, 45]. For the purposes of this thesis and for the sake of presenting an effective initial solution to our version of the domain adaptation problem, however, we proceed with the approach just described.

3.5 Clustering Algorithms

Our experiments focus on the subset of algorithms that proved to be the most effective and practical from our previous work on large-scale speaker clustering [46]. Each of the algorithms explored is designed to work given only a (potentially sparse) pairwise affinity matrix, A.[5] That is, we need not go back to the raw data (i-vectors); simply knowing the pairwise relationships between them will suffice.

In our earlier work [46], we found success using agglomerative hierarchical clustering and various random walk-based clustering algorithms. Both *Infomap* and *Markov Clustering (MCL)* are explained in [47] and [48], respectively; we cover their essen-

[4] In fact, it was admitted in [42] that this "standard approach" tends to perform only slightly worse than the significantly more complex approach proposed in [9]. However, we also realize, from [41], that performance could potentially degrade even further if assumptions about the unlabeled data are not met.

[5] As described in step (a) of Section 3.4, A_{ij} would correspond to the log likelihood ratio between the hypothesis that i-vectors i and j are from the same speaker and the hypothesis that they come from different speakers.

tials in the following subsections. In the random walk setting, each i-vector can be thought of as a node on a large graph, and each edge is weighted according to the affinity between the two associated i-vectors.

3.5.1 Agglomerative Hierarchical Clustering

The well-known and widely-used *agglomerative hierarchical clustering (AHC)* is a simple and greedy algorithm that works in bottom-up fashion, initializing each i-vector as its own cluster and iteratively merging two clusters at a time via some cluster-similarity metric until some stopping criterion (e.g., BIC [49], maximum distance [50], number of clusters [46], etc.) is met [51]. In our implementation, the number of clusters is provided as an input; Section 3.6.4 discusses how that stopping criterion can be estimated automatically. In choosing a cluster-similarity metric, we follow our previous work in [46] and use the unweighted pair group method with arithmetic mean (UPGMA) [52] as a measure of similarity between two clusters. In particular, if we let $\mathcal{X}$ and $\mathcal{Y}$ be distinct clusters, let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, and let $d(x, y)$ represent some distance measure between two elements x and y (e.g., $-A_{xy}$ from our pairwise affinity matrix), then the distance between clusters $\mathcal{X}$ and $\mathcal{Y}$ would be

$$\mathcal{D}(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}| \cdot |\mathcal{Y}|} \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} d(x, y). \tag{3.3}$$

We also considered representing a newly merged cluster as the mean of all of the i-vectors in question,[6] but this would have been more expensive both in terms of computation and memory requirements. Re-computing the pairwise PLDA log likelihoods between the new cluster i-vector and all other clusters also requires that all other i-vectors be accessed repeatedly. Experiments run using this flavor of AHC are discussed more extensively in [38]; our experiments will demonstrate the suitability of UPGMA as a cluster-similarity metric.

3.5.2 Markov Clustering

As summarized in [53], Markov Clustering (MCL) converts a pairwise affinity matrix to a stochastic matrix by dividing the elements of each row by their sum and then iterates between the following two steps. During *expansion*, we compute an integer power of this matrix (usually a square), which yields the probability matrix of a random walk after that number of steps (e.g., 2). During *inflation*, each element

[6] One could even imagine extracting a new “cluster” i-vector altogether from the original utterances in the cluster.

of the matrix is raised to some power, α, artificially enhancing the probability of a random walker being trapped within a community. These steps are iterated until the stochastic matrix reaches a fixed point; the components of the resulting "forest" (i.e., disconnected clusters) are our discovered communities. By solely iterating on the stochastic matrix, this method satisfies the Markov property, and we obtain clusters of separated communities upon convergence. We run this algorithm according to the implementation provided by [48] using the default settings for the parameter $\alpha = 2$.

3.5.3 Infomap

The problem of finding the best cluster structure of a graph can be seen as the problem of optimally compressing its associated random walk sequence [47]. The goal of Infomap is to arrive at a two-level description that exploits both the network's structure and the fact that a random walker is statistically likely to spend long periods of time within certain clusters of nodes. More specifically, we look for a module partition $\mathbf{M}$ (i.e., set of cluster assignments) of N nodes into m clusters that minimizes the following expected description length of a single step in a random walk on the graph:

$$L(\mathbf{M}) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p^{i}_{\circlearrowright} H(\mathcal{P}^{i}). \tag{3.4}$$

This equation comprises two terms: first is the entropy of the movement between clusters, and second is the entropy of movements within clusters, both of which are weighted respectively by the frequency with which it occurs in the particular partitioning. Here, $q_{\curvearrowright}$ is the probability that the random walk switches clusters on any given step, and $H(\mathcal{Q})$ is the entropy of the top-level clusters. Similarly, $H(\mathcal{P}^i)$ is the entropy of the within-cluster movements and $p^{i}_{\circlearrowright}$ is the fraction of within-cluster movements that occur in cluster i.

Ultimately, Eqn. (3.4) serves as a criterion for a bottom-up agglomerative clustering search. The implementation provided by [47] uses Eqn. (3.4) to repeatedly merge the two clusters that give the largest decrease in description length until further merging gives an increase. Results are further refined using a simulated annealing approach, the specifics of which can be found in [47]. Our work in this paper, however, did not use this algorithm according to the exact implementation from [47]; rather, we used the modified version detailed in the following section.

Infomap-λ

Although the original formulation of Infomap in [47] involves no tuneable parameters except for the scale and sparsity of the affinity matrix provided as input, the minimization criterion presented in Eqn. (3.4) implicitly assigns equal weight to the between-cluster and within-cluster entropies. As such, we introduce a tuneable parameter, λ, into the equation as follows [46]:

$$L(\mathbf{M}) = q_{\curvearrowright} H(\mathcal{Q}) + \lambda \sum_{i=1}^{m} p_{\circlearrowright}^{i} H(\mathcal{P}^{i}). \tag{3.5}$$

The original Infomap corresponds to $\lambda = 1$. Letting $\lambda \to \infty$ increases our relative sensitivity to within-cluster entropy and yields more clusters that are smaller in size. Conversely, letting $\lambda \to 0$ favors larger and fewer clusters. In our previous work using the SRE data [46], we consistently obtained improved results using $\lambda > 1$ and achieved our best result using $\lambda = 1.5$. Even though this implies the use of prior knowledge, we run our experiments with $\lambda = 1.5$ to present an upper bound on our system performance.

3.5.4 Local Node Refinements

In agreement with our previous observations in [46], all of the methods just discussed worked well assuming some reasonable choice of a sparse graph (i.e., affinity matrix). As such, we implemented a local node pruning algorithm to automatically sparsify the affinity matrix [46].

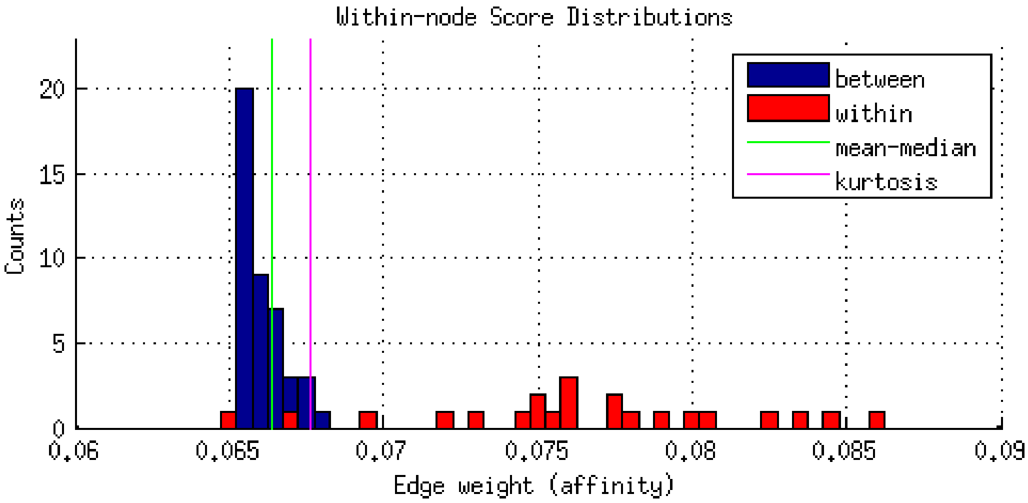


Figure 3-4: *Example histogram of within- and between-speaker score distributions for one particular node, as well as the cutoff thresholds discussed in Section 3.5.4.*

Figure 3-4 shows the distribution of the top 100 PLDA log likelihoods between an arbitrary utterance A produced by speaker s_A and the rest of the utterances in the data. These scores are separated into two different histograms: red "within-speaker" scores and blue "between-speaker" scores. The combined score distribution, including both within- and between-speaker scores, which is what the clustering algorithms see, has a right skew. Assuming, per the speaker recognition literature, that both within- and between-speaker scores can be modeled using respective Gaussian distributions [4], we can use simple measures of symmetry and kurtosis to arrive at the following heuristic to prune away between-speaker edges.

Let Z_A denote the combined distribution of scores for some node A. We keep the subset of scores Z_A^+, or edges, that are greater than some threshold θ_{mm} (i.e., $Z_A^+ = \{z \in Z_A | z > \theta_{\text{mm}}\}$), where θ_{mm} is the largest value such that for the subset of scores $Z_A^- = \{z \in Z_A | z \leq \theta_{\text{mm}}\}$, $\text{mean}(Z_A^-) \leq \text{median}(Z_A^-)$. This method assumes that the mean should be greater than the median in a combined score distribution with a right skew, but without the tail of within-speaker scores, the remaining between-speaker score distribution should be symmetric.

Taking the assumption of between-speaker score Gaussianity a step further, we introduce kurtosis into our local-node pruning. In this case, we choose θ_{kurt} to be the largest score value such that $\text{kurtosis}(Z_A^-) \leq 3$, where 3 is the kurtosis of a normal distribution. Figure 3-4 shows the cutoff found by kurtosis in magenta, as well as the cutoff, in green, found by the mean-median method above.

In our implementation of this heuristic, we take our full affinity matrix, consisting of all the pairwise PLDA log likelihoods between all of the i-vectors in the data, and sparsify it such that only the top 100 scores for each row (i.e., utterance) have non-zero entries, thus turning it into a 100-nearest neighbors graph. Then, for each row of the sparse matrix, we use the threshold $\tilde{\theta} = \max\{\theta_{\text{mm}}, \theta_{\text{kurt}}\}$. An edge was pruned away if either node in the edge-pair deemed the connection unnecessary.

3.6 Experiments and Results

3.6.1 i-vector System Implementation

In this section, we give an overview of the i-vector system implementation used in our experiments. Both the UBM, which is a Gaussian mixture model (GMM) characterizing speaker-independent speech feature distributions [4], and the total variability matrix, T [12], were trained using just SWB. Before obtaining any PLDA hyper-parameters for either SWB or SRE, we obtain a whitening transform (global mean subtraction and scaling by the inverse square root of the global covariance matrix, W) from just the unlabeled SRE data;[7] note that whitening is an unsupervised procedure and does not require any speaker labels [18]. This whitening transform is applied to both the i-vectors from the SWB and SRE, and finally, all the i-vectors are length-normalized to unit length. The initial PLDA hyperparameters, Φ_{SWB} and Σ_{SWB}, are then obtained using the speaker labels from SWB and their respective pre-processed i-vectors. From here, we can proceed to the approach outlined in Section 3.4.

3.6.2 Evaluating Cluster Error

In our initial experiments, we examine "speaker confusion error" as a metric to measure clustering performance in addition to the more standard measures of average cluster purity and average speaker fragmentation. We also show plots of the speaker confusion error in Figures 3-5 and 3-6. Admittedly, there exist a number of different metrics for evaluating cluster quality, including Precision and Recall, Normalized Mutual Information, F-score, B-cubed, et cetera [54]. We describe the one we chose below, which met our desire for a single number that summarizes the essential aspects of precision, recall, cluster confusion and purity and allows us to seamlessly compare performance across all algorithms and their parameters.

Let our r hypothesized clusters be indexed by i and our s true clusters be indexed by j. We evaluate our clustering output by considering all possible alignments that assign each hypothesized cluster i to exactly one true cluster j. Given such an alignment, say $i \leftrightarrow j$, we define cluster error as the number of elements in hypothesized cluster i whose true cluster is not actually j. Furthermore, any of the $|r - s|$ extra hypothesized or true clusters that did not receive an assignment are also counted as errors. Every possible alignment is considered, and the alignment that provides the smallest clustering error is used. In enforcing a one-to-one assignment of hypothesized-to-true clusters, we are able to summarize both the precision and recall of our clustering output. The procedure described above is equivalent to the evaluation procedure of "speaker confusion error" in the NIST Speaker Diarization task [55].

3.6.3 Initial Results and Observations

The results from our initial experiments are shown in rows 1-3 of Table 3.4. Instead of simply obtaining Σ_{SRE} and Φ_{SRE} by clustering just once on the entire SRE dataset, we strove to attain some form of statistical significance by randomly sampling just

[7] The only time we used SWB data for whitening was to obtain the results in row 1 of Table 3.2.

Table 3.4: *Results from initial experiments in domain adaptation. Clustering performance was evaluated using labels from the SRE data; recognition performance (EER's) is reported for the 1c task in SRE10. Section 3.6.3 explains rows 1-3; Section 3.6.4 discusses rows 5-6.*

		# Spkrs	# Clstrs	Clustering Performance		
#		K	$\hat{K}$	Confusion	Purity	Frag.
1	AHC	1000	1000*	7.4%	94.9%	1.20
2	Infomap-λ	—	918	18.2%	85.9%	1.44
3	MCL	—	997	15.1%	90.3%	1.45
4						
5	Infomap+AHC	1000	918	9.0%	92.6%	1.19
6	MCL+AHC	—	997	7.5%	94.9%	1.20

		α^* EER (%)			$\alpha = \mathbf{1}$ EER (%)		
#		Perfect	Hyp.	Gap	Perfect	Hyp.	Gap
1	AHC	2.37	**2.55**	**7.8**%	2.77	**3.16**	**14**%
2	Infomap-λ	—	2.71	14%	—	3.45	25%
3	MCL	—	2.68	13%	—	3.40	23%
4							
5	Infomap+AHC	2.37	2.56	8.2%	2.77	3.18	15%
6	MCL+AHC	—	**2.56**	**8.0**%	—	**3.16**	**14**%

a subset of the SRE data ten different times. For each sample, we randomly select $K = 1000$ speakers, cluster all their utterances using the various algorithms described previously in Section 3.5, and then obtain the corresponding hyper-parameters for the speaker recognition task. Rows 1-3 of Table 3.4 show our results averaged over ten such trials.

For each clustering algorithm described previously in Section 3.5, we report the following measures of clustering performance: number of speakers estimated ($\hat{K}$), cluster confusion error as detailed in Section and implemented in the evaluation of "speaker confusion error" in NIST Speaker Diarization scoring script [55], average cluster purity (ACP), and average speaker fragmentation (ASF). A given cluster's purity is the maximal proportion of that cluster that is represented by a single speaker; ACP is simply the mean purity computed over all clusters.[8] We define ASF as the average number of clusters used to represent all utterances of a single speaker.[9]

For recognition performance, we present the EER's obtained using the following configurations for α:

[8] Trivially, an ACP = 1 can be obtained when all clusters contain exactly one element.

[9] Conversely, an ASF = 1 can be achieved trivially if all elements are grouped into one single cluster.

$\alpha = \mathbf{0}$ **EER** – the SRE10 performance of the hyper-parameters obtained using *only* the labeled SWB data; i.e., $\Sigma_{\text{F}} = \Sigma_{\text{SWB}}$ and $\Phi_{\text{F}} = \Phi_{\text{SWB}}$. Independent of clustering, this result was reported on row 1 of Table 3.2: 5.54% EER.

$\alpha = \mathbf{1}$ **EER** – the SRE10 performance of hyper-parameters trained using *only* the hypothesized speaker labels obtained from clustering on the unlabeled SRE data; i.e., $\Sigma_{\text{F}} = \Sigma_{\text{SRE}}$ and $\Phi_{\text{F}} = \Phi_{\text{SRE}}$.

α^* **EER** – the best SRE10 performance obtained from hyper-parameters trained using both the hypothesized speaker labels obtained from clustering on SRE *and* combined (via some α) with the hyper-parameters from the labeled SWB data, as represented in Eqns. (3.1) and (3.2). In this scenario, we report the best result obtained for $\alpha \in [0, 1] \times [0, 1]$, where for simplicity we sample the space in intervals of 0.2.

In addition to reporting the results from our hypothesized (Hyp.) clusters, we also show the results of a perfect clustering, which is the SRE10 performance using hyper-parameters obtained from the use of exact speaker labels *and* the selection of the optimal values for α^*. Since we only use sampled subsets of the SRE data in this experiment, this result represents the best we can do. Admittedly, the "α^* EER" is an oracle-based experiment that assumes knowledge of some best-case scenario. We report our results this way so as to establish performance bounds in a controlled environment.

The results from our initial experiments are shown in rows 1-3 of Table 3.4. Instead of simply obtaining Σ_{SRE} and Φ_{SRE} by clustering just once on the entire SRE dataset, we strove to attain some form of statistical significance by sampling just a subset of the SRE data ten different times. For each sample, we randomly select all utterances from $K = 1000$ out of the original 3790 speakers, cluster all their utterances using the various algorithms described in Section 3.5, and then obtain the corresponding hyper-parameters for the speaker recognition task.

We can see that AHC provides the best clustering and recognition results by a significant margin. Yet, despite a rather wide range of clustering performances, the resulting range in speaker recognition performance is not nearly as dramatic. This could be specific to the SWB and SRE datasets; we know that the EER is upper-bounded at 5.5% using just the SWB hyper-parameters. When $\alpha = \mathbf{1}$, a better clustering algorithm yields better recognition results; however, the impact of clustering on recognition performance is attenuated by the presence of adaptation (i.e., when $\alpha \in (0, 1) \times (0, 1)$).

Figure 3-5 shows a summary of AHC clustering and subsequent recognition results as we vary the number of speakers sampled from the SRE data. The top plot shows, in red and green lines respectively, the α^* EER and $\alpha = \mathbf{1}$ EER, while the results obtained via hypothesized versus perfect clusters are denoted by dash-dotted and solid lines, respectively. In black is the $\alpha = \mathbf{0}$ EER line, while the blue line, whose corresponding y-axis is on the right side of the plot, denotes cluster confusion error. Despite cluster error increasing as we sample larger and larger speaker subsets, we can see that EER's on speaker recognition continue to decrease, although the rate of decline seems to slow after 2500 speakers.

The values of α^* given perfect clusters may be different from the values of α^* under the hypothesized clusters. In our experiments, we observed that α^*_{WC} for hypothesized clusters was consistently similar to α^*_{WC} for perfect clusters. However, the bottom plot of Figure 3-5 shows that the difference between α^*_{AC} for hypothesized (red) and perfect (blue) clusters increases with the number of speakers sampled. For reference, we also plot the value of $\hat{\alpha}_{\text{AC}}$ (black), which is the value that reflects the relative proportion of the number of speakers between the SRE data used and the SWB data. While values for $\alpha^*_{\text{WC}} \in [0.4, 0.8]$ for the most part, we see that $\alpha^*_{\text{AC}} \in [0.2, 0.6]$. This may suggest that adapting to the SRE WC matrix is of higher relative importance than adapting to the SRE AC matrix.

From rows 1-3 of Table 3.4, it is clear that AHC, when given the number of speakers, provides the best clustering and recognition results. Nevertheless, Infomap and MCL are able to do a reasonable job in detecting the number of speakers, which we did not explicitly explore with AHC. Instead, we could simply consider running MCL or Infomap to obtain an estimate of the number of speakers, $\hat{K}$, and use that estimate as an input to the AHC algorithm. This brings to bear the question of how robust AHC is to error in estimating the number of speakers. In particular, if either MCL or Infomap provides an imperfect estimate of the number of speakers as input to AHC, how much does that affect subsequent recognition results?

3.6.4 Effect of Cluster Number on Recognition Performance

Figure 3-6 shows the result of stopping AHC at varying numbers of clusters. These results are averaged over ten random draws of 1000 speakers, and α^* is optimized as previously discussed. The plot of cluster confusion error, in blue, is scaled according to the y-axis on the right and shows that clustering performance is best when AHC is provided a number of clusters equal to the number of speakers present. Yet, considering recognition performance alone, we can see that the resulting EER is relatively robust to stopping AHC at incorrect numbers of clusters. That is, we can actually

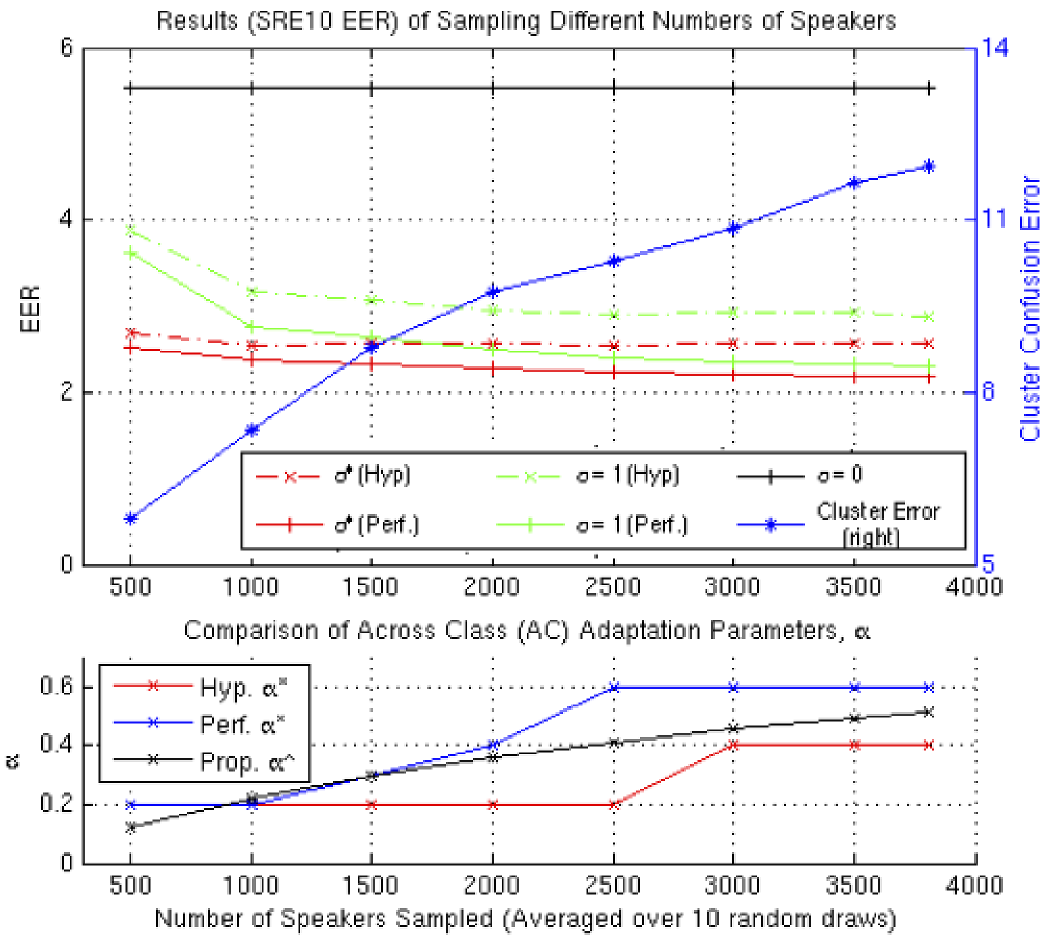


Figure 3-5: *Summary of clustering (AHC) and recognition (SRE10) results as a function of the number of speakers sampled from the SRE data. In the top plot, the blue line for Cluster Error is plotted according to the y-axis show on the right; all other lines are plotted according to the y-axis on the left.*

provide AHC with a significant underestimate of the number of speakers and still do fairly well on the SRE10. Additional experiments are necessary to better understand this phenomenon; in particular, an underestimate seems more forgiving than an overestimate, which implies the somewhat counterintuitive idea that modeling multiple speakers as one cluster is acceptable. One possible explanation is that an overestimate of the number of speakers will lead to an underestimate of the actual WC matrix, while conversely, an underestimate of the number of speakers allows the resulting WC matrix to model additional uncertainty that is somehow necessary and beneficial for speaker recognition. For now, we leave this as an open thread for further analysis in future work.

In rows 5-6 of Table 3.4, we show the results of using Infomap and MCL to

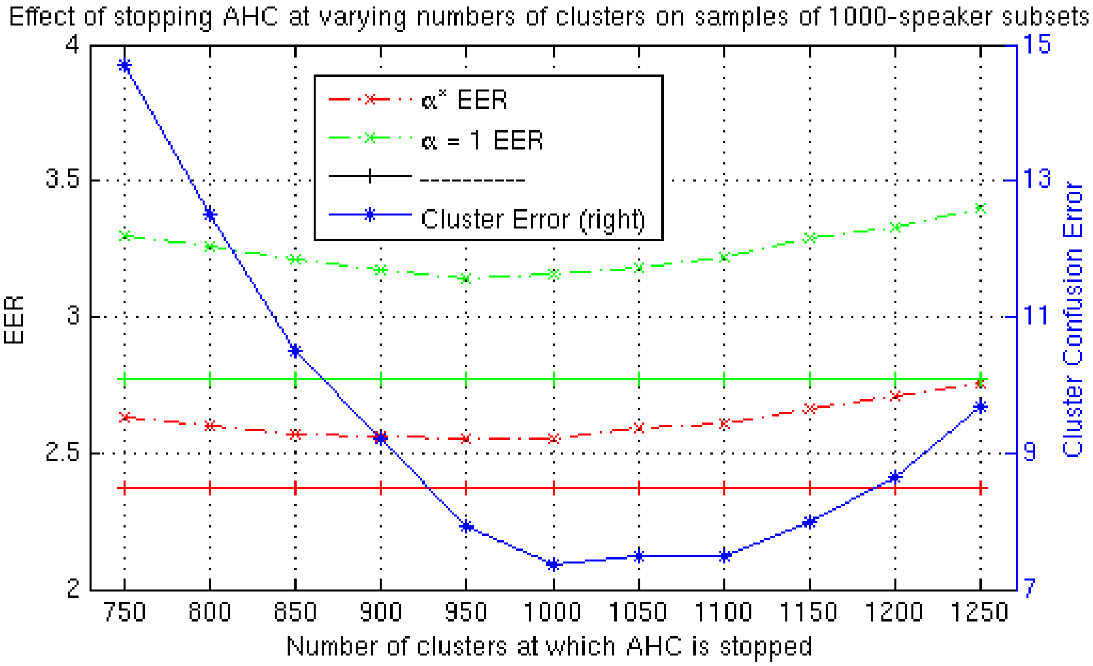


Figure 3-6: *Effect of stopping AHC at varying numbers of clusters, averaged over ten random draws of 1000 speakers each. Dash-dotted and solid lines correspond to results using hypothesized and perfect clusters, respectively. The blue line for Cluster Error is plotted according to the y-axis show on the right; all other lines are plotted according to the y-axis on the left. A more detailed explanation can be found in Section 3.6.4.*

estimate the number of speakers and taking that estimate as an input to AHC for clustering and recognition. We can see that both random walk algorithms are able to provide a reasonable estimate of the number of speakers, and the resulting recognition performance is just about as good as the case in which AHC is given the exact number of speakers (row 1). This is expected, as subsequent partitions produced at each step of AHC differ only by a single cluster merge and thus yield only small changes in cluster error. But more significantly, the gap in recognition performance between knowing and not knowing *a priori* the number of speakers in the unlabeled SRE is effectively nil. Indeed, when final recognition performance is the main priority, obtaining an exact estimate of the number of speakers may, in fact, be unnecessary.

As a final experiment, we run our proposed adaptation procedure on the full SRE data, using Infomap and MCL to estimate the number of speakers for input to AHC. Table 3.5 shows our final results, which were obtained with $\alpha^*_{\text{AC}} = 0.4$ and $\alpha^*_{\text{WC}} = 0.8$. Note how Infomap+AHC severely underestimates the number of speakers – thus obtaining the worst clustering performance of the three algorithms – but manages to attain recognition performance that is at least as good as when AHC is given the correct number of clusters. We hope to better understand this phenomenon in future work.

Table 3.5: *SRE10 results obtained using the entire unlabeled SRE dataset and optimal hyper-parameter adaptation, with $\alpha^*_{AC} = 0.4$ and $\alpha^*_{WC} = 0.8$. It should be noted that the 2.23% EER given a perfect clustering is different from the 1c EER of 2.30% shown in row 3 of Table 3.2 because of the adaptation with SWB hyper-parameters. The latter result is obtained with no adaptation, or $\alpha^*_{AC} = \alpha^*_{WC} = 1$.*

	$\hat{K}$	Perfect	Hypothesized	Gap (%)
AHC	3790*	2.23	2.58	16%
Infomap+AHC	3196	—	**2.53**	**13**%
MCL+AHC	3971	—	2.61	17%

3.6.5 Automatic Estimation of Adaptation Parameters

In this section, we examine the issue of automatically determining optimal values for $\alpha = \{\alpha_{\text{AC}}, \alpha_{\text{WC}}\}$. Figure 3-7 shows the result of independently optimizing both α_{AC} and α_{WC}, averaged over ten sampled subsets of 1000 speakers; the color scaling is shown to the right of each subplot, and blue indicates a relatively low EER, while red indicates a relatively high EER. The plot on the left suggests that there is a reasonably wide range of possible values for α that yield EER's less than 3%; this fact is consistent for sampled subsets that contain different numbers of speakers as well (e.g., 500, 1500, 2000, etc.). The heatmap on the right, in which the color scaling is limited to only the values that are within 10% of the optimal EER of 2.55% shown on the left, further confirms this notion. It seems as though we can obtain sufficiently good results simply by erring on the low side (i.e., $[0, 0.4]$) in our estimate of α_{AC} and using a moderate value of α_{WC} (i.e., $[0.4, 0.8]$). More experiments are needed to better understand this phenomenon and how it might generalize to other datasets, but this does seem to reflect our finding in previous sections that adapting to the SRE WC matrix is of higher relative importance than adapting to the SRE AC matrix. A parallel study on supervised domain adaptation reported a similar finding [29]. All of this suggests, perhaps, that the global space of speakers (modeled by the AC matrix) may be sufficiently well-represented in both the SRE and the SWB data; however, we need to find good ways to adapt our within-speaker modeling (WC matrix) to changes in the telephone recording channel over time.

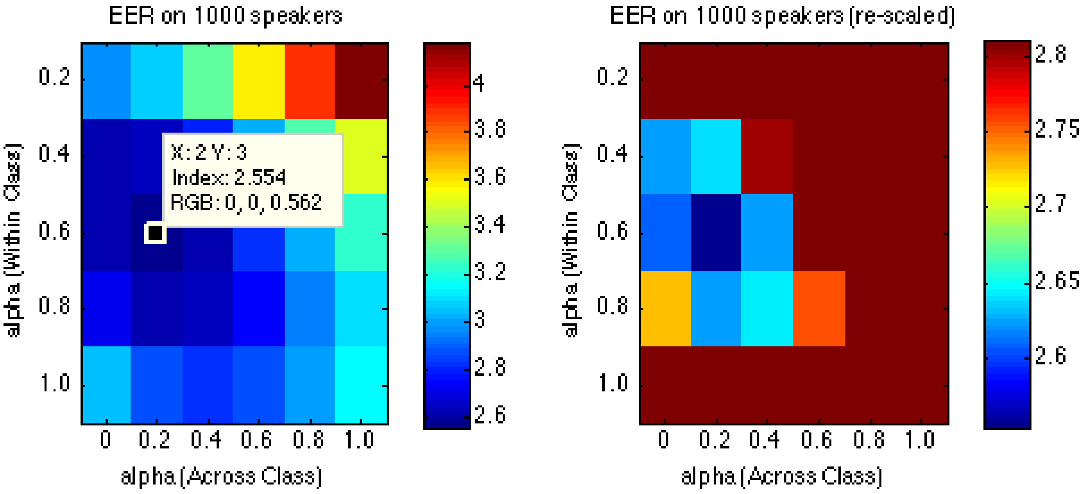


Figure 3-7: *Heatmaps showing the result of independently optimizing the adaptation parameters, α. Both plots involve the same raw data but different color scalings to illustrate the range of α that is appropriate for domain adaptation.*

3.7 Follow-up Work

Since our work on the task [1], a number of other approaches have achieved success on the domain adaptation task. The work in [56] extends our initial explorations in Section 3.3 and proposes a dynamic approach to data whitening that yields improvements without the need for any in-domain data, labeled or unlabeled, whatsoever. The work in [57] managed to obtain a more robust acoustic feature representation using a deep neural network trained from transcribed English. These features improved performance on all fronts: using hyperparameters trained on out-of-domain data, [57] managed to achieve speaker recognition performance similar to our proposed clustering methods without employing any explicit method for domain adaptation. Furthermore, using in-domain data to train their hyperparameters yielded a new state-of-the-art benchmark. We surmise that a combination of the clustering methods proposed in this chapter and the improved feature representation from [57] might yield even further performance gains. Our work in Chapter 5 proposes a method to extend the work of [57] and obtain a similarly effective feature representation without the need for any transcribed speech.

3.8 Summary

In this chapter, we motivate and define the domain adaptation challenge task for speaker recognition, in which both in-domain and out-of-domain data are available for system development, but labels are provided only for the mismatched, out-of-domain data. Our initial investigations into the cause of this mismatch seem to point towards the progression of telephone technology moving from landline to cellular. We propose an approach to overcome this mismatch that combines both agglomerative and graph clustering techniques to adapt our out-of-domain hyperparameters to fit the unlabeled, in-domain data. Results from our experiments suggest that both an imperfect clustering and an imprecise estimate of the number of speakers are forgivable in the presence of adaptation with out-of-domain hyperparameters. And although the optimal selection of their values remains an open question, we are able to obtain reasonable results from a wide range of adaptation parameter values. Ultimately, our final system obtains speaker recognition performance that is within 15% of a system that has access to all speaker labels.

In the following chapter, we consider a slightly different paradigm that no longer requires our system to passively contend with unlabeled, in-domain data in unsupervised fashion. Instead, we consider the scenario in which our system begins with no labeled data whatsoever, but has the ability to actively ask for information about particular examples. This moves us towards a semi-supervised and, more specifically, active learning approach to speaker recognition.

Chapter 4

Towards Active Learning for Speaker Recognition

The work we just discussed in Chapter 3 explored domain adaptation techniques that utilized, in an unsupervised manner, a set of matched, but unlabeled, data to supplement an existing system trained from labeled, but mismatched, data [1, 38]. Related work has also explored domain adaptation in the fully supervised sense [29]. While both scenarios are relevant, the actual deployment of a speaker recognition system into the real world is unlikely to warrant such extreme circumstances.

Obtaining a complete and exhaustive labeling of a set of N unlabeled utterances would require, in the worst case, $\frac{N \cdot (N-1)}{2}$ pairwise comparisons, but rather than deprive ourselves of any labels whatsoever as in the fully unsupervised case, perhaps we can obtain useful information from the expert labeling of some small fraction of these utterances. In this chapter,[1] we attempt to quantify how many labels are actually necessary to obtain state-of-the-art performance. We consider a scenario similar to the one considered in [1, 29, 38] in which we are provided with a vast quantity of matched, but completely unlabeled, data and are asked to build a speaker recognition system for subsequent audio.

To focus solely on the effect of limited labels, we remove the notion of mismatched domains – as in the domain adaptation problem previously explored – and assume that the unlabeled data at our disposal sufficiently matches the conditions in which we evaluate our speaker recognition system. Indeed, the existence of a previous system should only further reduce the number of expert labels required to obtain optimal performance, but we hope to keep things simple and not belabor our explorations with the implementation details of previous work on domain adaptation. Finally,

[1] A portion of this work was previously published in [58].

to simulate the presence of expert labeling, we query an oracle for the answers to pairwise comparisons, such as, “Do utterances A and B contain the same speaker?”

4.1 Related Work

The setup of this problem moves us into the realm of semi-supervised and, more specifically, active learning, in which the system we build is allowed to ask for input and supervision on a limited number of specific examples [11]. We design our oracle setup to operate like humans might; in particular, a human asked to separate N utterances into homogeneous speaker clusters would likely break the problem down into a set of pairwise comparisons. This setup also provides a framework for a potentially crowd-supervised [7, 59] speaker recognition system, which we plan to pursue as future work.

For now, we consider a set of N unlabeled utterances and allow our system to ask for additional information in the form of pairwise comparisons. These comparisons yield a set of pairwise constraints that can be used to help in our process of active, semi-supervised clustering, which was formally developed into a variant of the general K-means algorithm in [60]. Our work utilizes a much more primitive connected-components algorithm that does not require an estimate of the number of clusters, K.

The movement towards requiring less labeled data to build a speaker recognition system has been explored in the past. The saga of work in [61, 62] managed to obtain solid results without the use of any labeled data. Based on previous work in speaker diarization, they utilize an unsupervised clustering technique resembling a K-means algorithm that also estimates the number of clusters via a heuristic that re-assigns the elements of small clusters to larger ones. We adapted their methods to our setup (i.e., from supervectors to i-vectors) and were able to obtain clusters of comparable purity, but we were unable to produce an estimate of the number of speakers that was as accurate as reported in [61]. As we continue to explore their methods, we reserve for future work a more in-depth consideration of unsupervised methods for speaker recognition. But in accordance with the theme of this thesis, we continue to allow the use of labels, albeit as few as possible.

The rest of this chapter is organized as follows. We begin with an elementary theoretical overview in Section 4.2 that will set the expectations for our experimental results. Section 4.3 presents an overview of our system setup. Then Section 4.4 both outlines a naive initial algorithm that queries pairwise labels based on a nearest-neighbor approach and discusses its initial results. We propose techniques to further minimize the number of queries needed in Section 4.5, and finally, Sections 4.6 and 4.7 conclude with a discussion of potential avenues for future work.

4.2 Expectations on Sample Complexity

Consider a setting in which we are provided a set of N utterances for which speaker labels are unknown or unavailable. Asking humans to provide a complete and exhaustive labeling of this set of utterances would require $\mathcal{O}(N^2)$ pairwise comparisons, in which each comparison would involve two utterances, u_A, u_B, and the provided label $c(u_A, u_B) \in \{1, -1\}$, corresponding to "same" or "different" speakers, respectively. The setup of our problem can thus be re-framed as the problem of learning a binary classifier from a set of $\mathcal{O}(N^2)$ training examples, where each training example, x, is feature vector constructed by stacking a pair of i-vectors; i.e., $x = [w_A^* \; w_B^*]^*$. As it does not change the result of this analysis, we ignore for now the fact that the structure of this setup can be further exploited with the knowledge that the labeling of $c(x) = c(x')$, where $x' = [w_B^* \; w_A^*]^*$. Along similar lines, the work in [63] achieved competitive speaker recognition performance learning SVM classifiers from stacked i-vector pairs; for the sake of consistency within this thesis, we will continue scoring speaker similarity using PLDA.

In the passive learning setting under probably approximately correct (PAC) model assumptions [64], we know that the number of labeled examples required for a classifier to achieve at most ϵ error is, ignoring other factors, $\mathcal{O}(\frac{1}{\epsilon})$. If a system is given control over which training examples it would like to have labeled, however, the active learning literature [65, 66] has shown, under fairly reasonable assumptions, that the same performance can be achieved using just $\mathcal{O}(\log \frac{1}{\epsilon})$ labels.

In our evaluation framework, if we let the speaker recognition performance (in the EER sense) serve as a proxy for the ϵ error produced by our classifier, we should expect our active learning-based system to achieve performance (EER) that exhibits exponential decay as we increase the number of labeled examples, L, used; i.e., $\text{EER}(L) \propto e^{-\tau L}$, where $\tau > 0$. Our experiments will provide evidence to this claim, and our proposed methods will explore ways to increase the effective value of τ and quicken the exponential decay of our EER.

4.3 System Setup

We follow a similar setup to those presented in [1, 29, 38] and extract i-vectors [12] from a total variability matrix T of rank 600 and a gender-independent Universal Background Model containing 2048 Gaussian mixtures of acoustic features (MFCC+deltas). A more detailed background on how these hyper-parameters are obtained was covered in Chapter 2 and can also be found in [12, 67, 68].

Just like in our work on domain adaptation described in Chapter 3, we use the data from the NIST Speaker Recognition Evaluations (SRE) of previous years (2004-2008) to train these hyper-parameters. These telephone calls contain roughly 3800 unique speakers (1100 male, 2700 female) and 33,000 phone calls. The average number of calls per speaker is roughly 8.7, and each speaker is represented by 2.8 different phone numbers. While our work in the previous chapter also used data from all phases of Switchboard to train the hyper-parameters [1], this work is focused more on the use of limited labels than the issue of mismatched domains, so we focus only on the SRE data. Our performance evaluation is conducted on the one conversation (1c) telephone data from condition 5 (normal vocal effort) of the SRE 2010 (SRE10) [30, 22].[2]

To reiterate from Chapter 2, the training of these hyper-parameters does not require any labels; these initial i-vectors contain both speaker and channel (i.e., nuisance) information. Labels are not made available *a priori*, but if they were provided – or estimated via some clustering method – then we could obtain a within-class (WC) matrix, characterizing how i-vectors from a single speaker vary, and an across-class (AC) matrix, characterizing how i-vectors between different speakers vary [1]. The scoring function that has obtained state-of-the-art results is Probabilistic Linear Discriminant Analysis (PLDA) and is described in [20].

Our setup begins with a set of N utterances from the SRE data – the SRE10 data is *not* included here – represented as N i-vectors. We are allowed to ask some noiseless oracle for input in the form of pairwise labels; that is, "Are i-vectors i and j from the same speaker or different speakers?" The next section discusses a simple algorithm that makes use of these oracle queries to obtain respective WC and AC matrices, from which we can derive an appropriate PLDA scoring function and assess speaker recognition performance on SRE10.

[2] We also applied the methods discussed in this paper to various microphone conditions (1,2 – int-mic vs. int-mic; 4 – int-mic vs. room-mic) of the SRE10 and obtained trends similar to those reported on condition 5 (tel vs. tel) in this paper. In order to stay consistent with our preceding chapter, as well as previous work [1, 29, 38], we will continue reporting our performance based on the 1c telephone results from SRE10.

4.4 Naively Labeling Nearest-Neighbor Pairs

In this section, we propose a naive algorithm based on querying nearest-neighbor pairs to a noiseless oracle to obtain speaker clusters from which we can obtain WC and AC matrices for PLDA training and subsequent evaluation. We should note that there are a large variety of selective sampling-based query strategies in active learning that have been studied extensively [11, 69]; our initial approach here can be seen as a naive version of “balancing exploration and exploitation” [70] that we will further refine in Section 4.5.

4.4.1 The Algorithm

(a) Obtain pairwise cosine similarities in the form of an affinity matrix, A, where each entry A_{ij} represents the cosine similarity between i-vectors i and j. The cosine similarity (i.e., a length-normalized dot product) has been shown to be both a reasonable and fast metric for comparisons between i-vectors [12, 71, 72].

(b) Sort each row of A in descending order to obtain an ordered list of each node’s nearest neighbors and their similarities. Specifically, row $\tilde{A}(i,:)$ is the sorted list of cosine similarity scores produced by i-vector i. Accompanying $\tilde{A}$ is a matrix $\tilde{I}$ such that $\tilde{I}(i,:)$ is the corresponding list of i-vector indices with whom i-vector i produced each of the scores in $\tilde{A}(i,:)$.

(c) The c^{th} column of $\tilde{I}$ and $\tilde{A}$ specifies the respective indices and scores for the c^{th} nearest neighbor of each of our N i-vectors. For $c = 1, 2, ...$, query the pair $\left(i, \tilde{I}(i,c)\right)$ for each $i = 1, ..., N$. The number of unique pairs that are actually queried for each column $Q_c \leq N$, as each i-vector’s ranking of its respective nearest neighbors will differ.[3] Operate on all i-vectors for a given column (i.e., c^{th} nearest neighbor) before moving on to the $(c+1)^{\text{th}}$ nearest neighbor.

(d) Let G be an N-by-N binary matrix such that $G_{ij} = 1$ if i-vectors i and j originate from the same speaker, and $G_{ij} = 0$ otherwise. Initialize G as a matrix of all zeros, implying a completely disconnected graph, and when given a query that returns a same-speaker result, update G and its affected cliques. That is, if i-vectors i and j are a same-speaker pair (i.e., $G_{ij} = 1$), and $i \in I = \{i_1, i_2, ...\}$ while $j \in J = \{j_1, j_2, ...\}$ for cliques I and J, then this `clique-update` step automatically connects every element in I with every element in J. Because

[3] That is, without loss of generality, i-vector j may be i-vector i’s c^{th} nearest neighbor, but i may be j’s $\hat{c}^{\text{th}}$ nearest neighbor, for some $\hat{c} \leq c$.

we assume that our oracle is noiseless, this is easy and saves us from making superfluous queries.

(e) As G becomes more and more connected as a result of querying the oracle, the size of the cliques in G will increase. Since these cliques correspond to perfectly pure speaker clusters, use the non-singletons to obtain WC and AC matrices for PLDA scoring. In the next section, we present our initial results as a function of the number of labels actively queried from the oracle.

In (c), the choice to operate on each column separately instead of simply querying the pairs corresponding to the highest global similarity scores, i.e., `global score sort`, is in an effort to maximize coverage of the i-vector space [73, 74]; i.e., "explore" [70]. In our unknown manifold, the highest similarity scores may simply correspond to parts of the manifold that are most densely packed. By requiring that each column be treated separately, we enforce a more `uniform coverage` of the i-vector space, thus ensuring that every node (i-vector) in our graph (dataset) is considered at least once every N queries. This argument is not unlike the difference between using K nearest neighbors (uniform coverage) versus an ϵ-ball to build a graph; previous work has shown that the former (i.e., K-NN) yields better results in speaker recognition [46, 73, 74]. In the next section, the results of our initial experiments justify this hypothesis.

Along the lines of (d), we can further reduce the number of unnecessary queries by making use of information about different pairs that were previously encountered. For example, suppose the oracle returns (i, j) as a different-speaker pair, where $i \in X_q$ and $j \in Y_q$ for separate cliques X_q and Y_q in our graph G_q, where the subscript q denotes the state of the graph after q queries. Then for every subsequent query $q' > q$, if $x \in X_{q'}$ and $y \in Y_{q'}$, we already know without querying the oracle that (x, y) must be a different-speaker pair.

Admittedly, there are a number of other ways in which this algorithm can be further optimized. At the moment, we adhere to a static representation of the data, but as information is obtained from our oracle, we would ideally be able to continuously update and improve our representation of the i-vectors in the form of A, $\tilde{A}$, and $\tilde{I}$. Furthermore, apart from avoiding a few unnecessary queries per the explanation above, the generative-modeling framework of our PLDA setup has no way of making better use of negatively labeled pairs, which is valuable discriminative information obtained from the oracle. We re-consider some of these ideas in Section 4.5.

4.4.2 Initial Results

We visualize the results of our naive algorithm as a function of different subsets of the SRE data. Figure 4-1 shows the histograms of two different distributions of utterances-per-speaker in our sampled subset of SRE i-vectors. The subset represented in the plot at the top, `vanilla`, consists of all the utterances from 1000 speakers chosen uniformly at random, while the distribution displayed in the lower plot allows no more than five utterances per speaker, `max-5`. As this is a more difficult subset to work with, we will see that the baseline result of using all speaker labels for the `max-5` distribution (Figure 4-3) is slightly worse than the result of using all labels for the `vanilla` distribution (Figure 4-2).

While it serves no purpose from a theoretical point of view, it may be helpful to consider how many queries are needed to verify a perfect cluster-labeling of all of the utterances in each of our SRE subsets considered. Note that this is neither clustering from scratch nor any measure about sample complexity as discussed in Section 4.2; we are simply verifying the validity of some hypothesized partition, $\mathcal{H}$, of N i-vectors from M speakers. This can be done using $(N-M)+\frac{M\cdot(M-1)}{2}$ queries. The first term, $(N-M)$, comes from verifying the purity of each cluster – i.e., a speaker cluster of size $|C|$ would require $(|C|-1)$ queries to verify its purity – while the second term comes from verifying that each of the M clusters is indeed different from the rest. Verifying a clustering of the `vanilla` distribution would require 500,000 queries, while exhaustively checking a partition of the `max-5` distribution would require 7.2 million queries. We also show this information as part of the x-axis label at the bottom of Figures 4-2 and 4-3.

We plot our results as a function of the number of labels queried from the oracle (x-axis). While this includes queries of different-speaker pairs, this does not include the edges that are automatically created as a result of the `clique-update` step, only those in which the oracle is actively accessed. After every 1000 queries, we use the resulting cliques to define the speaker clusters, which are then applied to train a PLDA scoring function, and then run speaker recognition on the SRE10.

In Figure 4-2, we consider a number of different metrics as a function of the number of pairs queried (x-axis). On the top plot, we show the number of different-speaker pairs encountered by the oracle (blue), the number of automatic connections made (green), and the resulting number of edges in our graph G (red). The middle plot compares the number of clusters found with the number of actual speakers, and the bottom plot shows both our subsequent results on SRE10 – in the form of an equal error rate (EER) – using the labels queried as well as the results using all possible labels. In these lower plots, we also justify our algorithmic choice in (c) of

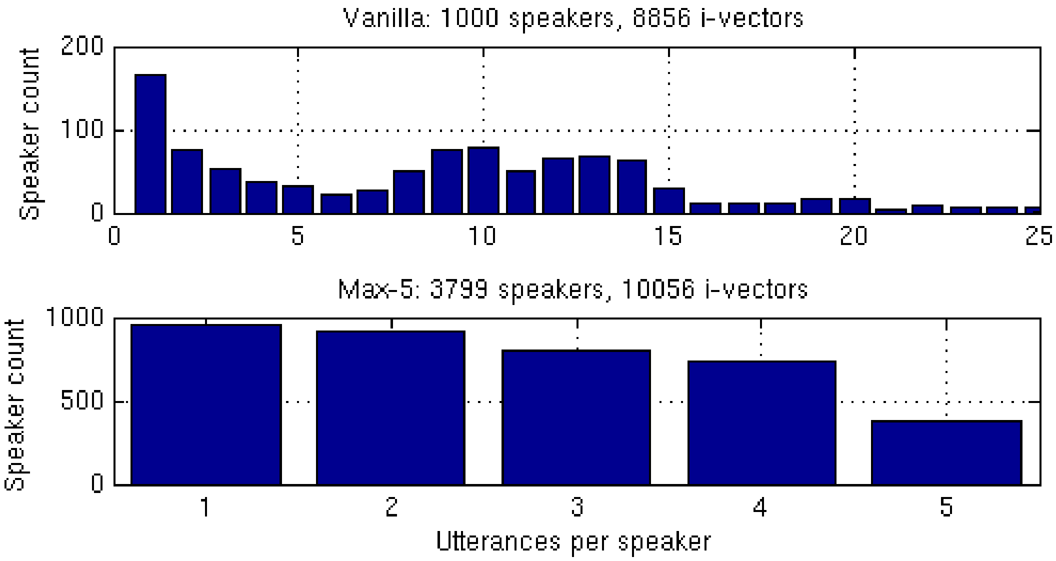


Figure 4-1: *Distributions of utterances per speaker in sampled subsets of SRE data: (top)* **`vanilla`** *– all utterances from 1000 randomly chosen speakers; (bottom)* **`max-5`** *– no more than five utterances from every speaker in the SRE data.*

Section 4.4.1 by showing the difference in results obtained using our chosen `uniform coverage` approach (blue), which queries all pairs corresponding to each i-vector's respective K^{th} nearest neighbor before moving to the $(K+1)^{\text{th}}$, as well as the `global score sort` approach (green), which sorts *all* possible pairs in order of decreasing similarity score.[4] Lastly, while both seem to exhibit the expected exponential decay in EER as a function of pairs queried – as discussed in Section 4.2 – the `uniform coverage` approach obtains a better EER with fewer pairs queried.

A relatively small number of queries already produces results comparable to those obtained using all corpus labels. In fact, the `vanilla` distribution yields the same EER using just about 9000 pairwise queries. On the other hand, using fewer than 5000 queries is worse than simply using the cosine similarity metric to evaluate on SRE10, which yields a 6.61% EER. This is because there are not enough edges on the graph to form reliable cliques that can faithfully model the WC and AC matrices for PLDA; in fact, with just 1000 queries using `global score sort`, these covariance matrices were not even of sufficient rank to complete PLDA training. As such, detecting when we have enough queries to sufficiently represent our speaker space would be an interesting direction for future work. Conversely, the rate of change in the number of cliques detected (middle plot) may be a reasonable indicator for when we have utilized enough queries from our data.

[4] Furthermore, when pairs were queried in a completely random order, obtaining results comparable to those shown required an unreasonably large number of oracle queries.

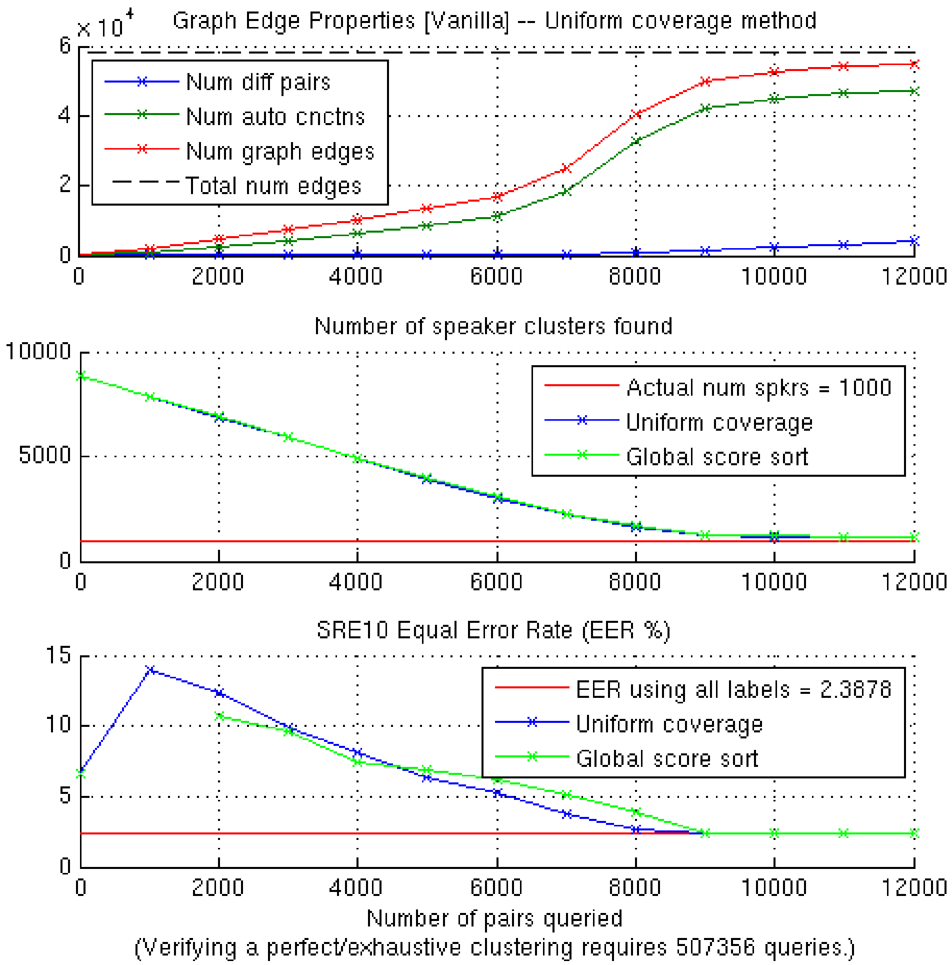


Figure 4-2: *Initial results obtained on the* `vanilla` *distribution of utterances per speaker: (top) graph edge properties as a function of pairs queried; (middle & bottom) estimated number of speakers and resulting SRE10 EER for the* `uniform coverage` *and* `global score sort` *approaches, respectively, as discussed in Section 4.4.1.*

The plot at the top of Figure 4-2 shows a rapid increase in the number of automatic connections made after 8000 queries. This is approximately where we begin querying pairs corresponding to second nearest neighbors. Once these connections are made, we start to see both the number of clusters found and the EER leveling off. The sudden spike in interconnected-ness makes sense given the `vanilla` distribution of utterances per speaker in our set of i-vectors; however, such a trend should not be expected for all such distributions.

For example, the top plot of Figure 4-3 shows no rapid increase in the number of automatic connections made for the `max-5` distribution and, relatively speaking, a much larger number of different-pairs encountered as a function of pairs queried.

The two lower plots in Figure 4-3 show the rest of the results obtained on the `max-5` distribution, namely the number of clusters found (middle) as well as the resulting SRE10 EER (bottom) as a function of the number of pairs queried (x-axis). For now, note that the `uniform coverage` approach (blue) exhibits similar trends on both the `vanilla` (Fig. 4-2) and `max-5` distributions, even though their respective baseline performances are different due to different speaker-utterance distributions. The other approaches will be discussed in the following section.

4.5 Refinements

So far, we have demonstrated the ability to obtain a good speaker recognition system using a relatively small number of pairwise queries. So far, these queries are chosen naively based on nearest neighbors according to a cosine distance metric. Having set an initial baseline, our interests turn to minimizing the number of active queries needed to obtain similar results.

4.5.1 Automatic, Noisy Labeling

One of the most straightforward ways to minimize the number of queries is to add edges automatically. Doing so would require making at least some assumptions about the data, such as the number of speakers present or a minimum cluster size. For example, it turns out that 95% of the first nearest-neighbor pairs correspond to the same speaker in the `vanilla` distribution.[5] Such an assumption would be dangerous to make for any general distribution of utterances per speaker, but for the `vanilla` scenario, one approach could be to simply connect all such first nearest-neighbor pairs and begin querying from the pairs corresponding to second nearest neighbors.

Unfortunately, upon initial experimentation, this turns out not to work so well. Allowing any impurities or noisy labels at such an early stage causes clustering errors to compound exponentially. One heuristic to avoid this might be to check the cluster purity on the largest clusters. Another remedy would be to use softer scores (i.e., weights $\in [0, 1]$) instead of adhering to $\{0, 1\}$ hard edge assignments. We note from earlier work [46], however, that the mere presence of an edge is more significant than the value of the weight itself. In light of this, we choose not to pursue this path of investigation in this work for now but note that there do exist principled ways to handle such noisy labels in clustering with partially labeled data that can be explored in future work [45].

[5] Unfortunately, the distribution of cosine similarity scores from the same- and different-speaker pairs show no discernible separation.

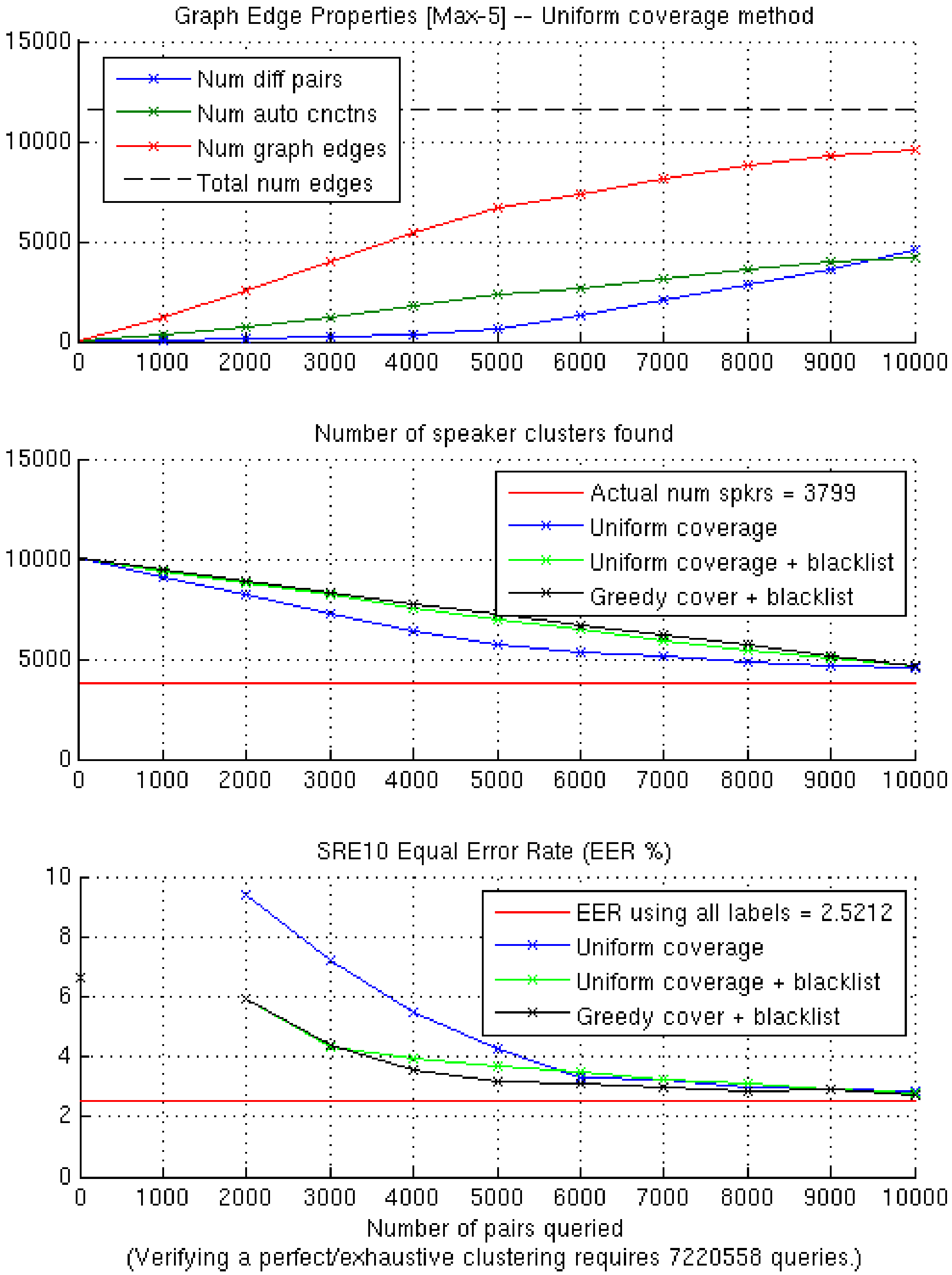


Figure 4-3: *Results obtained on the* ***max-5*** *distribution of utterances per speaker: (top) graph edge properties as a function of pairs queried; (middle & bottom) estimated number of speakers and resulting SRE10 EER for the* ***uniform coverage*** *approach as well as techniques discussed in Section 4.5.*

4.5.2 Data Re-representation

One way to make use of the additional information obtained via pairwise queries would be to re-represent the data using a new pairwise affinity matrix A'. Instead of the cosine similarity, we could let A'_{ij} be the log-likelihood ratio (LLR) that i-vectors i and j belong to the same speaker; this can be computed via PLDA, whose AC and WC matrices can be determined by the cliques in our graph G_q after q queries. The mechanics of this are straightforward; the more interesting problem is determining when such a re-representation is appropriate. That is, we would like to know when the scoring function we are currently using to compare i-vectors – whether it is a cosine similarity or a PLDA LLR – is no longer suitable.

We consider a "blacklisting" approach, where each i-vector i is queried against its successively more distant neighbors, $\{j_1, j_2, j_3, ...\}$, until the oracle returns a different-speaker pair (i.e., $(i, j_d) \in \mathcal{D}$ for some $d \geq 1$). Once that occurs, we know that either our scoring function is no longer reliable or, ideally, that all the utterances involving the speaker in i-vector i have been found. As such, we add i to the blacklist and ignore subsequent comparisons involving it. This method greedily finds, with respect to the scoring function at hand, all of the same-speaker neighbors in the local neighborhood of i, thus accelerating the growth of speaker cliques. Once all – or some predetermined percentage – of the nodes have been blacklisted, we know that either all the speaker clusters in our data have been found or a re-representation of the data is necessary.

4.5.3 Greedy Manifold Sampling and Clique-Growing

Another potential way to reduce the number of queries is to be more selective about the order in which we pose them. For a given set of c^{th} nearest-neighbor pairs, our initial algorithm saw no difference between asking in a random order or in order of decreasing similarity score, but perhaps we can do better by modifying the "blacklisting" approach to also sample the entire i-vector space as quickly and uniformly as possible.

Suppose we start at node i and query its nearest neighbors, $\{j_1, j_2, j_3, ...\}$, in order of decreasing similarity until the oracle returns a different-speaker pair for (i, j_d). Then we pick a node k that is as far away (i.e., dissimilar) from $\{i, j_1, ..., j_d\}$ as possible. This can easily be done by averaging the corresponding rows of A and picking the i-vector corresponding to the minimum average similarity. As before, we query the neighbors of k until a different-speaker pair is returned, and so on. Along the lines of the "exploration and exploitation" algorithm in [70], the hope is that this method will sample all corners of the manifold (exploration) in as few queries as possible and, at the same time, grow as large of speaker clusters as possible (exploitation) with every node visitation.

The result of these refinements is shown in Figure 4-3, which compares the `uniform coverage` algorithm (blue) from Section 4.4.1 to the methods described previously. Without even needing to re-represent the data, the "blacklisting" approach (green) immediately yields a significant improvement on the SRE10 EER using just 2000 pairwise queries. This shows the effectiveness of greedily growing speaker cliques. Finally, the impact of the `greedy cover` approach (black) can be seen starting at 4000 queries, thus demonstrating that maximal coverage of the i-vector manifold can indeed help maximize performance in speaker recognition with limited labeled data.

These ideas can be further refined along the lines of Maximal Marginal Relevance (MMR) introduced in [75]. In the parlance of information retrieval (IR), given some document query – not to be confused with our pairwise oracle queries – the idea behind MMR is to return an ordered list of documents, each of which is both as related as possible to the query itself (absolute relevance) and is as different as possible from the documents previously returned (relevant novelty) [75]. In our scenario, we can think of i as a document query and its nearest neighbors being the returned documents in order of decreasing absolute relevance. But instead of querying these nearest neighbors in order (i.e., $\{j_1, j_2, j_3, ...\}$), perhaps there is a way to incorporate the notion of relevant novelty that can further minimize the number of pairwise oracle queries needed per node visited.[6] We see this as a potential avenue for future work.

4.6 Future Work

We have left open a number of avenues for future work. As a way to minimize the number of queries needed, we have not yet considered the idea of adding edges automatically and how to handle such potentially noisy labels; this may warrant the use of soft graph edge weights (i.e., $[0, 1]$) instead of hard assignments ($\{0, 1\}$), as well as clustering techniques discussed in the previous chapter. To that end, we should apply the developments of our previous work on domain adaptation and verify that knowledge gained from previously labeled data, albeit from a mismatched domain, can improve our initial representation of the data in the form of a better pairwise affinity matrix, A. Lastly, our work so far has been based on the existence of a noiseless oracle, but previous work has shown that both naive and expert human listeners can be imperfect [59]. In future work, we plan to bridge the gap between

[6] That said, picking the next document query (node to visit), k, that is as unrelated as possible from documents (nodes) previously seen can simply be considered "absolute novelty!"

our noiseless oracle and a crowd-sourced system for speaker recognition.

4.7 Summary

In this chapter, we quantified the amount of labeled data needed to build a speaker recognition system. Beginning with unlabeled i-vectors and the cosine similarity metric, we query a noiseless oracle with nearest-neighbor pairs. Our initial results confirm the expectation that the sample complexity in the number of active labels needed to achieve state-of-the-art results is significantly lower than the number of labels needed in the passive learning case. We further refine our techniques in accordance with the "exploration and exploration" paradigm to maximize both cluster size and manifold coverage while minimizing both the number of queries needed and the resulting EER.

In both this and the previous chapter, we explored the problem of speaker recognition under various resource-constrained scenarios. In Chapter 3, we considered the problem of domain adaptation and discovered ways to utilize matched, but unlabeled data alongside a representation derived from labeled, but mismatched data. In this chapter, we entertained a scenario in which we were provided no labeled data to begin with, but our system was allowed to actively seek labels in the form of pairwise queries. In the following chapters, we examine analogous scenarios within the task of language identification and acoustic unit discovery. Along the lines of domain adaptation, in Chapter 5, we overcome the lack of transcribed data in certain target languages by discovering acoustic units in unsupervised fashion. Subsequently in Chapter 6, we look for ways to improve our discovery of acoustic units by incorporating weak supervision in the form of pairwise constraints.

Chapter 5

Acoustic Unit Discovery for Language Identification

5.1 Introduction

In this chapter,[1] we depart from our explorations of speaker recognition and consider the problem of language recognition. The effectiveness of deep neural networks (DNNs) for automatic speech recognition (ASR) [77] has led to their use in other speech-related classification tasks, including speaker and language recognition [78, 79, 80, 81, 57, 82, 83, 84]. The training of such DNNs, however, relies on the presence of pronunciation dictionaries and large amounts of transcribed speech, which may only be available for a small subset of the languages present in the evaluation task. For example, the work in [57, 82] used only transcribed English from the Switchboard I corpus [85] to build a system that could distinguish between 24 different languages, while the use of transcribed data from additional languages achieved even better results in [83]. On the other end of the spectrum, this brings to bear the question of how well we can do without any transcribed data. Following the premise of [86], we aim to exploit the existing sound pattern structure of speech without the need for transcription or a dictionary. More specifically, we investigate the effect of unsupervised acoustic unit discovery on language recognition.

To do so, we follow the framework proposed in [57, 82], where a DNN is trained from spectral input features and automatic speech recognition (ASR)-based output labels such that the activations at a so-called bottleneck (BN) layer provide frame-level features of manageable dimensionality. A BN feature of a given frame of audio can be seen as a compression of the information about both the frame's phonetic class

[1] A portion of this work has been accepted for publication in [76].

and context [81]. These features can then be treated as acoustic features of their own, from which an i-vector system can be built for language recognition [13]. Figure 5-1 presents an overview of this system.

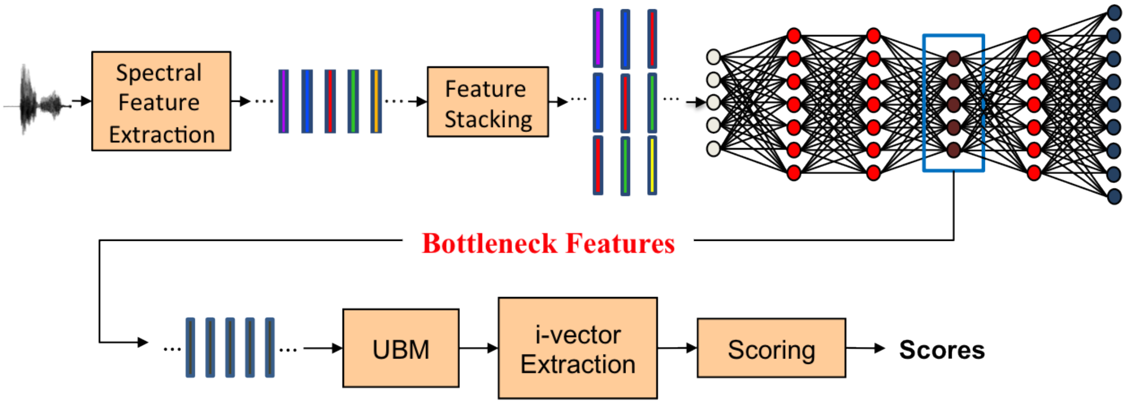


Figure 5-1: *An overview of a bottleneck i-vector system: stacked spectral features are passed as input to a neural network, whose activations at a bottleneck layer are used as features for an i-vector classification system. The resulting i-vectors are a low-dimensional summary of an utterance's distribution of bottleneck features.*

In our approach, unlike any recent work on language recognition to the best of our knowledge, we replace the ASR-based output labels from the original DNN-based setup with those learned from a Bayesian nonparametric model that learns an appropriate set of sub-word units automatically from speech data [5]. The development of this model was motivated by the desire for robust zero resource speech technologies that can operate without the expert-provided linguistic knowledge that standard ASR systems rely on [87]. Designed to uncover phone-like units from a given language, the resulting acoustic unit discovery system simultaneously segments the speech, discovers a proper set of sub-word units, and learns a Hidden Markov Model (HMM) for each [5]. Using neither transcribed data nor prior language-specific knowledge, this system obtained results on TIMIT that demonstrate the ability to discover sub-word units that are highly correlated with English phones, produced a better segmentation than the state-of-the-art unsupervised baseline, and performed well on a spoken term detection task [5].

Despite the promise of this model and that of similar systems [86, 88, 89], we are still unable to robustly and precisely uncover a particular language's phonetic inventory. As such, we instead broaden our consideration of unsupervised unit discovery from a monolingual setting to a multilingual one. Rather than focusing on any single language in particular, we aim to learn a set of acoustic units from many different languages at once. To paraphrase the analogy to human infants, who must specialize their speech perception and production systems to their native language (though perhaps with help from other sensory modalities) [87], we see our human infant as developing in a multilingual household. And more importantly, because we are not bound by any limited quantity of transcribed corpora, our models can instead be built on as much data as they can handle. Indeed, unsupervised methods give us the flexibility to work directly on data that matches the test domain, thus avoiding issues of language or channel mismatch.

In addition to the work reviewed in [87], the notion of transforming speech and audio data into a sequence of arbitrary symbols has been well-explored [90]. The work in [86] details the unsupervised training of an HMM-based self-organizing unit recognizer, while the work in [91] learns a set of "acoustic unit descriptors" to represent audio content for event classification and detection. The work in [92, 93, 94] proposes the Automatic Language Independent Speech Processing (ALISP) approach, which was initially developed for low bit-rate speech coding before evolving into a generic method for audio indexing, retrieval, and recognition, including initial attempts at speaker verification and forgery, as well as language identification [93].

Although more detailed explanations can be found throughout the rest of this chapter, let us first summarize the novel contributions and findings of our work below:

1. We show that a system built from learned acoustic units can be used effectively for language recognition. In particular, we find that a score-level fusion with a baseline system built from acoustic features yields substantial gains and significantly closes the gap between the acoustic feature baseline and a benchmark system built using transcribed English, suggesting that acoustic unit discovery provides complementary information to that of a bag-of-features baseline.

2. We find that using an improved representation of speech (i.e., supervised bottleneck features) as input to our acoustic unit discovery system can yield acoustic units that similarly improve performance on our language recognition task, and additional score-level fusion provides even further gains. This continues to motivate the need for a better understanding of the speech signal.

3. We demonstrate the ability to learn acoustic units in an unsupervised fashion on a dataset containing hundreds of hours of speech. This was achieved by modifying a Bayesian nonparametric model in a way that allows for effective parallelization. To the best of our knowledge, this is also the first Kaldi-based implementation of the acoustic unit discovery process [95].

4. We present our initial results on the Language Recognition Evaluation (LRE) from 2011 [24] presented by the National Institute of Standards and Technology

(NIST) and subsequently validate the generalizability of our proposed approach on the NIST 2015 LRE, which features a modified evaluation protocol involving specific language clusters.

5. We also run some initial experiments exploring a phonotactic approach to language recognition and use their results to motivate subsequent improvements to our proposed framework.

The rest of this chapter is organized as follows. Section 5.2 outlines our unit discovery process, and we reiterate the language recognition system from [57] that serves as our experimental framework in Section 5.3. Section 5.4 presents our initial results; Section 5.5 discusses some of the design choices that both worked and didn't work and validates our original results on another dataset. Finally, we take a look at phonotactic approaches to language recognition in Section 5.6 and then conclude in Sections 5.7 and 5.8 with a look ahead to future work.

5.2 Acoustic Unit Discovery

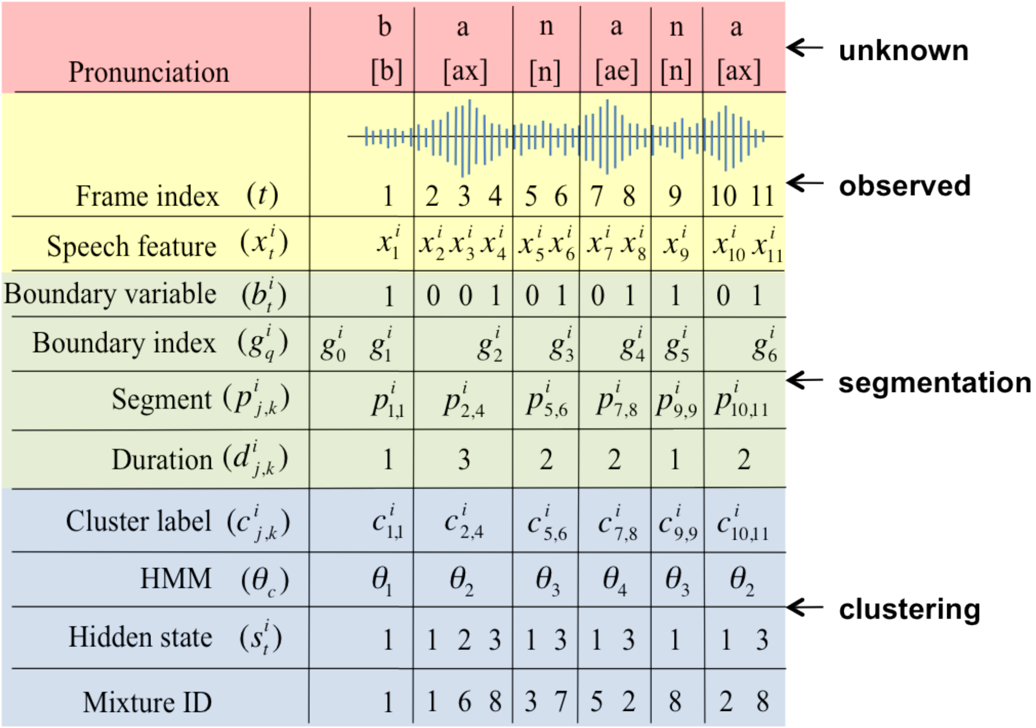
In this section, we outline the essentials of a previously-proposed acoustic unit discovery process [5] and highlight the modifications we make in its updated implementation to make inference more computationally feasible on data involving hundreds of hours of speech.

5.2.1 A Bayesian Nonparametric Model

Given a set of spoken utterances, the goal of acoustic unit discovery is to jointly learn the following [96]:

- segmentation – find the phonetic boundaries within each utterance;
- clustering – obtain an appropriate number of clusters within which acoustically similar segments can be grouped;
- modeling – learn a Hidden Markov Model (HMM) to model each sub-word acoustic unit.

In [5], all three sub-tasks were modeled using latent variables in a single Bayesian nonparametric model. More specifically, [5] formulates a Dirichlet process mixture model where each mixture is a HMM used to model a sub-word unit and generate observed segments of that unit. Via Gibbs sampling inference, the model seeks to obtain the set of sub-word units, segmentation, clustering, and HMMs that best represent the observed data.



Pronunciation	b [b]	a [ax]	n [n]	a [ae]	n [n]	a [ax]
Frame index (t)	1	2 3 4	5 6	7 8	9	10 11
Speech feature (x_t^i)	x_1^i	$x_2^i x_3^i x_4^i$	$x_5^i x_6^i$	$x_7^i x_8^i$	x_9^i	$x_{10}^i x_{11}^i$
Boundary variable (b_t^i)	1	0 0 1	0 1	0 1	1	0 1
Boundary index (g_q^i)	g_0^i g_1^i	g_2^i	g_3^i	g_4^i	g_5^i	g_6^i
Segment $(p_{j,k}^i)$	$p_{1,1}^i$	$p_{2,4}^i$	$p_{5,6}^i$	$p_{7,8}^i$	$p_{9,9}^i$	$p_{10,11}^i$
Duration $(d_{j,k}^i)$	1	3	2	2	1	2
Cluster label $(c_{j,k}^i)$	$c_{1,1}^i$	$c_{2,4}^i$	$c_{5,6}^i$	$c_{7,8}^i$	$c_{9,9}^i$	$c_{10,11}^i$
HMM (θ_c)	θ_1	θ_2	θ_3	θ_4	θ_3	θ_2
Hidden state (s_t^i)	1	1 2 3	1 3	1 3	1	1 3
Mixture ID	1	1 6 8	3 7	5 2	8	2 8

Figure 5-2: *An example of the observed data and hidden variables in the acoustic unit discovery model, modified directly from Figure 1 of [5].*

An explanation of the associated variables and the entire generative process, as well as a derivation of the conditional posterior distributions for each hidden variable in the model is provided in [5, 97]. Figure 5-2 directly replicates an example of the observed data and hidden variables of the setup [5]; we outline the essential ingredients below:

- speech feature (x_t) – 13-dimensional MFCCs and their first- and second-order derivatives extracted every 10 ms, resulting in a 39-dimensional observed feature vector;

- boundary (b_t) – a binary variable indicating whether a phone boundary exists $(b_t = 1)$ between x_t and x_{t+1} or not $(b_t = 0)$;

- HMM (Θ_c) – each HMM has three emission states, corresponding respectively to the beginning, middle, and end of each sub-word unit. A traversal of each HMM must start from the first (left-most) state, and transitions may only occur from left to right. While skipping of the middle and last states is allowed in

[5], our implementation requires that each segment be at least three frames in length. The emission probability of each state is modeled by a Gaussian Mixture Model (GMM).

- hidden state (s_t) – the hidden state index of the HMM associated with each feature vector, x_t.
- mixture ID (m_t) – the Gaussian mixture index associated with each feature vector, x_t.

If we assume, for the time being, that the values of the boundary variables, b_t, are given, then the generative process looks as follows:

1. Given a segment, $p = \{x_t | L < t \leq R\}$, as determined by two boundary variables ($b_L, b_R = 1$), choose a cluster label, $c \in \mathcal{C}$, which can either be an existing label or a new one. (The Dirichlet process allows for a potentially infinite number of clusters.) This cluster label will determine which HMM, Θ_c, is used to generate the segment.
2. Given the HMM corresponding to the cluster label, choose a hidden state, s_t, for each feature vector in the segment.
3. Given the hidden state of each feature vector, choose a mixture from the GMM of the chosen state, m_t.
4. Given the mixture ID, generate the observed feature vector, x_t.

A full derivation of conditional posterior distributions for each hidden variable in the model as needed by the Gibbs sampling procedure is beyond the scope of this thesis but is provided in [5, 97].

5.2.2 Boundary Variables and Landmark Detection

In practice, we reduce the inference load on the boundary variables, b_t, by exploiting acoustic cues in the feature space to eliminate the need for sampling on frames that are unlikely to be phonetic boundaries (i.e., $P(b_t = 0) = 1$). This is done by following the pre-segmentation method described in [98].

We should note that introducing boundary variables and allowing them to be sampled on or off during inference places this model in a unique space between more traditional HMM-based modeling and that of segment-based speech recognition [98]. While other methods use an initial segmentation to seed its HMM clusters, those segmentations tend to be fixed and subsequently discarded during later iterations of training in favor of the more traditional Viterbi decoding step [86, 94]. The model in [5] not only implements a form of duration modeling by forcing the 3-state HMM to represent the entire segment between two boundary variables ($b_l, b_r = 1$), it also allows for its boundary variables to be sampled on and off ($b_t \in \{0, 1\}$). This allows the model to continuously refine both its segmentation and clustering at a more localized level without having to rely on HMMs to model the duration of an acoustic unit.

5.2.3 Parallelization

While Gibbs sampling is theoretically guaranteed to converge to the true posterior distribution of the hidden variables in [5], the process can be quite slow – the sampling of each variable in turn requires updates to all its dependent variables at each frame of audio. In an effort to scale from processing the relatively small TIMIT corpus [99] containing less than ten hours of speech to a corpus containing a few hundred hours of audio, we focused our efforts on parallelizing the sampling algorithm. The drawback of such parallelization is that the resulting algorithm, while computationally scalable, will only *approximate* Gibbs sampling [100]. There have been attempts to better understand these effects at a more theoretical level, but these initial studies have been restricted to simpler models [101]; the impact of more complex approaches has largely been observed empirically.

A traditional, serial Gibbs sampler samples from one conditional posterior distribution at a time. Our implementation resembles that of a blocked Gibbs sampler and conditions on all of the HMM and GMM parameters to sample, in parallel, a new set of alignments (i.e., per-frame segmentation boundaries and cluster assignments) for all utterances. Assuming our data is split into some arbitrary number of partitions, P, this allows for parallelization that can effectively decrease the required computation time by a factor of $\frac{1}{P}$. Given an entirely new set of alignments, we then accumulate statistics to update, in batch mode, our Dirichlet process counts, HMM transition probabilities, and GMM emission probabilities accordingly. Distributing the sampling process across a number of parallel workers before accumulating statistics globally is a technique that has been explored as a parallelized version of Latent Dirichlet Allocation (LDA) known as Approximate Distributed LDA (AD-LDA) [100]. Despite losing the traditional Gibbs sampling guarantee of converging to the true posterior distribution of the hidden variables, our implementation achieves strong empirical performance in its ability to scale to large datasets.

5.2.4 Model Selection

Another difference between the model in [5] and our implementation is in the Dirichlet process (DP) mixture model. While the model allows for an infinite number of cluster labels in theory; practical implementations tend to over-initialize the number of possible mixtures and allow both the data and the DP concentration parameter, γ, to influence how many of those clusters actually retain probability mass. In our experiments, however, we found that our model would end up using all of the available mixtures, no matter how many we allowed for (up to 1000) or how small of a value we set for γ (as low as 0.001). This may have been a collective outcome of all our modifications; it may also indicate a lack of fit between our data and the model. Nevertheless, we found that the number of allowed clusters had a significant impact on both computational complexity and language recognition results; as such, we decided to fix the number of clusters at $|\mathcal{C}| = 100 << \infty$ and $\gamma = 1$, which achieves a balance between runtime and performance. While we realize this is no longer a principled implementation of a DP mixture model, it worked well in practice.

5.2.5 Other Modifications

We use the same pre-segmentation method used in [5] to obtain a set of candidate boundaries; this method essentially hypothesizes phonetic boundaries where the difference in spectral energy is large in magnitude. Originally built for a segment-based speech recognition system [98, 102], we further tuned this procedure to propose more candidate boundaries than usual, since the number of boundaries actually used (i.e., $b_t = 1$) will be a subset of those candidates. Lastly, while [5] imposed an equal prior probability on these candidate boundaries (i.e., $P(b_t = 1) = P(b_t = 0) = 0.5$), we found success in biasing the model towards keeping the boundary turned on with a prior of $P(b_t = 1) = 0.8$ and incorporating a localized post-processing step that merges consecutive segments if their respectively sampled cluster assignments are the same.

The model in [5] allows its HMMs to skip its middle and last states for segments shorter than three frames; our implementation does not allow for state-skipping and thus requires each acoustic unit to have a minimum duration of three frames. Our implementation also updates the GMM parameters in maximum likelihood fashion and increases the number of Gaussian mixtures it uses to model acoustic features at every pass through the data;[2] the formulation in [5] samples the emission probabilities of each HMM state using eight Gaussians with diagonal covariance matrices.

[2] This is done according to the method used by default in Kaldi [95].

5.2.6 Unit Recognizer Training

The unit discovery process essentially produces an acoustic model consisting of HMM parameters for each individual acoustic unit (i.e., mono-unit), as well as a set of per-frame alignments from the training data indicating the Gaussian mixture, the HMM state, and the HMM cluster label that are associated with each acoustic feature vector. If we collapse these per-frame alignments into unit sequences, they can be used as transcripts to train a "unit recognizer" in the traditional way. This releases the model from the segment-based rigidity of boundary variables and lets boundaries be determined automatically via a forced alignment of the data. Recognizer training also allows for context-dependent modeling (i.e., "tri-units") – something our unit discovery method cannot do – which can ultimately provide us with per-frame alignment sequences in the form of senones [103].

In addition to showing the results of using per-frame unit sequences and per-frame HMM state sequences from our context-independent acoustic unit discovery, we will also show the results obtained at the speaker-independent (SI) and speaker-dependent (SD) stages of context-dependent unit recognizer training. For the SI stage, we build a context tree containing roughly 2500 senones, and for the SD stage, we use roughly 4500 senones and include MLLT, fMLLR, and speaker adaptive training.[3] The exact number of senones obtained from the context tree is a function of the data and will differ between experiments; the resulting per-frame senone sequences are used as targets for training the DNN bottleneck features.

5.3 The Bottleneck i-vector System

In this section, we summarize the essential pieces of our bottleneck i-vector language recognition system. To the extent possible, we followed the setup of the state-of-the-art LID system presented in [57]. As discussed in Section 5.1, the overall process is summarized in Figure 5-1.

5.3.1 DNN Bottleneck Features

A DNN classifier is essentially a multi-layer perceptron with more than two hidden layers that typically uses random initialization and stochastic gradient descent to initialize and optimize its weights [77]. To provide temporal context, the input to the DNN is typically a stacked set of spectral features extracted from short (20 ms)

[3] These stages roughly follow the **tri2** and **tri4a** steps, respectively, in the `s5b` recipe of the Kaldi example for Switchboard I [95].

segments (frames) of speech. In our system, we computed 13 Gaussianized PLP coefficients as well as their first and second derivatives and then stacked $+/-10$ frames of context around the current input frame to obtain a $(13 * 3) * (10 + 10 + 1) = 819$ dimensional input feature vector to the DNN. The output of the DNN is a prediction of the posterior probability of the target classes for the current input frame; our experiments in Section 5.4 will explore the use of unit cluster labels, c_t, hidden states, s_t, and senones as target classes.

We use this DNN as a means of extracting features for use by a secondary classifier (i.e., an i-vector system). This is accomplished by using the activation of one of the DNN's hidden layers as a feature vector. In particular, we optimize a dimension-reducing linear transformation as part of the DNN training that results in a special "bottleneck" layer with fewer nodes and, thus, a manageable dimensionality. The bottleneck layer uses a linear activation and behaves very much like a LDA or PCA transformation on the activation of the previous layer [104, 105]. In addition to the previous work in [57], BN features also have been shown to work well for language recognition in [78, 81, 83, 84].

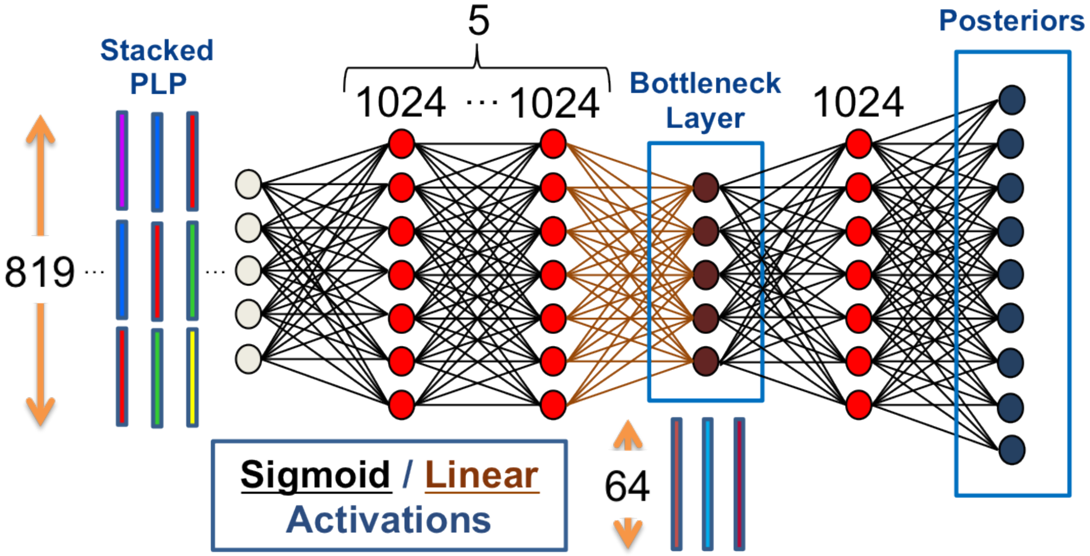


Figure 5-3: *The configuration of our proposed DNN. Its input is 819-dimensional vector of stacked PLP frames. The first five hidden layers contain 1024 nodes featuring sigmoid activations. This is followed by a 64-node bottleneck layer that uses linear activations (from which we draw our bottleneck features) and a final sigmoid layer with 1024 nodes. The number of output targets for which we obtain posteriors via a softmax depends on the result of the acoustic unit discovery step.*

As illustrated in Figure 5-3, all of our experiments utilize a common DNN structure containing seven hidden layers of 1024 nodes each with the exception of a bottleneck sixth layer, which has 64 nodes instead. All hidden layers use a sigmoid activation function except for the fifth layer, which is linear [105]. As mentioned above, the input layer contains 819 input features covering 21 frames of context. In this setup, the only difference between experiments is the number of target classes, which is determined by the acoustic unit discovery system described in Section 5.2.

5.3.2 i-vector System and Scoring

A more detailed description of the i-vector system and underlying theory was covered in Chapter 2 (and can be found in [12]); here, we provide a high-level overview of such a system built for language recognition (Figure 5-1) and note that our framework closely follows that of [57, 13, 27]. In our experiments, the only difference between the various systems will be in the original acoustic/bottleneck features used.

A test utterance whose language we hope to ascertain is first passed through a GMM-based speech activity detector, after which the detected speech is represented by a sequence of bottleneck feature vectors as obtained from a DNN classifier explained above. From these features, we obtain the zeroth-order (counts) and first-order (means) sufficient statistics of the utterance from a Universal Background Model (UBM), which is a 2048-mixture GMM characterizing a speaker- and language-independent feature distribution. These statistics are then transformed into a raw i-vector of 600 dimensions using a total variability matrix, T [12]. We transform this raw i-vector using linear discriminant analysis (LDA) and within-class covariance normalization (WCCN) [13, 26], both of which are estimated a priori from the training data and their language labels, and finally length-normalize the result to obtain a test i-vector. We use the dot product to compute the similarity score between the test i-vector and each language-representing model i-vector. These scores are calibrated using the discriminative Gaussian backend described in [27], which is trained using both scores and utterance durations.

5.4 Experiments

In this section, we first provide an overview of the data used in our experiments and present our initial results. Then we explore the use of score-level fusion and the incorporation of transcribed data on language recognition performance.

5.4.1 Corpora

Our experiments utilized three corpora in various ways. We evaluate all of our language recognition systems on the 2011 NIST Language Recognition Evaluation (**LRE11**), which covers 24 languages[4] coming from telephone and broadcast audio and has test durations of 3, 10, and 30 seconds [24]. The hyper-parameters for each of these systems – i.e., the UBM, the i-vector extractor, and the discriminative backend – are trained using the same training and development data from [27],[5] which we will refer to as LRE-train and LRE-dev, respectively. For our acoustic unit discovery on multilingual data, we used a subset of LRE-train consisting of 10 hours from each of the 24 evaluation languages, yielding a 240-hour **LRE-subset** dataset. For proper comparison with previous work in [57], we also use a 100-hour subset of Switchboard I [85] as defined by the example system distributed with Kaldi [95], which we will abbreviate as **SWB**. Finally, while all of the experiments in this section report results on LRE11, we demonstrate the generalizability of our methods in Section 5.5 by applying them to the 2015 NIST LRE (**LRE15**) [25, 106], which features a modified evaluation protocol involving explicit language clusters.

5.4.2 Spectral Feature Baseline

Following previous work, our baseline results come from an i-vector system built using spectral features. The baseline in [57], as well as in other work [13, 27, 107, 108], used Shifted Delta Cepstral (SDC) features in the conventional 7-1-3-7 scheme, which we described in Chapter 2. The seven static cepstra are appended to the 49 SDC features to produce a 56-dimensional acoustic feature vector. A more detailed explanation on how the SDC are obtained can be found in [107]. Whereas the work in [27] included vocal tract length normalization and feature-domain nuisance attribute projection, these techniques are neither used in our work nor that of [57].

5.4.3 Transcribed SWB Benchmark

We use the results obtained in [57] as our supervised benchmark system. This system trains a DNN from 4,199 senone target labels generated at the **tri4a** step from the

[4] The LRE11 languages include Arabic-Iraqi, Arabic-Levantine, Arabic-Maghrebi, Arabic-MSA, Bengali, Czech, Dari, English-American, English-Indian, Farsi, Hindi, Lao, Mandarin, Pashto, Polish, Punjabi, Russian, Slovak, Spanish, Tamil, Thai, Turkish, Ukrainian, and Urdu.

[5] Some of the corpora represented include CallFriend, CallHome, Mixer, OHSU, and OGI-22, VOA, Radio Free Asia/Europe, GALE broadcasts, and Arabic corporal from the LDC and Appen [27].

s5b recipe of the Kaldi example for Switchboard I [95], which we also adopted in Subsection 5.2.6.

5.4.4 Initial Results

In our initial experiment, we fix the number of acoustic units at 100 and run acoustic unit discovery on SWB and LRE-subset. This results in per-frame unit sequences for the 100 units and corresponding 300 states (for each 3-state HMM), both of which can be used as targets for DNN training. As described in Subsection 5.2.6, we also treat the resulting unit sequences as transcriptions and train an acoustic unit recognizer. We present our results obtained at two different stages of recognizer training: speaker-independent triphones (SI) and speaker-dependent (SD) modeling, which includes MLLT, fMLLR, and speaker adaptive training.

Table 5.1 presents our initial results, where for simplicity, we only show the average detection cost, C_{avg}, (as described in Chapter 2 and [24, 27]) on 30 second test segments of LRE11. In comparing between rows, we can see that running acoustic unit discovery on the multilingual LRE-subset is consistently better than running the unit discovery on the English-only SWB. This can be explained as either a result of domain adaptation to the multiple LRE11 languages or the effect of having 240 hours in the LRE-subset data versus 100 hours in SWB, or some combination of both. That said, the virtue of unsupervised methods is that they can be applied to the untranscribed multilingual data that matches the test domain; as such, subsequent results will be limited to units discovered on the LRE-subset.

Examining the columns from left to right, we can also see that each additional step of model refinement corresponds to additional improvements. Going from per-frame unit sequences (100) to context-independent HMM state sequences (300) yielded the most substantial gain; we note once again that both sets of sequences are solely the result of the acoustic unit discovery inference process (involving segment boundaries) and *not* a result of the unit recognizer training discussed in Subsection 5.2.6.[6] During unit recognizer training, the SI step utilizes context tree-based clustering that yields ∼2500 senones, while the SD step yields ∼4500 senones and incorporates speaker adaptive training [95]. As a result, we are able to obtain results comparable to the acoustic baseline using SDC features (Subsection 5.4.2). However, such performance is still significantly worse than that of the transcribed SWB benchmark (Subsection 5.4.3).

[6] Recall from Subsection 5.2.6 that the subsequent unit recognizer training ignores the original segment boundaries, b_t, from the acoustic unit discovery process and defines its own via the standard HMM training algorithms (forward-backward and Viterbi).

Table 5.1: *Initial language recognition results on 30 second test segments of LRE11; the numbers shown are the average detection costs $C_{avg} \times 100$. The* ***SWB*** *row shows the results of a system built from acoustic units discovered on a 100-hour subset of Switchboard I (English), while the* ***LRE-subset*** *row corresponds to that of a system build from units discovered on 240 hours of multilingual data. The various columns show the results at different stages of unit discovery (unit cluster labels versus HMM hidden state labels) and unit recognizer training (speaker-independent and speaker-dependent). The bottom two rows show our baseline and benchmark results, respectively.*

	100 units	300 states	SI	SD
SWB (100 hrs)	9.10	7.36	6.29	5.89
LRE-subset (240 hrs)	9.02	6.67	5.65	**5.24**
Spectral Feature Baseline (Subsection 5.4.2)				5.29
Transcribed SWB Benchmark (Subsection 5.4.3)				**2.60**

5.4.5 Incorporating Fusion

Given the similar performance of both the unit discovery-based system and the spectral feature baseline, we present the result of a score-level system fusion (via multi-class logistic regression [27]) between the baseline and the best-performing system from Table 5.1, which was built on the LRE-subset data using acoustic unit discovery (AUD) and speaker-dependent recognizer training. The results in Table 5.2 suggest that the unit discovery-based system captures language-related information complementary to that of the spectral feature baseline. Fusing these two systems together yields a 27%, 29%, and 14% relative gain on 30-, 10-, and 3-second test segments, respectively, and significantly reduces the gap between the baseline and the transcribed SWB benchmark without using any transcribed data.

Table 5.2: *Score-level fusion results on LRE11 for various test segment lengths (30, 10, and 3 seconds); the numbers shown are the average detection costs $C_{avg} \times 100$. We fuse the Spectral Feature Baseline (Subsection 5.4.2) with the best-performing system from Table 5.1, which was built on the LRE-subset data using acoustic unit discovery (AUD) and speaker-dependent unit recognizer training (Subsection 5.2.6). The Transcribed SWB Benchmark is discussed in Subsection 5.4.3.*

		30 sec	10 sec	3 sec
*	**Spectral Feature Baseline**	5.29	10.4	21.4
*	**AUD(LRE-subset), SD**	5.24	10.1	20.1
	Fusion of [*] above	**3.80**	**7.17**	**17.2**
	Transcribed SWB Benchmark	2.60	6.25	16.5

5.4.6 Incorporating Transcribed Data

The work presented thus far has focused on a fully unsupervised scenario that involves no transcribed data. Fusing an acoustic unit discovery-based system with a spectral feature baseline reduces the performance gap between the baseline and supervised system based on transcribed English. In practical scenarios, however, we will seldom be limited to situations in which we have absolutely no access to transcribed data or pronunciation dictionaries for any language. Instead, we are more likely to find ourselves in a situation where the linguistic knowledge we have at hand (e.g., American English) does not necessarily match the data we need to work with (e.g., the 23 other languages of LRE11). This situation was thoroughly explored in [57, 82] and is directly reflected in our transcribed SWB benchmark system (Subsection 5.4.3). In this section, we further investigate the effect of utilizing existing transcriptions, but strictly in the context of improving acoustic unit discovery for subsequent language recognition. We would like to see whether an improved representation of speech might result in the discovery of more salient acoustic units and thus improve language recognition performance.

To do so, we use the bottleneck features obtained from the transcribed SWB benchmark system described in Subsection 5.4.3 as the feature representation for acoustic unit discovery. That is, instead of using 39-dimensional MFCC features, we run the entire unit discovery and recognizer training process described in Section 5.2 using the 64-dimensional bottleneck features extracted from the transcribed SWB DNN (Subsection 5.4.3). The resulting per-frame senone labels are then used as output targets to train (from scratch) a brand new DNN whose input layer is, as before, the original 819-dimensional stacked PLP features. In this way, the transcribed SWB bottleneck features (SWB-BN) are used only as a pre-processing step for the unit discovery; thus, their impact manifests solely in the quality of the resulting per-frame senone labels.

Table 5.3 summarizes these results. The first row repeats the results from Tables 5.1 and 5.2 that ran acoustic unit discovery using MFCC features, while the second row displays the results of running acoustic unit discovery using SWB-BN features as described above. We can see the immediate impact of the improved feature representation and note that SD training did not provide any improvement over SI modeling – this makes sense, since the SWB-BN bottleneck features are trained using the speech of many speakers for the explicit purpose of discriminating between phonetic variability[7] – so we only show our SI results. But what is most important to realize is

[7] The work in [57] notes, however, that these same SWB-BN bottleneck features can be effective for speaker recognition when used in tandem with spectral features.

that these SWB-BN features are seen only by the unit discovery process to obtain the resulting per-frame senone sequences that we use as targets for subsequent DNN training. Everything else in our bottleneck i-vector system, from stacked PLP coefficients as DNN inputs to i-vector extraction, remains exactly as described in Section 5.3. Aside from the supervision involved in obtaining the SWB-BN features, the rest of the unit discovery system is still fully unsupervised. This clear difference in performance between the first and second rows of Table 5.3 demonstrates yet again the limitations of MFCC's as acoustic features.

Because we are now using transcribed SWB data to obtain our bottleneck features, it is only fair to compare our results against those of the transcribed SWB benchmark. While this benchmark is still the best individual system, its fusion with our (now semi-supervised) unit discovery-based system (using supervised SWB-BN features) yields relative gains of 19%, 16%, and 9% on 30-, 10-, and 3-second test segments, respectively. These results suggest that the information each of these two systems focuses on to make its classification decisions may be complementary.

Table 5.3: *Score-level fusion results on LRE11 for various test segment lengths (30, 10, and 3 seconds); the numbers shown are the average detection costs $C_{avg} \times 100$. The first row,* ***AUD(LRE-subset, MFCC), SD****, is the same result reported in Table 5.2. As discussed in Subsection 5.4.6, the results in the second row,* ***AUD(LRE-subset, SWB-BN), SI****, incorporate transcribed SWB data into the acoustic unit discovery process in the form of BN features. Our best results are obtained by fusing this semi-supervised system with the Transcribed SWB Benchmark (Subsection 5.4.3).*

		30 sec	**10 sec**	**3 sec**
	AUD(LRE-subset, MFCC), SD	5.24	10.1	20.1
**	**AUD(LRE-subset, SWB-BN), SI**	2.87	7.27	18.1
**	**Transcribed SWB Benchmark**	2.60	6.25	16.5
	Fusion of [**] above	**2.10**	**5.21**	**15.0**

We realize that there are a variety of other ways in which the transcribed SWB data can be utilized alongside the LRE-subset data in the form of unsupervised domain adaptation. For the sake of comparison, we explore a few of these methods but note that a thorough treatment of this problem is not the intended focus of this paper; our work simply aims to address the use of large-scale acoustic unit discovery as a tool to obtain features for language recognition. As a first experiment, we built and tuned an English recognizer from the transcribed SWB data (SWB-ASR), used the recognizer to decode the LRE-subset data, and built a new DNN from the corresponding per-frame senone sequences. For the next experiment, instead of a recognizer, we simply used the original SWB-BN DNN to classify each frame of the

LRE-subset data. Those classification results were then used as (potentially noisy) targets to train a brand new DNN from scratch.[8] In Table 5.4, we can see that both experiments yielded individual and score-level fusion results that were similar to those obtained via our acoustic unit discovery bottleneck i-vector system.

Table 5.4: *Individual and score-level fusion results on LRE11 for various test segment lengths (30, 10, and 3 seconds); the numbers shown are the average detection costs* $C_{avg} \times 100$. *All of the systems shown here use transcribed SWB in some way, and all but the benchmark system (Row 4) incorporate the use of LRE-subset data.* ***AUD*** *(Row 1) is the same result from Table 5.3 for a system that uses transcribed SWB data to obtain BN features for acoustic unit discovery on LRE-subset data.* ***SWB-ASR*** *(Row 2) uses a recognizer built from SWB to decode the LRE-subset data.* ***SWB-BN DNN*** *(Row 3) classifies each frame of the LRE-subset data using a DNN built from transcribed SWB data. Finally, our transcribed SWB benchmark result is shown again in Row 4.*

		30 sec	**10 sec**	**3 sec**
1	**AUD(LRE-subset, SWB-BN), SI**	2.87	7.27	18.1
2	**SWB-ASR, Decode LRE-subset**	2.96	7.25	17.2
3	**SWB-BN DNN, Classify LRE-subset**	2.69	6.72	16.8
4	**Transcribed SWB Benchmark**	2.60	6.25	16.5
	Fusion: 1 + 4	**2.10**	5.21	15.0
	Fusion: 2 + 4	2.12	**5.07**	**14.2**
	Fusion: 3 + 4	2.16	5.11	14.9

While the results shown in Table 5.3 demonstrate that even a little linguistic knowledge from just a single language (i.e., English) can have a huge impact in a multilingual setting (i.e., LRE11 performance), the results from Table 5.4 further suggest that such supervision can be utilized in a variety of ways and still yield good results. We can also see that the original benchmark system (Row 4 of Table 5.4), which simply uses the original SWB-BN features for language recognition, continues to be the single best-performing system. This may imply that bottleneck features are the most robust when trained using only labels obtained in a fully supervised fashion. We defer a more in-depth exploration of this phenomenon to future work.

5.5 Discussion

So far, we have seen that a large-scale acoustic unit discovery system can yield a segmentation and clustering of untranscribed data that is useful for language recog-

[8] Training a new DNN from scratch yielded better results than simply fine-tuning the original SWB-BN DNN.

nition. And while our focus continues to be on a fully unsupervised approach, we have also seen that even a little bit of supervision can go a long way to improve results. In this section, we discuss some of the other approaches we tried that did not work as well as planned and then demonstrate the generalizability of our approach to the previously unseen LRE15 data.

5.5.1 Negative Results

In the development of our proposed system, we explored the use of various levels of supervision in initializing the acoustic unit discovery process. In one experiment, we initialized our HMMs with a set of transcription-derived alignments (i.e., SWB). In another, we initialized our HMMs at random, but at every iteration, we updated our models using statistics accumulated from both the newly-sampled alignments (on LRE-subset) *and* the transcription-derived alignments (from SWB). We also tried scaling the statistics from these respective alignments in various ways to adjust the amount our updated model relied on each one. Our hypothesis was that maintaining some level of supervision might help anchor an otherwise unsupervised process; however, this set of experiments yielded results that were no different from simply initializing the HMMs at random and accumulating statistics from just the newly-sampled alignments.

We also considered different ways to obtain our set of potential phonetic boundaries (i.e., $\{b_t\}$ from Section 5.2). In addition to using the acoustic cues-based method from [98] with various parameter settings, we also experimented with the phone boundaries obtained from decoding the data using a speech recognizer trained on SWB. We found, however, that language recognition performance overall remained fairly stable across different segmentation methods, so long as the determined boundaries occurred at a reasonable frequency and, as discussed in Section 5.2.5, the prior on the boundary variables is biased towards being turned on (i.e., $P(b_t = 1) = 0.8$). Because the unit recognizer training step subsequently redefines these initial boundaries, the quality of the discovered units seems to be most dependent on how well we can cluster the data given the feature representation (i.e., MFCC or SWB-BN).

5.5.2 The 2015 NIST Language Recognition Evaluation

Despite the amount of system development involved in obtaining the results described in Section 5.4, we demonstrate here that our proposed methods can generalize from LRE11 to a previously-unseen language recognition evaluation. The 2015 NIST Language Recognition Evaluation encompasses 20 languages that can be grouped into six clusters[9] and, just like the LRE11, contains test durations of 3, 10, and 30 seconds [25]. Unlike previous evaluations, LRE15 focused on classifying target languages within the six language clusters. As such, we present our results on each language cluster, with the exception of French, as well as an average over all clusters. It was determined during the post-evaluation workshop that the French language cluster data featured a systematic channel mismatch between the train and test segments that led to near-random classification performance for most submitted systems. Furthermore, it was noted that Haitian Creole has a range of spoken forms, with the more formal variety being more French-like and the informal variety much less so [106]. Addressing this issue is beyond the scope of this investigation, so we omit these results; but for future work, it would be interesting to investigate more channel-robust (and speaker-independent) methods for unit discovery.

We used the development data provided by NIST: 141 hours of Arabic, 52 hours of Chinese, 65 hours of English, 4 hours of French, 23 hours of Iberian, and 30 hours of Slavic. A more complete breakdown of the amount of speech provided for the languages within each cluster can be found in [106]. Despite the uneven distribution of these data across the various language clusters, we decided against selecting a more balanced subset and ran our initial experiments using all of it.

The top three rows of Table 5.5 are analogous to the results shown in Table 5.2, which respectively include baseline results using just spectral features, results from units discovered on MFCC features, and results from a fusion of the two. The lower three rows present results analogous to those shown in Table 5.3, which include results from units discovered on SWB-BN features, benchmark results using just SWB-BN features, and results from a fusion of the two, respectively.

Table 5.5: *Results on LRE15 broken down by language cluster –* ***Ar****abic,* ***Ch****inese,* ***En****glish,* ***Ib****erian, and* ***Sl****avic – the numbers shown are the average detection costs* $C_{avg} \times 100$.

	Ar	**Ch**	**En**	**Ib**	**Sl**	**Avg**
*** Spectral Baseline**	26.6	23.4	16.9	23.4	11.4	20.3
*** AUD(MFCC), SD**	24.6	18.4	18.3	21.9	7.27	18.1
[*] above fused	**24.1**	**18.1**	**14.7**	**20.9**	**6.32**	**16.8**
**** AUD(SWB-BN), SI**	19.6	13.3	12.7	18.5	3.89	13.6
**** SWB-BN Benchmark**	19.6	13.1	11.2	18.4	3.27	13.1
[**] above fused	**18.6**	**11.8**	**10.3**	**17.1**	**2.89**	**12.1**

[9] Arabic – Egyptian, Iraqi, Levantine, Maghrebi, Modern Standard; Chinese – Cantonese, Mandarin, Min, Wu; English – British, General American, Indian; French – West African, Haitian Creole; Iberian – Caribbean Spanish, European Spanish, Latin American Spanish, Brazilian Portuguese; Slavic – Polish, Russian

We can see in Table 5.5 that the same trends from our LRE11 results persist in LRE15, thus demonstrating the applicability of our approach to different languages. In fact, acoustic unit discovery using MFCC features does substantially better than the spectral feature baseline on all language clusters except for English. Furthermore, the performance of our acoustic unit discovery-based system using SWB-BN features is just about the same as that of the SWB-BN Benchmark, again with the exception of English. Finally, we should note that in our actual submission to the LRE15, the two best performing individual systems were the SWB-BN Benchmark with a slightly different DNN configuration (i.e., 80 nodes at the bottleneck layer) and our acoustic unit discovery-based system as described here [106].

5.5.3 Exploring Language Specificity

Because the LRE15 explicitly focuses on distinct language clusters, we also explore the impact of language-specific perspectives in our unit discovery. Each row of Table 5.6 shows the result of building a language recognition system via unit discovery (on MFCC features) on just the specified language cluster. In each column, we highlight the best result obtained on the corresponding language cluster. This yields a fairly strong diagonal, where the only off-diagonal element is likely due to the lack of Iberian data relative to the amount of Arabic data. Otherwise, our results seem to confirm the notion that learning units on a particular family of related languages does indeed improve recognition performance for that specific language cluster, which may not be terribly profound but serves as further evidence that our acoustic unit discovery process captures language-specific information. Finally, fusing together the five language cluster-specific systems also yields a stronger result on each language cluster than simply discovering acoustic units from all languages pooled together.

A natural extension of these results would be a set of experiments that separate the impact of data amount from that of language specificity. In particular, Table 5.7 shows the results of running unit discovery on the same amount of data from each language cluster and building respective language recognition systems for each one. Iberian is our most data-limited language cluster, so we use all of its data and randomly select a 23-hour subset of data from every other language cluster for acoustic unit discovery. Upon highlighting the corresponding row that obtains the best result for each column, we see a diagonal that is not confounded by an imbalance of training data between language clusters. Compared to the results shown in Table 5.6, we can see that a decrease in the amount of data used for acoustic unit discovery on each language cluster seems to worsen language recognition performance on average, but not all language clusters are affected in the same way. For example, all of the results

Table 5.6: *Results on LRE15 broken down by language cluster –* ***Ar****abic,* ***Ch****inese,* ***En****glish,* ***Ib****erian, and* ***Sl****avic – the numbers shown are the average detection costs* $C_{avg} \times 100$*. For each of these systems, we run acoustic unit discovery using only the data from the language cluster specified and build a language recognition system to classify languages from all five language clusters.*

	# hrs	**Ar**	**Ch**	**En**	**Ib**	**Sl**	**Avg**
Ar	141	**25.1**	20.3	18.8	**22.0**	8.21	18.9
Ch	52	25.6	**19.8**	17.8	22.9	9.14	19.0
En	65	25.8	**19.8**	**15.9**	23.0	8.20	18.5
Ib	23	27.3	21.5	19.8	22.5	9.71	20.2
Sl	30	26.5	20.5	19.6	22.7	**7.63**	19.4
Fused	(315)	24.5	17.6	14.8	20.6	6.11	**16.7**
All	315	24.6	18.4	18.3	21.9	7.27	18.1

on the English language cluster actually improve with the decrease in the amount of provided data for acoustic unit discovery. Table 5.5 also reflects this anomaly: the spectral baseline significantly outperforms the unit discovery method (16.9 vs. 18.3) on the English language cluster. Our future investigations will explore the causes of such systematic discrepancies in our results on the English language cluster.

Table 5.7: *Results on LRE15 broken down by language cluster –* ***Ar****abic,* ***Ch****inese,* ***En****glish,* ***Ib****erian, and* ***Sl****avic – the numbers shown are the average detection costs* $C_{avg} \times 100$*. For each of these systems, we run acoustic unit discovery using only a 23 hour subset of the data from the language cluster specified and build a language recognition system to classify languages from all five language clusters.*

	# hrs	**Ar**	**Ch**	**En**	**Ib**	**Sl**	**Avg**
Ar	23	**25.8**	21.4	18.6	22.9	8.84	19.5
Ch	23	26.7	**21.0**	17.5	22.9	10.2	19.7
En	23	26.5	21.8	**15.7**	23.0	9.37	19.3
Ib	23	27.3	21.5	19.8	**22.5**	9.71	20.2
Sl	23	26.1	21.2	19.1	**22.5**	**8.69**	19.5
Fused	(115)	**24.9**	18.4	**14.2**	**20.7**	**7.00**	**17.0**
All	115	25.3	**18.2**	16.4	22.0	7.89	18.0

As expected, score-level fusion of the five separate language cluster-specific systems provides the best result on each individual language cluster and overall. The fused system (Fused, 115 hours) also achieves better performance on average and on each individual language cluster except Chinese when compared to the experiment shown in the last row of Table 5.7 in which we pool the 23 hour subsets together and discover a single set of units on all of the languages combined (All, 115 hours). Lastly, comparing the (Fused, 115) and (All, 115) results in Table 5.7 with the (Fused, 315) and (All, 315) results in Table 5.6, we can see that the results are actually quite similar despite an almost three-fold difference in the amount of data used to for acoustic unit discovery. This seems to suggest that the amount of data may not be the primary factor inhibiting performance.

All of the results from Tables 5.6 and 5.7 are obtained via acoustic unit discovery on MFCC features and, despite significantly improving upon the spectral feature baseline result in Table 5.5, are still far from the respective performances of the SWB-BN benchmark and acoustic unit discovery-based system that uses English-inspired SWB-BN features. Our original intention was to ascertain the true effect of language specificity at the spectral feature level; as such, we chose not to use these English-inspired features in our initial investigation. But for completeness, Table 5.8 presents the result of using SWB-BN features for acoustic unit discovery on a per-language cluster basis, and we can see the corresponding performance improvements. In addition to being better than systems in which all languages are pooled together for acoustic unit discovery, our fusion of language-specific systems does even a bit better than the SWB-BN Benchmark on average.

Table 5.8: *Results on LRE15 broken down by language cluster –* ***Ar****abic,* ***Ch****inese,* ***En****glish,* ***Ib****erian, and* ***Sl****avic – the numbers shown are the average detection costs* $C_{avg} \times 100$. *For each of these systems, we represent the audio using English-inspired SWB-BN features and run acoustic unit discovery using a 23 hour subset of the data from the language cluster specified. Using these discovered acoustic units, we build a language recognition system to classify languages from all five language clusters.*

	# hrs	**Ar**	**Ch**	**En**	**Ib**	**Sl**	**Avg**
Ar	23	**20.9**	16.0	15.2	20.3	6.39	15.8
Ch	23	22.1	16.1	15.2	20.3	5.72	15.9
En	23	21.6	15.4	**12.8**	19.2	5.84	15.0
Ib	23	21.4	**15.3**	15.5	**19.1**	5.40	15.3
Sl	23	21.3	16.0	15.6	20.8	**4.66**	15.7
Fused	(115)	**19.5**	**12.9**	**11.2**	**17.6**	**3.53**	**12.9**
All subsets together	115	20.2	14.3	13.1	18.3	3.94	14.0
All languages, all data	315	19.6	13.3	12.7	18.5	3.89	13.6
SWB-BN Benchmark	–	19.6	13.1	**11.2**	18.4	**3.27**	13.1

Our experiment results continue to motivate the need for an improved feature representation for all languages. Our results in Table 5.8 were obtained using transcribed English and did fairly well. However, appropriate features for English may be insufficient for other language clusters. The tonal nature of Chinese, for example, is completely ignored in the standard MFCC-based representation used for English; as such, there remains plenty of room for improvement. As we look ahead to future work, one experiment that would help ascertain the generalizability of our methods is to use a more appropriate feature representation for Chinese, learn acoustic units on those, and evaluate the new system.

5.6 A Phonotactic Perspective

So far, our discussion has been limited to an acoustic approach to language recognition. For a given utterance whose language we would like to identify, we extract an i-vector that summarizes that utterance's distribution of acoustic (spectral or bottleneck) features. Another possible approach to language recognition is a phonotactic approach, which deals with combinations of phonemes [3, 109]. In this section, we discuss our initial experiments on phonotactic language recognition.

5.6.1 Introduction to Phonotactic Language Recognition

Languages can, to some extent, be characterized by their phoneme sets.[10] In linguistics, a phoneme is a basic element of a given language that can be used to build words and distinguish them from one another [3]. Phonotactics deals with the rules governing the possible phoneme sequences in a language. In particular, two languages may share a very similar set of phonemes, but their respective combinations of sounds may differ and thus provide information useful for langauge recognition.

A typical phonotactic approach uses one or more linguistic unit recognizers to tokenize a speech signal into a stream of linguistic units upon which n-gram statistics are extracted. Note that the recognizer can be trained on a language (or languages) that is disjoint from the set of languages that we would ultimately like to identify. These n-gram statistics can be seen as information about the language phonotactics in the utterance and can be used to train statistical language models (or other related representations) for each of the languages in the evaluation set. [3]

In one traditional approach, known as *phoneme recognition followed by language modeling* (PR-LM) [110, 111], we run a phoneme recognizer on training data from each target language and then train a separate n-gram language model for the corresponding language, thus yielding N language models for N target languages in our language recognition task. During the test phase, we can calculate the likelihood of each utterance with respect to each of the target languages by running the phoneme recognizer on the test speech sample and calculating the likelihood of the produced

[10] For a fantastic review of the current state-of-the-art in this field, please refer to [3]; much of this overview is taken directly from it.

phoneme sequence with respect to the corresponding language model. This simulates a situation in which a monolingual person is exposed to utterances from a list of target languages and asked to identify the language of each utterance. [3]

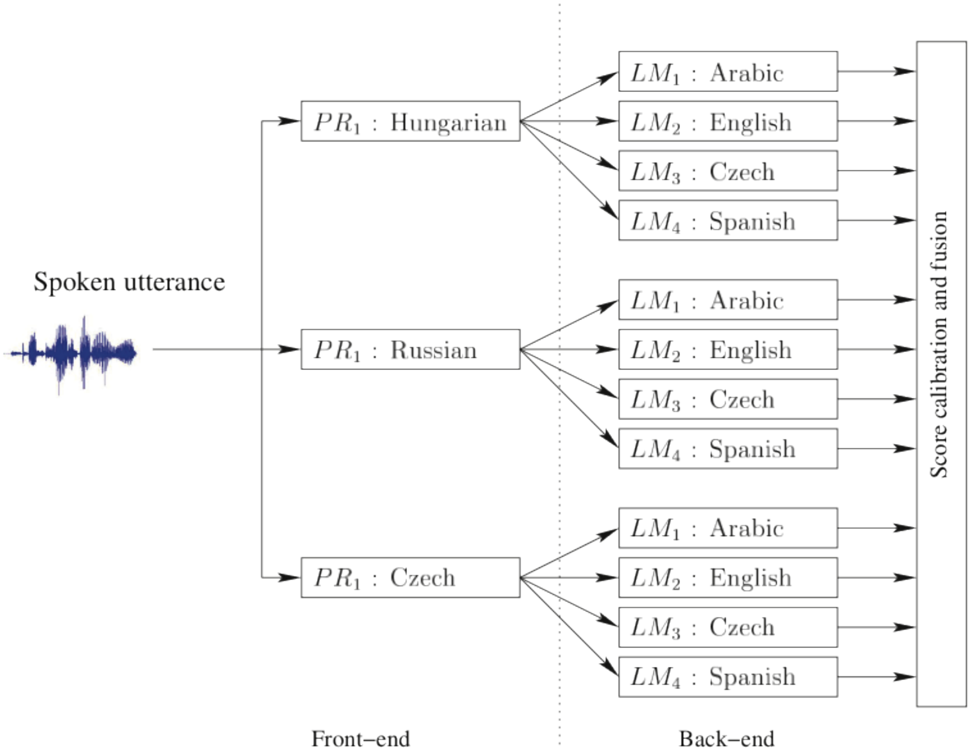


Figure 5-4: *System diagram of parallel phoneme recognition followed by language modeling (PPR-LM). In this case, we have three phoneme recognizers at our disposal (Hungarian, Russian, and Czech), and we build generative n-gram language models for each of our languages of interest (Arabic, English, Czech, and Spanish). (Taken from [3].)*

This analogy can be extended to that of a multilingual individuals, giving us *parallel* PR-LM (PPR-LM), in which each utterance is decoded with multiple phoneme recognizers and their corresponding target language-specific n-gram language models [110, 111]. An example of such a system is illustrated in Figure 5-4. Incorporating more phoneme recognizers built for different languages broadens a system's ability to recognize the different acoustic sequences of all languages. Other attempts to incorporate a better coverage of the various languages include different configurations for multilingual phoneme recognizers [112], as well as the use of acoustic sub-word units without language-specific designations as an alternative to a language-dependent phoneme recognizer [113].

Our work in unsupervised acoustic unit discovery also falls within this framework.

The work in [113] presents a way to obtain a version of our acoustic units – referred to in their work as “acoustic segment models (ASMs)” – by bootstrapping from a set of phone models trained on transcribed multilingual data. Models representing these ASMs are then used to tokenize a test utterance, and a classification decision is made based on its n-gram co-occurrence statistics [113]. In our investigation, we replace their (semi-supervised) ASMs with our discovered acoustic units, which can be obtained in a fully unsupervised or semi-supervised fashion, and run a similar set of experiments to evaluate the efficacy of acoustic unit discovery for phonotactic language recognition.

5.6.2 i-vectors for Phonotactic Language Recogntition

Upon tokenizing a test utterance using a unit/phoneme recognizer, recent approaches to phonotactic language recognition have achieved success accumulating an utterance’s n-gram counts into a fixed length vector, which can then be used as input to a discriminative classifier, such a support vector machine (SVM) [114, 115, 116] or logistic regression (LR). However, because the length of this count vector scales exponentially with the order of the n-gram (usually 3 or 4), recent work has been devoted to obtaining a good compact representation of the original n-gram counts that is computationally feasible in terms of both time and memory. One approach achieved success using principal component analysis (PCA) [115]; a more recent approach uses i-vectors in the form of a subspace multinomial model (SMM) [117].

In Chapter 2, we introduced the i-vector, w, as the posterior distribution of coordinates within a linear subspace, T, that parameterizes an utterance’s distribution of acoustic features in the form of a supervector, M, of stacked GMM component means with respect to some existing UBM, m; that is,

$$M = m + Tw. \tag{5.1}$$

While this formulation is based on continuous features, we can similarly model discrete features under the subspace paradigm [117]. Discrete events can be modeled using a multinomial distribution and, in a way similar to the original i-vector formulation, we can assume that there is a low-dimensional subspace in which the parameters of the multinomial distributions for individual utterances live. Along similar lines, the work in [118] develops a method to model the component weights of a GMM using non-negative factor analysis.

In the following sections, we provide an overview on how to represent an utterance-specific multinomial distribution using a low-dimensional i-vector. First, we assume that all observations are outputs of a single multinomial distribution (i.e., unigram) and derive the subspace multinomial model (SMM) [117]. Then we extend the SMM to allow us to model language phonotactics in a way that is consistent with the n-gram model ($n > 1$); this is known as the subspace n-gram model (SnGM) [119]. A more comprehensive derivation of these models can be found in [3]; our overview offers just the essentials.

The Subspace Multinomial Model (SMM)

In a deliberate abuse of notation, we define for utterance u and unigram $e \in \{1, ..., E\}$ an utterance-dependent model parameter, M_{ue}, representing the probability for the corresponding unigram, as

$$M_e(u) = \frac{\exp(m_e + \mathbf{t}_e \mathbf{w}(u))}{\sum_{i=1}^{E} \exp(m_i + \mathbf{t}_i \mathbf{w}(u))}, \tag{5.2}$$

where $\mathbf{w}(u)$ is an utterance-dependent latent variable, m_e is the e^{th} component of the E-dimensional vector $\mathbf{m}$, and $\mathbf{t}_e$ is the e^{th} row of the subspace matrix, $\mathbf{T}$, which spans a linear subspace in the log-probability domain.

Given parameters $\mathbf{m}$ and $\mathbf{T}$, we can estimate $\mathbf{w}(u)$ to maximize the log-likelihood of utterance u, which is given as

$$\log P(c(u)|M(u)) = \sum_{e=1}^{E} c_e(u) \log M_e(u), \tag{5.3}$$

where $c_e(u)$ is the occupation count for unigram e in utterance u and, correspondingly, $c(u)$ is an E-dimensional vector of observed unigram counts. As defined with the softmax normalization above, the E-dimensional vector $M(u)$ parameterizes a probability distribution over the unigrams in u. A more comprehensive explanation of how these parameters $(\mathbf{m}, \mathbf{T}, \mathbf{w})$ are estimated in iterative fashion using maximum likelihood (ML) estimation is beyond the scope of this overview, but can be found in [3, 117, 120]. This is known as the *subspace multinomial model* (SMM).

This formulation assumes that each element in utterance u is generated independently from a single multinomial distribution (i.e., $\sum_e M_e(u) = 1$), which limits its application to the unigram approximation.[11] A subsequent extension of this work estimates i-vectors that correctly maximize the likelihood of the observed phoneme sequences under n-gram model assumptions for $n > 1$ [119].

[11] In practice, however, this model still achieved relative success when applied to 3-grams [117].

The Subspace n-gram Model (SnGM)

In the n-gram model, we assume that n-grams with the same history, h, are drawn from the same multinomial distribution and n-grams with different histories are drawn from different multinomial distributions. That is, suppose we are given a decoded sequence of units from an utterance $u = \{l_1, l_2, l_3, ..., l_W\}$. Then, in another deliberate abuse of notation, we assume that the conditional distribution of a phoneme l given its history h is a multinomial distribution with parameters M_{hl}; i.e.,

$$\log P(l|h) = \log M_{hl}, \tag{5.4}$$

where $M_{hl} > 0$ and $\sum_l M_{hl} = 1$. Note in this case that $\sum_h \sum_l M_{hl} \neq 1$, whereas in the unigram SMM discussed previously, $\sum_h \sum_l M_{hl} = \sum_e M_e = 1$. The parameters of the corresponding utterance-specific multinomial distribution can be represented as

$$M_{hl}(u) = \frac{\exp(m_{hl} + \mathbf{t}_{hl}\mathbf{w}(u))}{\sum_{i=1}^{E} \exp(m_{hi} + \mathbf{t}_{hi}\mathbf{w}(u))}, \tag{5.5}$$

where m_{hl} is the log-probability of n-gram hl calculated over all the training data, $\mathbf{t}_{hl}$ is a row of a low-rank rectangular matrix $\mathbf{T}$ and $\mathbf{w}(u)$ is an utterance-specific low-dimensional i-vector. Given parameters $\{m_{hl}\}$ and $\mathbf{T}$, we can estimate $\mathbf{w}(u)$ to maximize the log-likelihood of utterance u, which is given as

$$\sum_{i=1}^{W} \log P(l_i|h_i) = \sum_h \sum_l c_{hl}(u) \log M_{hl}(u). \tag{5.6}$$

The EM-like algorithm involved in estimating these parameters in iterative fashion to maximize a regularized form of this likelihood function is discussed in [3, 119]. We apply this subspace n-gram model in our subsequent experiments.

5.6.3 Initial Experiments

A large proportion of recent developments in phonotactic language recognition utilize the Hungarian phone recognizer developed at the Brno University of Technology (BUT) [121] to obtain the necessary n-gram counts for processing [3, 115, 117, 119]. To avoid introducing the additional complexity involved in comparing with another system developed using a completely different set of labeled data, we will continue presenting our results using only the SWB and LRE datasets previously described and note here that our intent is not to present state-of-the-art results in phonotactic language recognition, but to explore the effectiveness of our discovered acoustic units in a phonotactic setting. As such, we present as our baseline a phonotactic system built from the speech recognizer trained on SWB data, which we refer to as **SWB-ASR**. For consistency, this is the same SWB-ASR with which we built a bottleneck i-vector system and obtained the results on acoustic language recognition shown on Row 2 of Table 5.4. Because this recognizer employs a language model tuned for English, this is indeed a weak and unfair baseline; however, our attempts to improve its performance via the use of a null grammar yielded insignificant gains. It is possible that the discrepancy between these baseline results on Row 0 and the rest of those shown in Table 5.9 is the result of a language and recording channel domain mismatch between SWB and LRE data, but further experimentation would be required to substantiate this claim. For now, what we observe is that the English speech recognizer trained on 100 hours of SWB data seems to be an ill-suited tokenizer for phonotactic language recognition.

In our initial experiments, we use the acoustic units discovered on the LRE-subset data represented by our SWB-BN features (Table 5.3) and build various phonotactic i-vector language recognition systems. The most primitive system involves using the context-independent acoustic model obtained from the original acoustic unit discovery and running one iteration of Gibbs sampling on all of the LRE11 data; we will refer to this as **CI sampling**. As a more advanced system, we use the context-dependent, speaker-independent acoustic model derived from these units alongside a null grammar, with which we decode all of the LRE11 data; we will subsequently refer to this as **CD-SI decode**. In both of these systems, we use the resulting unit sequences to train an i-vector extractor based on the SnGM described above. From our 100 acoustic units, we accumulate statistics on $100^2 = 10,000$ bigrams.[12] Furthermore, while better results can be obtained using expected (soft) counts from our decoding lattices [122], we simplify our process and accumulate hard counts from the 3-best paths through our phone lattices.

For comparison, we also use the corresponding 1-best, per-frame senone sequences from the decoded LRE11 data and apply both the SMM to extract i-vectors as well as the PCA-based feature extraction from [115]. Because senones themselves are context-dependent triphone states, they should provide relevant contextual information that is comparable to the bigram model above; we will refer to these systems as **senone SMM** and **senone PCA**, respectively. Table 5.9 displays our initial results; note that the results for all the systems shown in Rows 1-6 are obtained from the same set of acoustic units discovered on the LRE-subset data using SWB-BN features.

[12] Previous works using the BUT Hungarian phone recognizer [121] accumulated $33^3 = 35937$ trigram counts [115, 117, 119].

Furthermore, all systems including the baseline result in Row 0 use exactly the same amount of labeled data.

Table 5.9: *Initial results on LRE11 for various test segment lengths (30, 10, and 3 seconds) using a phonotactic i-vector approach; the numbers shown are the average detection costs, $C_{avg} \times 100$. The first six systems (Rows 0-5) are discussed in Section 5.6.3; the **Acoustic bottleneck i-vector** result is repeated from Table 5.3 in Section 5.4.6.*

		30 sec	**10 sec**	**3 sec**
0	**SWB-ASR / phone seq. / bigram SnGM**	18.6	28.9	40.0
1	**CI sampling / unit seq. / bigram SnGM**	11.4	18.2	31.6
2	**CD-SI decode / unit seq. / bigram SnGM**	10.5	**16.2**	**29.5**
3	**CD-SI decode / per-frame senone / SMM**	9.26	16.7	30.4
4	**CD-SI decode / per-frame senone / PCA**	**8.23**	16.8	30.9
5	**Fusion: 2 + 4**	6.73	13.8	27.4
6	**Acoustic bottleneck i-vector (cf. Table 5.3)**	**2.87**	**7.27**	**18.1**

Despite the myriad ways in which we can improve upon these results – using expected trigram counts, tuning the i-vector dimensionality and SngM regularization parameter, et cetera – it is clear from Table 5.9 that there is a significant discrepancy between the results obtained via any of our phonotactic approaches (Rows 0-5) and the results of our acoustic approach to language recognition (Row 6). Note that the results in Rows 1-5 used SWB-BN features for acoustic unit discovery and are, even with the expert-level knowledge, still significantly worse than the spectral feature baseline acoustic approach (cf. Table 5.2) that uses neither labeled data nor unit discovery.

Nevertheless, a slightly closer look at these results yields a few insights that we can explore further. First, decoding the LRE data using our trained, context-dependent unit recognizer (Row 2) seems to do better than sampling from our mono-unit acoustic model (Row 1). Second, while a simple PCA applied to per-frame senone sequences (Row 4) seems to perform best on the longer, 30-second test segments, the bigram SnGM (Row 2) stands out on the shorter, 10- and 3-second test segments. In light of this, we show a fusion of these results in Row 5 to illustrate the best we can do. However, these results are still far worse than the previous result obtained using acoustic bottleneck features (Row 6).

5.6.4 Exploiting Fusion and Language Specificity

Throughout this chapter, we have observed both the effectiveness of score-level fusion and the importance of language specificity. We should also realize that, with the exception of the fusion result in Row 5, each of the results in Table 5.9 were obtained from just a single PR-LM system, in which we have a "phoneme" (i.e., acoustic unit) recognizer followed by a language modeling scheme in the form of SnGM, SMM, or PCA-based i-vectors. Recall from our introduction in Section 5.6.1 that performance gains in phonotactic language recognition are often obtained via the use of multiple phoneme recognizers in the form of *parallel* PR-LM, or PPR-LM. To this end, we return to the notion of language specificity explored earlier and exploit the LRE data in a way that only our acoustic unit discovery method can:

1. Build a separate acoustic unit recognizer for each of the 24 languages in LRE11;[13]
2. For each language-specific acoustic unit recognizer, accumulate bigram statistics from all of the LRE data and train an i-vector extractor based on the SnGM;
3. Fuse together all 24 language recognition systems at the score level.

Table 5.10 provides a summary of these results. To provide additional clarity, we also report the worst, average, and best results obtained by individual languages for the various test segment lengths. Comparing these results with those in Table 5.9, we can see that the individual phonotactic systems built from specific languages are, at best, still slightly worse than a single phonotactic system built from all languages combined; however, we should also keep in mind that each of these systems was built on only a fraction of the data! Ultmately, a score-level fusion of systems built from individual languages yields immense improvements, thus suggesting once again the importance and benefit of combining different language-specific perspectives.

Table 5.10: *Results of score-level fusion on LRE11 for various test segment lengths (30, 10, and 3 seconds) using a phonotactic i-vector approach; the numbers shown are the average detection costs, $C_{avg} \times 100$. For this experiment, we ran acoustic unit discovery on 10 hours of each of the 24 LRE11 languages, built a unit recognizer for each of them, and then used each recognizer to build a phonotactic language recognition system based on unit bigram counts and i-vectors based on the SnGM.*

	30 sec	**10 sec**	**3 sec**
Worst of Individual Languages	12.9	19.8	33.2
Mean of Individual Languages	11.6	18.3	31.9
Best of Individual Languages	10.3	17.0	30.7
All 24 LRE11 Languages Fused	**6.27**	**9.73**	**22.4**
Acoustic bottleneck i-vector (cf. Table 5.3)	**2.87**	**7.27**	**18.1**

In another experiment, we run the same procedure outlined above, but instead of using the bigram SnGM to obtain i-vectors, we accumulate per-frame senone unigram

[13] We use 10 hours from each language that comprise the LRE-subset data.

statistics from each of the recognizers and simply apply PCA to obtain our i-vectors. We saw in Row 4 of Table 5.9 that this PCA-based method performs relatively well on the longer, 30-second test segments, but worse than the bigram SnGM on shorter test segments. Table 5.11 shows that score-level fusion helps the PCA-based i-vectors obtain comparable performance on shorter test segments and attain even better performance on the 30-second test segments. The effectiveness of a simple PCA-based approach relative to that of the SMM and SnGM is somewhat surprising; we defer to future work a more in-depth analysis of the differences between these i-vector models (PCA, SMM, SnGM) for phonotactic language recognition using discovered acoustic units.

Table 5.11: *Results of score-level fusion on LRE11 for various test segment lengths (30, 10, and 3 seconds) using a phonotactic i-vector approach; the numbers shown are the average detection costs,* $C_{avg} \times 100$. *In this experiment, we ran acoustic unit discovery on 10 hours of each of the 24 LRE11 languages, built a unit recognizer for each of them, and then used each recognizer to build a phonotactic language recognition system based on unigram senone counts and PCA-based i-vectors.*

	30 sec	**10 sec**	**3 sec**
Worst of Individual Languages	11.3	21.6	34.7
Mean of Individual Languages	10.2	20.2	33.4
Best of Individual Languages	9.24	19.1	32.5
All 24 LRE11 Languages Fused	**4.24**	**9.72**	**23.4**
Acoustic bottleneck i-vector (cf. Table 5.3)	**2.87**	**7.27**	**18.1**

5.6.5 Towards Accumulating DNN Softmax Posteriors

As an alternative to phoneme recognizer-based phonotactic approaches, recent work has found success in accumulating the softmax posteriors at the output layer of a DNN over each frame of an entire utterance and using the resulting "expected senone counts" vector as a feature from which to extract SMM-based i-vectors [106]. In this section, we adopt the same approach using the DNNs trained on acoustic unit and present our results in Table 5.12. Each of the systems depicted uses the exact same DNN trained to obtain its respective result in Table 5.3.

We can see that the results in Table 5.12 are indeed better than the results of more traditional phonotactic approaches from Table 5.9; however, they are still significantly worse than the acoustic approaches shown in Table 5.3. In fact, the **Transcribed SWB Benchmark** result from both Tables 5.12 and 5.3 represent, to some extent, the supervised upper bounds on the performance potential of phonotactic and acoustic approaches, respectively, since the DNN is trained using transcribed English and

Table 5.12: *Results on LRE11 for various test segment lengths (30, 10, and 3 seconds) using SMM-based i-vectors extracted from expected senone counts obtained by accumulating the DNN softmax over each frame of an utterance; the numbers shown are the average detection costs, $C_{avg} \times 100$. Each of the systems depicted below uses the exact same DNN trained to obtain the results in Table 5.3.*

	30 sec	**10 sec**	**3 sec**
AUD(LRE-subset, MFCC) SD	8.69	14.7	25.5
AUD(LRE-subset, SWB-BN), SI	5.98	12.9	25.6
Transcribed SWB Benchmark	**4.81**	**10.5**	**23.2**

involves no acoustic unit discovery. The significant gap in performance isn't surprising, as acoustic approaches to language recognition have surpassed phonotactic approaches in recent years. The typical strategy for a NIST LRE submission is to fuse together a select subset of these systems that covers both approaches [106].

Finally, the performance gaps between the rows of Table 5.12 are readily apparent. The final row portrays the benefit of expert linguistic knowledge, while the first row represents the outcome of acoustic unit discovery given no initial supervision. The middle row, which uses SWB-BN features for acoustic unit discovery, presents results in between. Yet if SWB-BN features are based on expert linguistic knowledge and acoustic unit discovery can purportedly adapt to new and unseen languages, we should expect to see results that are an improvement to those shown in the final row. As it stands, acoustic unit discovery actually makes matters worse. While we have demonstrated the ability to achieve gains via fusion and language specificity, there remains much to be done to improve the acoustic unit discovery process (and its input feature representation) in such a way that it can ascertain more salient linguistic knowledge. We embark upon this investigation in the following chapter.

5.7 Future Work

As we have commented throughout this chapter, there exist many avenues for future work. Apart from a cursory visual inspection, we have not done much to measure, using existing metrics [123], the quality of the discovered acoustic units using such a large set of multilingual data. To bridge the gap between running unit discovery on MFCCs and discovery acoustic units on SWB-BN features, we could consider incorporating autoencoders of stacked MFCC features, such as in [124], or adopting a bootstrapping approach, in which we learn an initial set of units on MFCCs, train a DNN on those targets, and use the resulting bottleneck features as input to another round of acoustic unit discovery, and so on. We can also update our DNN architecture to account for language specificity at the output layer; for example, each target class could be a language-specific senone where, during training, we compute our softmax only for the senones pertinent to the current language [125, 126]. Lastly, it might be useful to consider developing a new architecture altogether that can integrate the acoustic unit discovery, context-dependent modeling, and bottleneck feature extraction into a single, data-driven process.

Continuing to explore different language-specific perspectives is certainly ripe with possibilities, especially in the context of LRE15 and our observations with the English and French language clusters. Another easy extension would be to consider the use of different acoustic features for different languages – for example, we mentioned earlier that running unit discovery using MFCCs is far less meaningful on tonal languages such as Chinese. If we were to view the notion of language specificity as a weak form of supervision, then it is evident that even just a little supervision of any form and from any language can be extremely useful. There are many ways to incorporate this that can still be explored, including equivalence constraints on acoustic sequences (e.g., examples of the same word) [127, 128, 129].

5.8 Summary

In this chapter, we explored the application of large-scale acoustic unit discovery as a tool to obtain features for language recognition. To do so, we implemented a parallelized version of a Bayesian nonparametric model from previous work, which let us scale the acoustic unit discovery process from operating on 10 hours of data to discovering acoustic units on hundreds of hours of multilingual data. We use these per-frame sequences of units (or states or senones) as targets to train a DNN that, given stacked spectral features as input, provides a bottleneck feature representation that can be used for i-vector language recognition. We found that a score-level fusion with a baseline system built from acoustic features yields substantial gains and significantly closes the gap between the baseline and a benchmark system built using transcribed English, suggesting that discovered acoustic units may be complementary to spectral features. Subsequently, we also found that using an improved representation of speech (i.e., supervised bottleneck features) as input to our acoustic unit discovery system can yield substantial performance gains, thus motivating the need for a better understanding of the speech signal. We validated the generalizability of our proposed approach by presenting results that exhibit similar trends on the LRE's from both 2011 and 2015. Finally, we demonstrated on LRE15, as well as in our attempts at phonotactic language recognition, the continued importance and efficacy of different language-specific perspectives for unit discovery.

In the following chapter, we dig a little deeper into possible improvements for our acoustic unit discovery process. To this end, we return to a monolingual setting and attempt to modify the original, fully unsupervised method of acoustic unit discovery such that it can take into account weak constraints. These constraints may come in the form of pairs or clusters of repeated words, phrases, or even sentences. While an appropriate segmentation and clustering still needs to be learned for each of these utterances, we hope this form of weak supervision can bring forth the discovery of more linguistically meaningful sub-word units.

Chapter 6

Improving Acoustic Unit Discovery: Analysis and Experiments

Previously in Chapter 5, we successfully applied acoustic unit discovery to language recognition and, in doing so, observed a fair amount of room for improvement in our ability to ascertain linguistic knowledge in an unsupervised (or semi-supervised) manner. We saw that an improved feature representation in the form of supervised DNN bottleneck features can yield more salient acoustic units for language recognition; however, these acoustic units were unable provide an improvement over the straightforward application of those supervised bottleneck features, suggesting that there are still many linguistic properties that are not captured by either the unit discovery model or the bottleneck features.

In this chapter, we analyze the behaviors of our acoustic unit discovery method and explore various ways to improve the quality of our discovered units in the linguistic sense. To this end, we return to a monolingual setting and exploit the human-labeled ground-truth conveniences of the TIMIT corpus [99] to run our initial experiments. Instead of using language recognition performance as our primary metric, we will use various methods to evaluate the segmentation and clustering returned by our unit discovery model and proposed modifications. These metrics will include a combination of boundary F_1 score (for segmentation), normalized mutual information (for clustering), word (or phone or unit) error rate (for both), and visual inspection. Our hope is that a closer look at the intricacies of the entire process may yield insights into how linguistic knowledge can be better ascertained from data.

6.1 Desiderata

Our ultimate goal is to modify the original, fully unsupervised method of acoustic unit discovery so that it can take into account weak constraints, such as equivalent realizations of words or sentences. For example, Figure 6-1 shows spectrograms for two instances of the same speaker (from Switchboard) saying the word "really;" the phonetic transcription can be found in the green box. We can see the result of our unit discovery run on SWB-BN features: underlined in blue are the parts of the unit sequence that match across both utterances, while highlighted in red are the segments that do not match. We can also see that the segmentation boundaries differ between the transcription and the result of our unit discovery. In particular, if there had been a hypothesized boundary somewhere within unit 5 on the lower spectrogram, perhaps we could have obtained a clustering that matches the 70-90 unit sequence on the upper spectrogram. That said, we should note that the two spectrograms actually look rather different; indeed, it may not be obvious to an untrained eye that these two segments are instances of the same word. It is quite remarkable that so much of the unsupervised segmentation is identical.

During the supervised training of a speech recognizer, the system is given an utterance and a corresponding phonetic sequence of finite length. Via a process known as *forced alignment*, the system is then tasked with fitting each frame of the utterance to the sequence of HMM states corresponding to the provided phone sequence, thus yielding a phonetic segmentation for the utterance. Recall from Section 5.2.6 that this is also how our unit recognizer training process modifies the original landmark boundaries specified by our unit discovery Gibbs sampling. The difference between traditional speech recognizer training and our unit recognizer training process, however, lies in the origin of these phone/unit sequences.

In the case of the latter, the unit sequences provided for unit recognizer training are the result of sampling from a generative model; as such, outside of the provided acoustic features and assumed (HMM/GMM) model parameters, the unit discovery process is otherwise unconstrained and, by definition, unsupervised. For example, if the respective features of a male `/ah/` and a female `/ah/` look different enough, then it is perfectly reasonable, from the point of view of the generative model, to have two separate acoustic units for a female `/ah/` and a male `/ah/`. On the other hand, the transcriptions provided in the fully-supervised training of a speech recognizer will likely require that an `/ah/` spoken by any speaker be modeled with the same unit. This encourages the `/ah/` model to take into account gender and speaker variability, thus making it more speaker independent.

Our goal in this work is to encourage the learning of more speaker-independent

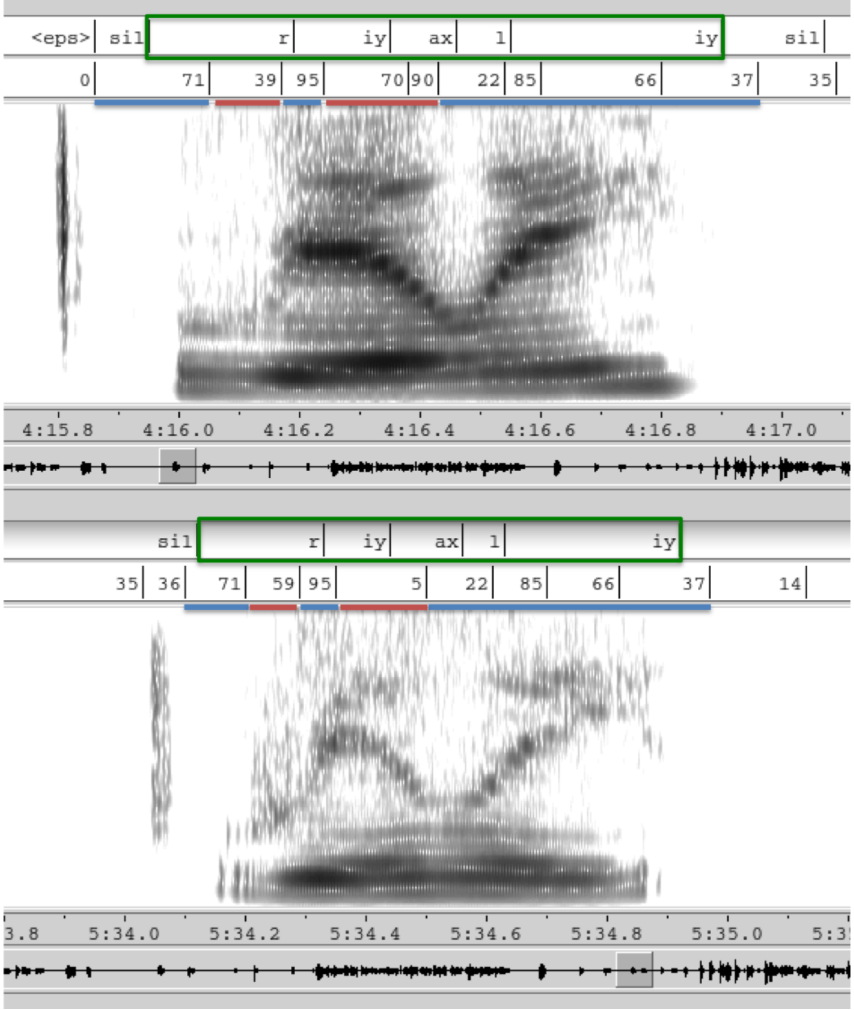


Figure 6-1: *Two spectrograms for two instances of the same speaker saying the word "really." The phonetic transcription can be seen in the green box. Underlined in blue are the parts of the discovered unit sequence that match across both utterances; highlighted in red are the segments that do not match.*

or, at the very least, instance-robust acoustic units. Since both instances of "really" in Figure 6-1 come from the same speaker and occur in isolation from potential coarticulating effects, it would be especially nice to be able to discover the same resulting unit sequence. At the end of the day, we would like to impose the same form of consistency that a supervised phonetic transcription might, but we would like to do so without the use of a lexicon or pronunciation dictionary. Instead, we imagine a scenario in which a user (or some oracle) points out clusters of two or more speech segments (i.e., utterances) that correspond to the same word (or phrase or sentence) and use this knowledge to guide our unit discovery process.

This notion of weak, cluster-based constraints likens back to our explorations using pairwise speaker labels in Chapter 4. The scenario here is a little more involved in that we still need to obtain a segmentation and clustering at the sub-word unit level. Our weak constraints here simply serve to encourage our acoustic unit discovery to learn the same sequences for sets of corresponding utterances;[1] the actual unit sequences themselves still need to be determined!

Other approaches to guided (i.e., semi-supervised) unit discovery have been explored in [127] and, more recently, in [128, 129]. The work in [127] used spoken term discovery to find pairs of repeated words. For each pair, they perform a dynamic programming alignment of the acoustic observations, thus generating a correspondence between cross-speaker frame pairs that can be used to partition the Gaussians of some GMM into sub-word unit models that exhibit more speaker independence. Subsequent work in [128, 129] uses these correspondences to learn a better neural network-based feature representation of the audio.

Given the relative success of pairwise constraints in [127, 128, 129] in guiding the partitioning of the mixtures of a GMM to obtain more speaker-independent acoustic units, we aim to explore the learning of sub-word unit HMMs (i.e., acoustic units) in a similar fashion. Instead of correspondences between pairs of individual frames, we consider correspondences between sub-word segments and use them to build speaker-independent HMMs. Enforcing such constraints, however, requires a more detailed understanding of each stage in the unit discovery process and the effect of perturbations at each stage. As segmentation and clustering are obviously inter-related, our subsequent explorations will separately consider the impact of segmentation on the resulting clustering, and vice versa. That is, we will begin by analyzing the effect of clustering using these pairwise constraints while assuming the existence of a consistent segmentation. Then we will explore ways to obtain a robust segmentation that utilizes these constraints.

6.2 Data

Because of its implementational convenience, we return to the TIMIT corpus used in [5] for the experiments in this exploration [99]. TIMIT is a corpus of read speech containing broadband recordings of 630 speakers of eight major dialects of American English, each reading ten phonetically rich sentences. Each utterance has been transcribed and verified, and we use these labels as a ground truth on which to base our

[1] In future work, we will also explore the effect of difference constraints; i.e., encouraging the learning of different unit sequences that can distinguish between pairs of utterances that contain different spoken content.

comparisons [99]. Our work will utilize the corpus-specified training, development (dev), and testing subsets in various ways that we will explicitly detail for each experiments. The speakers present in each of the three subsets are disjoint, and none of the sentences recorded in the train subset are present in the dev or test subsets. However, the dev and test subsets share a few realizations of the same sentence recorded by different speakers; we will specify how they are used in our experiments.

The entirety of TIMIT can be broken down into three different types of sentences: the **sa** sentences recorded by all speakers; the **si** sentences are each read by two different speakers, and the **sx** sentences are each spoken by seven different speakers. None of the recorded **sa** sentences are used in the corpus-specified train, dev, or test sets. The train subset contains 8 recordings (3 **si** and 5 **sx**) from each of 462 speakers (326 male and 136 female) – yielding a total of 3696 utterances – which can be broken down as 1 recording for each of 1386 distinct **si** sentences and, more importantly, 7 separate recordings for each of 330 distinct **sx** sentences (2310 total utterances). We can see these **sx** sentences as providing *text-equivalent* constraints, in which seven different speakers provide realizations of the same text.

It is extremely important to note, however, that even though a sentence may contain the same underlying words, it may not have the same underlying phonetic transcription. For example, one of the most well-known sentences in TIMIT is **sa1**, which reads, "She had your dark suit in greasy wash water all year." While this is not a part of the specified training subset that we will use, this is a sentence that all 462 speakers recorded, and the expert-level transcription of these sentences resulted in 458 unique phonetic sequences. In only four instances was there a recording whose pronunciation was exactly the same as that of another speaker!

Of course, this is largely due to utterance length and thus breaks down when we begin to consider speech segments within these utterances. For instance, the words "dark suit" has 102 instances (out of 462) of the same pronunciation: `/d aa r kcl k s ux tcl t/`. Another popular alternative pronunciation (72 instances) is `/d aa r kcl k s ux dx/`, where the unvoiced `/tcl t/` from the first sequence is replaced with continued voicing in the form of a flap `/dx/` in anticipation of the vowel at the start of the next word, "in." To this end, we must stress that our goal is not to force our acoustic units to marginalize out pronunciation variability across speakers or instances. This would be extremely dangerous – confusing or combining models of inherently different phonemes due to pronunciation variablity would be detrimental to our goal of ascertaining linguistic insight for any language. Rather, our intent is to enforce constraints on utterances that we know contain the *same* pronunciation (i.e., sequence of phonemes) to increase the speaker independence (and instance-robustness) of our acoustic units; we will refer to these as *pronunciation-equivalent* constraints.[2]

In some of our subsequent experiments, we limit ourselves to just the pronunciation-equivalent (**pron-eq-train**) subset of the training data. This set of 66 utterances contains 28 distinct sentences, where each sentence has between 2 and 6 recordings that feature the exact same pronunciation sequence from different speakers. We realize that the size of this data set is quite small and that it is possible to obtain a larger set of sub-sentences (i.e., words or phrases) from the many utterances of **sx** sentences, but we feel that this is a reasonable place to begin our initial explorations.

6.3 A Broader Perspective

Previously, the work in [130] proposed a model of unsupervised phonological lexicon discovery to learn phoneme-like and word-like units from acoustic input. Building upon the acoustic unit discovery work in [5], this model integrates an Adaptor Grammar framework [131] that allows for the unsupervised learning of lexicons from unsegmented symbolic sequences. Bridging the gap between the sequences of discovered acoustic units and the Adaptor Grammar, however, required the use of a noisy-channel model to map variations in speech production – including differences not only in pronunciation but also in stress pattern, phonetic context, et cetera – to a more unique representation [130]. In the context of the work in this chapter, we propose the use of *pronunciation-equivalent* constraints to improve the low-level acoustic model in a way that makes the task more straightforward for the noisy-channel model. In a subsequent exploration, it would be interesting to explore the use of the *text-equivalent* constraints mentioned above as a form of weak supervision to aid the adaptor grammar in its lexicon discovery process.

6.4 Metrics and Baselines

In this work, our goal is to discover acoustic units subject to a set of weak constraints. As such, we need a set of metrics that can measure both acoustic unit quality with respect to some ground truth, as well as some notion of consistency within our pronunciation-equivalent utterances. This sort of a tradeoff is necessary, as one could imagine satisfying the consistency constraint trivially with discovered

[2] From a practical point of view, this methodology begets the somewhat circular question of how we can guarantee without expert-level knowledge (e.g., transcribed TIMIT) that these imposed constraints satisfy our assumptions – for now, we defer this to future work.

acoustic unit sequences that are all the same.[3] To this end, we will report our results in terms of both normalized mutual information (NMI) and pronunciation-equivalent unit error rate (PE-UER), as described below. Finally, in our subsequent consideration of finding improved segmentation boundaries, we will revisit the boundary precision, recall, and F_1 score that were also used in [5]. While there are a number of other ways to measure differences between sets of sequences [54], as well as a few other existing metrics for measuring the quality of discovered acoustic units [87, 123], we believe this is a reasonable place to start for our initial explorations.

In our experiments, we will report these metrics as measured on the TIMIT test set. While unsupervised methods are, for the most part, agnostic to the strictness of train/dev/test protocols, our work now does involve some form of supervision, albeit weak. As such, we will discover our acoustic units on the train set and, where specified, enforce our weak constraints using the **pron-eq-train** subset of the training data. Then we will use the resulting model to sample a single alignment from the TIMIT test set and evaluate the NMI of that result against our ground truth.[4] We can also evaluate the consistency of the model by computing the PE-UER from the pronunciation-equivalent subset of the test data (**pron-eq-test**). Lastly, computing segmentation boundary precision, recall, and F_1 score are similarly straightforward.

6.4.1 Normalized mutual information

A robust clustering approach aims to strike a balance between cluster purity and fragmentation. The purity of a hypothesized cluster can be seen as the maximal proportion of the cluster that represents a single reference entity, while the fragmentation of a reference entity can be seen as the number of clusters needed to represent it. Trivially, a high average cluster purity can be achieved by assigning each element to its own cluster; conversely, a low entity fragmentation can be achieved by assigning all elements to a single cluster. While there exist many ways in which to express the balance between these two measures [54], we choose to compute the normalized mutual information (NMI) between the per-frame phone sequences given by the ground truth annotation and the corresponding per-frame unit sequences discovered automatically. Formally, for reference sequence, R, and hypothesized sequence, S, we have

$$\mathrm{NMI}(R, S) = \frac{I(R, S)}{\frac{1}{2}(H(R) + H(S))}, \tag{6.1}$$

[3] In reference to Footnote 1 above, this motivates once again the use of difference constraints.

[4] In practice, so long as we don't update the model using the alignments sampled from the test data, we can evaluate on multiple samples and average the results for a more statistically significant evaluation.

where $I(R,S)$ denotes the mutual information between R and S, and $H(R)$ and $H(S)$ are their respective marginal entropies. For completeness,

$$I(R,S) = \sum_{r \in R} \sum_{s \in S} p(r,s) \log \left(\frac{p(r,s)}{p(r) \cdot p(s)} \right), \tag{6.2}$$

and

$$H(X) = -\sum_{x \in X} p(x) \log(p(x)). \tag{6.3}$$

The mutual information, $I(R,S)$, can be seen as a measure of the information that S gives us about R, and vice versa. Normalizing the mutual information with their respective entropies scales the range of NMI to be between 0 and 1.

6.4.2 Pronunciation-equivalent unit error rate

We derive our measure of pronunciation-equivalent unit error rate (PE-UER) from the traditional word error rate (WER) metric typically used to evaluate the quality of speech recognition systems. Given a reference transcription, r, and a hypothesized word sequence, h, the WER is simply the number of word substitutions (S), deletions (D), and insertions (I) required to obtain the reference sequence from the hypothesis normalized by the length of the reference:

$$\mathrm{WER}(r,h) = \frac{\mathrm{S} + \mathrm{D} + \mathrm{I}}{|r|}. \tag{6.4}$$

Instead of words, we can use phones to obtain a phone error rate or, in our case, a unit error rate (UER) for discovered acoustic units. To obtain a PE-UER for our set of discovered acoustic sequences in TIMIT, we compute the average UER for each unit sequence u_P against all the other unit sequences in its pronunciation-equivalent subset, P, and average this across all pronunciation-equivalent subsets $P \in \mathbf{E}$. That is,

$$\text{PE-UER}(\mathbf{E}) = \frac{1}{|\mathbf{E}|} \sum_{P \in \mathbf{E}} \frac{1}{|P|} \sum_{u \in P} \frac{1}{|P|-1} \sum_{w \in \{P \setminus u\}} \mathrm{UER}(u,w), \tag{6.5}$$

where $\mathbf{E}$ is the set of distinct sentences whose set of realizations, P, contain two or more equivalent pronunciations. Each $u \in P$ gets used as a reference in the $\mathrm{UER}(u,w)$ computation and is compared to all the other sequences $w \in \{P \setminus u\}$. From our earlier discussion of the **pron-eq-train** data subset in Section 6.2, $|\mathbf{E}_{\text{train}}| = 28$ and $|P_{\text{train}}| \in [2,6]$. And for evaluation on the **pron-eq-test** subset of our TIMIT test set, $|\mathbf{E}_{\text{test}}| = 5$ and $|P_{\text{test}}| = 2, \forall P_{\text{test}} \in \mathbf{E}_{\text{test}}$. Of course, our benchmark using the ground truth TIMIT annotations is PE-UER($\mathbf{E}$) = 0.

6.4.3 Evaluating segment boundaries

We assess the quality of our segmentation boundaries using the precision, recall, and F_1 score metrics explored in [5]. Here, we ignore cluster labels and simply measure the presence or absence of segmentation boundaries. Precision is defined as the fraction of hypothesized segment boundaries that correspond to reference segment boundaries as given by the ground truth annotation. Recall is the proportion of reference segment boundaries that have a corresponding hypothesized segment boundary. The F_1 score is then defined as the harmonic mean of precision, P, and recall, R:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{6.6}$$

6.5 Clustering

In this section, we consider the problem of clustering assuming a fixed segmentation. In particular, assuming that we are provided with a meaningful segmentation, how consistent is our resulting clustering? To assess this, we use a segmentation derived from TIMIT's ground truth annotations and run the same acoustic unit discovery system as described in Chapter 5, but with a single modification: we enforce a fixed segmentation by setting strong priors on the boundary variables; i.e., $P(b_t = 1) = 1$ if the t^{th} frame corresponds to a ground truth segment boundary, and $P(b_t = 1) = 0$ otherwise. This causes our unit discovery system, which we run on MFCC features for all experiments in this chapter, to focus solely on the clustering of segments. The result of this approach can be seen in Row 1 of Table 6.1, denoted **TIMIT segmentation, unconstrained clustering**. Note that we have not yet used any information about pronunciation-equivalence, but both the NMI and mean PE-UER are already substantially better than the results of the baseline system shown in Row 0, which were obtained by running the exact same acoustic unit discovery system as described in Chapter 5. This demonstrates the impact of a consistent segmentation.

6.5.1 Incorporating sequence constraints

We take into account our pronunciation-equivalence constraints in the accumulation of statistics during the HMM-GMM model update step of the unit discovery process. Given utterance, u, and a corresponding pronunciation-equivalent utterance, w, we further assume that w has a corresponding segmentation such that each of the

Table 6.1: *Boundary F_1 scores, normalized mutual information, and pronunciation-equivalent unit error rates (PE-UER) obtained by our various acoustic unit discovery configurations. Both F_1 and NMI are evaluated against the respective ground truth annotations of our TIMIT train and test subsets, while the respective PE-UER values shown are evaluated on the 28-sentence, 66-utterance* ***pron-eq-train*** *subset and the 5-sentence, 10-utterance* ***pron-eq-test*** *subset described in Section 6.4.2. Our results here examine the effect of different clustering configurations assuming a fixed segmentation obtained from the TIMIT ground truth annotations.*

		$\mathbf{F_1}$		**NMI**		**PE-UER (%)**	
		train	test	train	test	train	test
0	**Unconstrained segmentation and clustering (Baseline)**	0.856	0.860	0.361	0.388	43.1	35.7
1	**TIMIT segmentation (fixed), unconstrained clustering**	1	1	0.554	0.575	32.8	26.7
2	**TIMIT segmentation (fixed), constrained clustering**	1	1	0.561	0.584	28.1	25.0
3	**TIMIT transcription (Benchmark)**	1	1	1	1	0.0	0.0

subsegments in u has a corresponding subsegment in w. Then during inference (at train time), after we sample a unit sequence for u, we can update our models by accumulating statistics from both u **and** w as though we had sampled the *exact same* unit sequence on w as we did on u. In practice, we use the unit sequence sampled from u to sample a corresponding HMM state sequence on w and update our models using the accumulated statistics from the sampled state sequences of both u and w. During each pass through the data, we sample $u \in P$ uniformly at random and then accumulate additional statistics for all $w \in \{P \setminus u\}$.

One of the limitations of this method is that it will only work if every utterance in P has a respective segmentation that corresponds exactly with one another. While this is not an issue for the segmentation derived from TIMIT's ground truth annotations, we consider ways to automatically obtain a consistent segmentation in Section 6.6. Until then, Row 2 of Table 6.1, denoted **TIMIT segmentation, constrained clustering**, shows that enforcing constraints on the **pron-eq-train** subset of the training data (i.e., the 66 pronunciation-equivalent utterances from the 27 distinct sentences out of 3696 total utterances of 1716 distinct sentences) can still improve our NMI and PE-UER compared to the unconstrained clustering shown in Row 1 and described above, thus suggesting that this constrained accumulation of state sequence statistics really can increase the speaker independence of our acoustic unit models, at least to some degree. Nevertheless, the magnitude of this impact is still rather small compared to the effect of simply switching from our landmark segmentation (Row 0) to the TIMIT-based segmentation.

6.5.2 Varying the composition of the training data

We should expect that the impact of constrained clustering depends on the size of the **pron-eq-train** subset relative to the rest of the training set. While Table 6.1 shows the slight positive effect of a **pron-eq-train** subset that is less than 2% of the total training set, Table 6.2 offers a glimpse of what might happen with training sets composed in different ways. In these experiments, we only vary the composition of the training data; the results shown are still obtained from evaluating on the full TIMIT test set and, as such, are comparable to those in Table 6.1. Because each system will feature a separate subset of training data, we will focus on just the test NMI and PE-UER, as all F_1 scores are 1 due to the use of a fixed segmentation obtained from the TIMIT ground truth annotations.

In this subsection and Table 6.2, we use as our new baseline the results of acoustic unit discovery on just the **pron-eq-train** subset of the training data – this is shown on Row 0. From here, we introduce additional training data along two dimensions: speaker variability and phonetic variability. Note that this additional training data is not subject to any constraints; these utterances do not have any pronunciation equivalents. In the unconstrained and unsupervised setting, this does not matter, as additional data simply gets modeled along directions of maximal variance in the feature space. We will see in the constrained setting, however, that this can affect results.

Row 1 of Table 6.2 shows the result of introducing speaker and pronunciation variability without introducing additional phonetic content. In particular, we use only the sentences present in **pron-eq-train**, but we include *all* realizations of those sentences. This yields 189 utterances of 28 distinct sentences spoken by 169 different speakers. While the unconstrained result is largely unaffected by the additional data and perhaps even offers a minor improvement, the PE-UER for the constrained data increases from our previous result in Row 0. This may be because the added unconstrained data is quite similar to that of the constrained data, and because no phonetic variability is introduced, some of the acoustic units that are not subject to pronunciation-equivalence constraints begin to model speaker-specific differences.

On the other hand, we see in Row 2 the result of introducing additional phonetic content without introducing additional speaker variability. In this system, we include *all* sentences spoken by each of the speakers present in **pron-eq-train**. This yields a substantial increase in the amount of training data, which explains the increase in NMI for both the constrained and unconstrained clustering scenarios. What is even more profound, however, is the resulting decrease in PE-UER of the constrained clustering, which seems to suggest that increasing within-speaker training data can potentially lead to the learning of more consistent, speaker-independent acoustic units. While additional experiments are necessary to make this notion more concrete, this may point to better ways to utilize data for more robust acoustic unit discovery.

Table 6.2: *NMI and PE-UER obtained by our various acoustic unit discovery configurations. NMI is obtained against the ground truth annotations of our TIMIT test subset, while the PE-UER values shown are evaluated on the 5-sentence, 10-utterance* ***pron-eq-test*** *subset described in Section 6.4.2. Our results here examine the effect of differences in training data composition and unconstrained vs. constrained clustering for acoustic unit discovery assuming a fixed segmentation obtained from the TIMIT ground truth annotations.*

		NMI		**PE-UER (%)**	
		unconst.	const.	unconst.	const.
0	**pron-eq-train only**	0.476	0.515	26.4	25.1
1	**pron-eq-train + spkr-var**	0.522	0.528	24.2	26.3
2	**pron-eq-train + phn-var**	0.561	**0.568**	26.3	**21.3**

6.5.3 The effect of different numbers of units

Clustering quality is inherently dependent on the number of units discovered. Since this is a parameter specified as input to our acoustic discovery model, we should examine the effect of varying the number of discovered units on both constrained and unconstrained clustering. A quick look at Figure 6-2, however, seems to suggest that the number of units learned is far less significant than the quality of the initial segmentation. As such, we defer to future work further optimization of the number of units to learn and consider ways to improve our initial segmentation.

6.6 Segmentation

We saw in Figure 6-2 and in the results from the previous section that a linguistically relevant initial segmentation (from TIMIT ground truth annotations) helps significantly in the discovery of better acoustic units. In this section, we explore techniques to automatically obtain an initial segmentation given our set of text-equivalent constraints. We make the distinction here that our goal is not necessarily to obtain at a perfect segmentation from the point of view of the TIMIT ground truth; instead, our

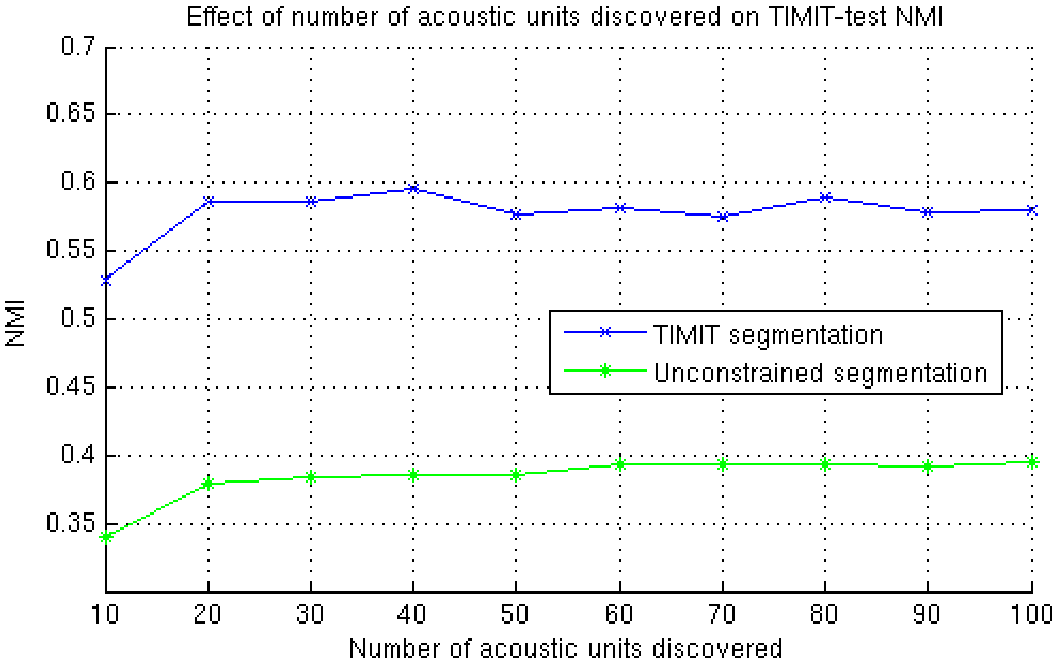


Figure 6-2: *A plot of NMI on the TIMIT test subset versus number of units learned. Here, we sweep across the number of units learned for two different segmentations: the TIMIT ground truth segmentation and the unconstrained, landmarks-based segmentation from Chapter 5. While the number of units may make a minor difference, it seems as though segmentation quality is a much more significant factor.*

goal is to obtain a form of consistency within our segmentation that can yield similarly consistent acoustic units. Following the work in [127, 132], we aim to utilize a dynamic programming alignment of text-equivalent utterances to find candidate segmentation boundaries that correspond across all utterances of the same text. Ideally, these resulting segmentations will also yield an exact correspondence that allows us to incorporate the sequence-constrained accumulation of model statistics as discussed in Section 6.5.1.

6.6.1 Parameter marginalization

So far, our experiments in acoustic unit discovery focused on the use of a pre-segmentation algorithm developed in [98, 102]. As mentioned in Section 5.2.5, this method essentially hypothesizes phonetic boundaries where the difference in spectral energy is large in magnitude. Despite its conceptual simplicity, the inherent variability of speech makes it extremely difficult to guarantee a consistent segmentation across all utterances, even if they contain the same spoken content. Furthermore, there are a number of parameters that can be adjusted in this landmark segmentation algorithm, and our work in Chapter 5 focused solely on the output of a single set of parameters tuned separately on a set of development data.

Here, we consider the effect of marginalizing over the entire set of landmark segmentation parameters, Θ, and obtaining, for each frame in a given utterance, an estimate of its likelihood as a landmark boundary. That is,

$$\hat{p}(b_t = 1) = \frac{1}{|\Theta|} \sum_{\theta \in \Theta} \delta(b_t; \theta), \tag{6.7}$$

where the indicator variable $\delta(b_t; \theta) = 1$ if a boundary at frame t is hypothesized under parameter setting θ, and $\delta(b_t; \theta) = 0$ otherwise. The idea is that different sets of parameters may yield different segmentations, but the more obvious boundary locations should be fairly persistent across many parameter settings.

6.6.2 Pronunciation-dependent utterance marginalization

While the method described above yields a segmentation that obtains an F_1 score that is not significantly better than that of our original, unconstrained baseline (Row 0, Table 6.1), the benefit is that we can extend the probability distribution in Eqn. 6.7 to additionally marginalize over all pronunciation-equivalent utterances. In particular, suppose we chose an exemplar utterance $u \in P$ and mapped the frames of all the other utterances, $w \in \{P \setminus u\}$ to u via the Dynamic Time Warping (DTW) algorithm [133]. Then we can write

$$\tilde{p}(b_t^{(u)} = 1) = \frac{1}{|P|} \sum_{w \in P} \frac{1}{|\Theta|} \sum_{\theta \in \Theta} \delta(b_{t'}^{(w)}; \theta), \tag{6.8}$$

where $t' = \mathrm{DTW}_{u \to w}(t)$ is the frame index of w corresponding to frame t in utterance u under a DTW alignment.[5]

Given these aggregated statistics, we can choose a segmentation for u from $\tilde{p}(b_t^{(u)}), \forall t$ via a local peak-picking algorithm and subsequently obtain a segmentation for the rest of $w \in \{P \setminus u\}$ using their corresponding DTW alignments. An example of this segmentation can be seen in Figure 6-3, which shows spectrograms of a male (top) and a female (bottom) speaker saying the TIMIT sx390 sentence, “He picked up nine pairs of socks for each brother.” Above each spectrogram is an enumerated segmentation (0-27) where each numbered segment is supposed to correspond across both spectrograms. Despite missing a few stop releases (e.g., the `/p/` in “pairs” in segment 13), we can see that the location of these proposed segment boundaries

[5] Trivially, $\mathrm{DTW}_{u \to u}(t) = t$.

mostly make sense and correspond fairly consistently across both spectrograms. In the region highlighted, however, we can also see a source of inconsistency. This region spans the `/s/` (realized as the voiced fricative, `/z/`) in "pairs" and the `/s/` in "socks" with a short vowel (i.e., a schwa, `/ax/`) in between. This should be three segments; however, our segmentation hypothesizes only two. Furthermore, segment 14 in the male spectrogram contains the schwa, whereas the schwa is attributed to segment 15 in the female spectrogram. Indeed, despite the ability to do quite well on a majority of the speech segments, our developing methods still have to contend with many of these inconsistencies.

Table 6.3 shows the result of an initial experiment using this segmentation method. For consistency, we simply discover our units on just the **pron-eq-train** subset of data. Because we do not have any pronunciation-dependent constraints for the test set, we obtain a segmentation using the parameter marginalization method described previously in Section 6.6.1. Despite the promise of our proposed segmentation approach as seen in Figure 6-3, there is still much work to be done on this front. One reason for such a wide performance discrepancy is that the proposed DTW-constrained segmentation is unable to provide a segmentation at a high enough resolution compared to both the TIMIT segmentation and the parameter-marginalized segmentation method from Section 6.6.1; we saw evidence of these missed segment boundaries in Figure 6-3. The marginalization over utterances seems to act as a low-pass filter of sorts that blurs out finer spectral differences within individual utterances, thus yielding long segments that cannot be faithfully modeled by single acoustic units consisting of 3-state HMMs. Since our acoustic unit discovery sampling method will use only a subset of these segment boundaries, the fact that this method is unable to err towards hypothesizing more segments is a disadvantage.

In light of this, perhaps our DTW-based correspondences can serve instead as constraints for a subsequent consolidation step after a more fine-grained, initial unit discovery, such as an extension to the noisy channel model of [130]. As an alternative to this top-down approach to segmentation, we subsequently discuss a bottom-up approach that involves bootstrapping the discovery and modeling of broad-class acoustic units.

6.6.3 Towards a bootstrapped approach to segmentation

In Section 6.5.3, we saw that clustering performance in both the NMI and PE-UER sense remained fairly stable in the face of varying the number of units discovered. This brings to bear the idea of replacing the landmark segmentation algorithm altogether and leveraging both clustering and unit recognizer training to bootstrap a

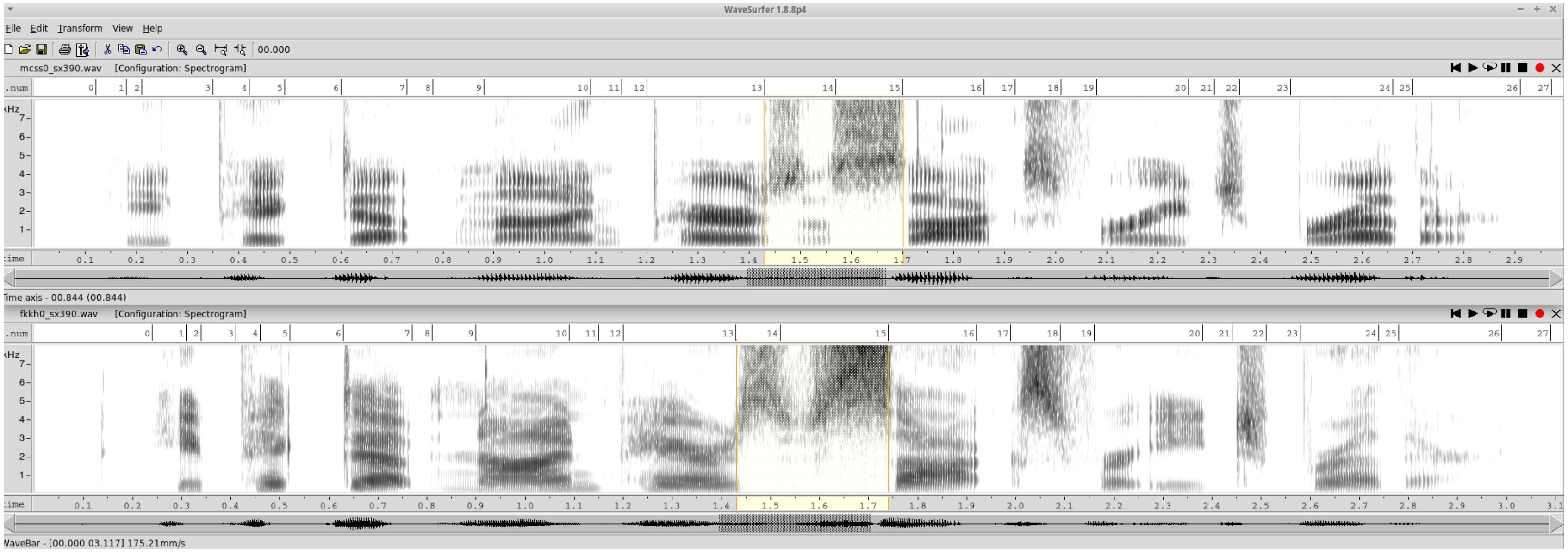


Figure 6-3: *Spectrograms of a male (top) and a female (bottom) speaker saying the TIMIT sx390 sentence, "He picked up nine pairs of socks for each brother." We show the result of our proposed segmentation method that uses a DTW alignment between the two utterances and marginalizes over all landmark detection parameters. While most of the segmentation is fairly reasonable, we highlight one area of inconsistency. This region spans the* **/z/** *in "pairs" and the* **/s/** *in "socks" with a schwa,* **/ax/**, *in between. While this should be three segments; our segmentation hypothesizes only two. Furthermore, segment 14 in the male spectrogram contains the schwa, whereas the schwa is attributed to segment 15 in the female spectrogram.*

Table 6.3: *NMI and PE-UER obtained on the TIMIT test subset using our DTW-constrained segmentation proposed in Section 6.6.2 and constrained clustering discussed in Section 6.5.1. Our results here demonstrate a need for continued improvement in our DTW-constrained segmentation algorithm.*

		NMI	**PE-UER (%)**
0	**TIMIT segmentation**	0.515	25.1
1	**DTW-constrained segmentation**	0.327	36.4

new segmentation. In particular, we considered running acoustic unit discovery on a uniform segmentation – e.g., enforce a new segment every 3 frames – and merging consecutive segments at every sampling iteration. That is, if a cluster sequence had consecutive repeated values, we would remove the boundary between the repeated values and train an HMM on the longer segment. To make such merges more likely, we allowed for only a very small number of acoustic units (e.g., 5-10) in hope that our discovered units would correspond to broad phone classes, such as vowels, fricatives, stops, et cetera. After the unit discovery, we exploit the recognizer training step to further refine these boundary locations and used these final locations as our initial segmentation to learn our final set of (50-100) acoustic units. While results using this method were not unreasonable, they did not yield any improvement over the original landmark segmentation algorithm. Furthermore, with respect to the primary objectives of the explorations in this chapter, this approach provided no straightforward way to exploit text-equivalent constraints. Despite the negative result, we can see this as a step towards a multi-scale approach to segmentation and, potentially, a top-down approach to acoustic unit discovery that can be guided by human input.

6.7 Summary

In this exploratory chapter, we examine various ways to increase the speaker independence of our discovered acoustic units. We propose a simple way to use pronunciation-equivalent constraints by appropriately accumulating statistics during the model update step of our unit discovery process. We also observe the effect of varying the composition of the training data from which we learn these units and find that proper selection of what data to use can have a significant impact on the quality and consistency of our resulting units. In addition to clustering, we further consider a method for constrained segmentation and develop a dynamic programming-based approach to take into account pronunciation-equivalent constraints. We find, however, that our proposed top-down approach may not yield segmentations that are temporally fine enough to be robustly modeled by our acoustic units. As such, despite the promise observed in the visualizations of our segmentation output, there is still much work to be done. Incorporating weak constraints into unit discovery is just the first step in a new saga of work in resource-constrained speech processing.

Chapter 7

Conclusion

In this thesis, we explored the themes of unsupervised domain adaptation and learning from weak supervision in the context of speaker and language recognition. We began in Chapter 3 by motivating and defining a task for domain adaptation in speaker recognition, in which labels are only available for the subset of training data that does not match our evaluation task. Without obtaining additional labels, we drew from our previous work on large-scale speaker clustering and proposed an approach that utilizes both agglomerative and graph clustering techniques to close the gap between our setup and that of a fully supervised system by 85%.

In Chapter 4, we entertained a different set of resource constraints in which we began with no labels of any sort and are tasked with selecting a subset of the training data to be labeled. In this exploration, we develop a greedy algorithm based on active learning that obtains speaker labels in the form of binary pairwise comparisons. We find that the actual number of pairwise comparisons needed to obtain state-of-the-art results is both consistent with the theoretical guarantees in active learning and manageable in practice.

Turning to language recognition in Chapter 5, we employed the use of a large-scale system for acoustic unit discovery to alleviate our reliance on transcribed, multilingual speech. Along the way, we observe the importance of different language-specific perspectives as a form of weak supervision as well as the profound impact of even small bits of linguistic knowledge.

Ultimately, this led us to explore ways to improve our methods for acoustic unit discovery in Chapter 6 so that they can incorporate weak, pronunciation-equivalent constraints and ultimately obtain more speaker-independent acoustic units.

7.1 Future Work

During these investigations, we came across many avenues for future work. Some of these ideas were alluded to towards the end of their corresponding chapters; here, we elaborate upon them at a higher level.

7.1.1 Adapting to broader domains

The domain adaptation task in Chapter 3 involved adapting between two disjoint sets of telephone data. In our exploration, we surmised the cause of this mismatch was driven, in part, by the progress in telephone technology moving from landline to cellular. Another natural source of mismatch is between telephone and microphone recordings, which has been explored in a fully supervised setting [134, 135]. It would be useful to develop methods that can leverage the abundance of telephone data and deploy practical speaker recognition systems built on other recording devices without the need to train on additional labeled data. This moves towards the theme of channel-robustness in general.

7.1.2 Out-of-domain detection

With the problem of domain adaptation comes the problem of detecting a need for it in the first place. In general, this problem can be seen as an instance of the anomaly/outlier detection problem [136, 137], but the ability to do such detection is key to the deployment of more organic spoken language systems, which we motivated at the start of this thesis. For our systems to be aware and able to adapt to ever-changing scenarios, they must start by recognizing when they don't know. Once we have attained the ability to robustly detect an out-of-domain situation, we can then find ways to attribute this mismatch to various factors – in speaker recognition, some possibilities might include utterance length, channel condition, speaker similarity, et cetera. Along these lines of uncertainty attribution, some recent work takes into account utterance length when making speaker recognition decisions [138, 139]; it would be nice to add to that perspective.

7.1.3 Noisy labels and active learning in the real world

Our simulations in active learning for speaker recognition were based on the existence of a noiseless oracle, but previous work has shown that both naive and expert human listeners can be imperfect [59]. A possible extension of our work would involve actually deploying crowd-sourced system for speaker recognition and developing new ways to handle potentially noisy labels. Furthermore, an analysis of the collected data would likely yield new insights into how humans themselves ascertain and evaluate speaker similarity.

7.1.4 Discovering improved feature representations

Our development and use of a large-scale system for acoustic unit discovery yielded insights into the importance of language-specific perspectives as well as the impact of an improved feature representation for speech. In a sense, we verified that "language matters;" that is, any knowledge about the language currently being processed – even information as broad as other closely-related languages – can be helpful and should be exploited. As a direct extension of our explorations in Chapter 5, we should consider using different feature representations to learn acoustic units in various languages. Perhaps this will also help explain the discrepancies in our phonotactic experiments. Automatically uncovering phonotactics is an extremely difficult problem, but our developed ability to process large volumes of data should provide a good place to start.

We also observed the unequivocally positive impact of an improved feature representation for speech, which was obtained using explicit transcriptions. We should continue to extend our work in acoustic unit discovery to automatically learn improved representations for speech by better leveraging speaker, language, and pronunciation-based side information. This may result in features obtained via correspondence autoencoders [128] or a bootstrapped acoustic unit discovery system on acoustic features or even some combination of both. We also have yet to explore the use of more recent developments in recurrent neural network architectures; better features tend to be better at representing longer-term contexts, which may be a task well-suited for LSTMs.

7.1.5 Towards crowd-supervised development of speech technologies

The thesis in [7] brings to bear the potential of crowds of non-expert humans to improve the quality of existing spoken language systems, while the thesis in [97] demonstrates the potential of fully unsupervised methods for discovering linguistic structures in speech. Our work in this thesis attempted to simulate the potential of crowd-based guidance to obtain better acoustic models, and there are many possibilities in which we can enhance our current methods.

We see a future in which speech technology reaches all corners of the world, where robust and strong performance in speech recognition is not limited to the major languages in which we have ample amounts of training data. We imagine the development of systems that can automatically ingest unlabeled recordings from a new language and present non-linguist speakers with queries designed to make better sense of that language's acoustics, semantics, and everything in between. We hope to see a day in which a speech recognizer can be grown from nothing more than the input of a few benign, native speakers.

Our work exploring the use of pronunciation-equivalent constraints were the first steps along those lines. But while we focused on pronunciation equivalence, we have not addressed the notion of pronunciation differences. In particular, knowing that the realizations of two sounds are distinctly different is just as valuable to learning a language's set of phonemes as knowing when two sounds correspond to the same acoustic unit. These bits of information should be relatively straightforward to obtain through the crowd-sourcing of native, but not necessarily expert-level, speakers. And incorporating both positive and negative constraints into our framework would be instrumental.

We should also extend from pronunciaton-equivalent constraints to that of text-equivalent constraints. We limited our initial experiments to pronunciation-equivalent utterances to ensure that our discovered acoustic units do not consider variations in pronunciation as a source of noise. When modeling at a level higher than that of acoustic units, however, incorporating textual constraints would likely be a tremendous source of help in, say, automatically building a lexicon for a particular target language [130].

In this thesis, we explored a variety of methods to make better use of unlabeled data. Yet, by the end of it all, we seem to have come around to looking for convenient ways to obtain *more* labels! As it stands, labeled data is still essential for our development of speech technologies, and while we have discovered many ways in which unlabeled data can help, it seems as though a combination of weakly and semi-supervised approaches will prove to be most useful in the future. In the vastness of this space in between, there are innumerable ways to leverage the powers of both fully supervised and fully unsupervised approaches; we are just getting started.

Bibliography

[1] S. H. Shum, D. A. Reynolds, D. Garcia-Romero, and A. McCree, “Unsupervised Clustering Approaches for Domain Adaptation in Speaker Recognition Systems,” in *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, 2014.

[2] N. Dehak, “Discriminative and Generative Approaches for Long- and Short-Term Speaker Characteristics Modeling: Application to Speaker Verification,” Ph.D. dissertation, Ecole de Technologie Superieure de Montreal, June 2009.

[3] M. Soufifar, “Subspace Modeling of Discrete Features for Language Recognition,” Ph.D. dissertation, Norwegian University of Science and Technology, November 2014.

[4] D. Reynolds, T. Quatieri, and R. Dunn, “Speaker Verification Using Adapted Gaussian Mixture Models,” *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[5] C. Lee and J. Glass, “A Nonparametric Bayesian Approach to Acoustic Model Discovery,” in *Proceedings of ACL*, 2012.

[6] V. Zue, “On Organic Interfaces,” in *Proceedings of Interspeech*, 2007.

[7] I. C. McGraw, “Crowd-supervised Training of Spoken Language Systems,” Ph.D. dissertation, Massachusetts Institute of Technology, June 2012.

[8] R. F. Lyon, “Machine Hearing: An Emerging Field,” *IEEE Signal Processing Magazine*, September 2010.

[9] H. Daume, III and D. Marcu, “Domain Adaptation for Statistical Classifiers,” *Journal of Artificial Intelligence Research*, vol. 26, pp. 101–126, May 2006.

[10] J. Hernandez-Gonzalez, I. Inza, and J. A. Lozano, “Weak supervision and other non-standard classification problems: A taxonomy,” *Pattern Recognition Letters*, vol. 69, pp. 49–55, January 2016.

[11] B. Settles, “Active Learning Literature Survey,” University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[12] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, May 2011.

[13] N. Dehak, P. A. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language Recognition via i-vectors and Dimensionality Reduction," in *Proceedings of Interspeech*, 2011.

[14] N. Dehak and S. Shum, "Low-dimensional Speech Representation Based on Factor Analysis and its Applications," in *Tutorial at Interspeech*, 2011.

[15] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech Communication*, vol. 52, no. 1, pp. 12–40, January 2010.

[16] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[17] S. Shum, "Unsupervised Methods for Speaker Diarization," Master's thesis, Massachusetts Institute of Technology, June 2011.

[18] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of I-vector Length Normalization in Speaker Recognition Systems," in *Proceedings of Interspeech*, 2011.

[19] P. Kenny, "Bayesian Speaker Verification with Heavy-Tailed Priors," in *Proceedings of Odyssey*, 2010.

[20] S. J. Prince and J. H. Elder, "Probabilistic Linear Discriminant Analysis for Inferences About Identity," in *Proceedings of ICCV*, 2007.

[21] N. Brummer and E. de Villiers, "The speaker partitioning problem," in *Proceedings of Odyssey*, 2010.

[22] A. Martin and C. Greenberg, "The 2010 NIST Speaker Recognition Evaluation (SRE10)," 2010, http://www.nist.gov/itl/iad/mig/upload/SRE10_maineval_workshop_public_brief.pdf.

[23] NIST, "The NIST Year 2010 Speaker Recognition Evaluation Plan," 2010, http://www.itl.nist.gov/iad/mig/tests/sre/2010/NIST_SRE10_evalplan.r6.pdf.

[24] ——, "The 2011 NIST Language Recognition Evaluation Plan (LRE11)," 2011, http://www.nist.gov/itl/iad/mig/lre11.cfm.

[25] ——, "The 2015 NIST Language Recognition Evaluation Plan (LRE15)," 2015, http://www.nist.gov/itl/iad/mig/upload/LRE15_EvalPlan_v23.pdf.

[26] A. Hatch, S. Kajarekar, and A. Stolcke, "Within-Class Covariance Normalization for SVM-based Speaker Recognition," in *Proceedings of ICSLP*, 2006.

[27] E. Singer, P. Torres-Carrasquillo, D. Reynolds, A. McCree, F. Richardson, N. Dehak, and D. Sturim, "The MITLL NIST LRE 2011 Language Recognition System," in *Proceedings of Odyssey*, 2012.

[28] N. Brummer, "FoCal Multi-class: Toolkit for Evaluation, Fusion and Calibration of Multi-class Recognition Scores – Tutorial and User Manual," Spescom DataVoice, Tech. Rep., 2007.

[29] D. Garcia-Romero and A. McCree, "Supervised Domain Adaptation for i-vector Based Speaker Recognition," in *Proceedings of ICASSP*, 2014.

[30] NIST, "Speaker Recognition Evaluation 2010," 2010, http://www.nist.gov/itl/iad/mig/sre10.cfm.

[31] C. Vaquero, "Dataset Shift in PLDA-based Speaker Verification," in *Proceedings of Odyssey*, 2012.

[32] L. Ferrer, H. Bratt, L. Burget, H. Cernocky, O. Glembek, M. Graciarena, A. Lawson, Y. Lei, P. Matejka, O. Plchot, and Others, "Promoting robustness for speaker modeling in the community: the PRISM evaluation set," 2012, http://code.google.com/p/prism-set/.

[33] H. Aronowitz, "Adaptation of PLDA to New Domains," in *Results from JHU CLSP Summer Workshop*, 2013.

[34] H. Shimodaira, "Improving Predictive Inference Under Covariate Shift by Weighting the Log-Likelihood Function," *Journal of Statistical Planning and Inference*, vol. 90, no. 2, pp. 227–244, October 2000.

[35] M. Yamada, M. Sugiyama, and T. Matsui, "Covariate Shift Adaptation for Semi-Supervised Speaker Identification," in *Proceedings of ICASSP*, 2009.

[36] M. Sugiyama, M. Krauledat, and K.-R. Muller, "Covariate Shift Adaptation by Importance Weighted Cross Validation," *Journal of Machine Learning Research*, vol. 8, pp. 985–1005, 2007.

[37] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bunau, and M. Kawanabe, "Direct Importance Estimation for Covariate Shift Adaptation," *Annals of the Institute of Statistical Mathematics*, 2008.

[38] D. Garcia-Romero, A. McCree, S. Shum, N. Brummer, and C. Vaquero, "Unsupervised Domain Adaptation for i-vector Speaker Recognition," in *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, 2014.

[39] M. McLaren and D. van Leeuwen, "Source-Normalized LDA for Robust Speaker Recognition Using i-Vectors From Multiple Speech Sources," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 755–766, March 2012.

[40] C. Chelba and A. Acero, "Adaptation of Maximum Entropy Capitalizer: Little Data Can Help A Lot," in *Proceedings of EMNLP*, 2004.

[41] R. K. Ando and T. Zhang, "A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data," *Journal of Machine Learning Research*, 2005.

[42] H. Daume, III, "Frustratingly Easy Domain Adaptation," in *Association of Computational Linguistics*, 2007.

[43] H. Daume, III, A. Kumar, and A. Saha, "Frustratingly Easy Semi-Supervised Domain Adaptation," in *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, 2010.

[44] B. Zadrozny, "Learning and Evaluating Classifiers under Sample Selection Bias," in *Proceedings of ICML*, 2004.

[45] M. Szummer and T. Jaakkola, "Information Regularization with Partially Labeled Data," in *Proceedings of NIPS*, 2002.

[46] S. H. Shum, W. M. Campbell, and D. A. Reynolds, "Large-Scale Community Detection on Speaker Content Graphs," in *Proceedings of ICASSP*, 2013.

[47] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, 2008.

[48] S. van Dongen, "Graph Clustering by Flow Simulation," Ph.D. dissertation, University of Utrecht, May 2000.

[49] D. A. van Leeuwen, "Speaker Linking in Large Data Sets," in *Proceedings of Odyssey*, 2010.

[50] M. Huijbregts and D. van Leeuwen, "Large Scale Speaker Diarization for Long Recordings and Small Collections," *IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

[51] S. E. Tranter and D. A. Reynolds, "An Overview of Automatic Speaker Diarisation Systems," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1557–1565, September 2006.

[52] R. Sokal and C. Michener, "A Statistical Method for Evaluating Systematic Relationships," *University of Kansas Science Bulletin*, 1958.

[53] A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Physical Review E*, 2009.

[54] E. Amigo, J. Gonzalo, J. Artiles, and F. Verdejo, "A Comparison of Extrinsic Clustering Evaluation Metrics based on Formal Constraints," *Information Retrieval*, vol. 12, no. 4, pp. 461–486, 2009.

[55] NIST, "Diarization Error Rate (DER) Scoring Code," 2006, www.nist.gov/speech/tests/rt/2006-spring/code/md-eval-v21.pl.

[56] E. Singer and D. A. Reynolds, "Domain Mismatch Compensation for Speaker Recognition Using a Library of Whiteners," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 2000–2003, July 2015.

[57] F. Richardson, D. Reynolds, and N. Dehak, "Deep Neural Network Approaches to Speaker and Language Recognition," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, 2015.

[58] S. H. Shum, N. Dehak, and J. R. Glass, "Limited Labels for Unlimited Data: Active Learning for Speaker Recognition," in *Proceedings of Interspeech*, 2014.

[59] W. Shen, J. Campbell, D. Straub, and R. Schwartz, "Assessing the Speaker Recognition Performance of Naive Listeners Using Mechanical Turk," in *Proceedings of ICASSP*, 2011.

[60] S. Basu, A. Banerjee, and R. J. Mooney, "Active Semi-Supervision for Pairwise Constrained Clustering," in *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2004.

[61] H. Sun and B. Ma, "Unsupervised NAP Training Data Design for Speaker Recognition," in *Proceedings of Interspeech*, 2012.

[62] ——, "Improved Unsupervised NAP Training Dataset Design for Speaker Recognition," in *Proceedings of Interspeech*, 2013.

[63] S. Cumani, N. Brummer, L. Burget, P. Laface, O. Plchot, and V. Vasilakakis, "Pairwise Discriminative Speaker Verification in the i-vector Space," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 6, pp. 1217–1227, June 2013.

[64] D. Haussler, "Probably Approximately Correct Learning," in *Proceedings of AAAI*, 1990.

[65] S. Dasgupta, "Coarse sample complexity bounds for active learning," in *Proceedings of NIPS*, 2005.

[66] M.-F. Balcan, S. Hanneke, and J. W. Vaughan, "The True Sample Complexity of Active Learning," in *Proceedings of COLT*, 2008.

[67] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice Modeling With Sparse Training Data," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 3, pp. 345–354, May 2005.

[68] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A Study of Inter-Speaker Variability in Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 980–988, July 2008.

[69] D. Cohn, L. Atlas, and R. Ladner, "Improving Generalization with Active Learning," *Machine Learning Journal*, vol. 15, pp. 201–221, 1994.

[70] T. Osugi, D. Kun, and S. Scott, "Balancing Exploration and Exploitation: A New Algorithm for Active Machine Learning," in *Proceedings of IEEE ICDM*, 2005.

[71] S. Shum, N. Dehak, and J. Glass, "On the Use of Spectral and Iterative Methods for Speaker Diarization," in *Proceedings of Interspeech*, 2012.

[72] O. Glembek, L. Burget, N. Dehak, N. Brummer, and P. Kenny, "Comparison of Scoring Methods Used In Speaker Recognition with Joint Factor Analysis," in *Proceedings of ICASSP*, 2009.

[73] Z. Karam and W. M. Campbell, "Graph Embedding for Speaker Recognition," in *Proceedings of Interspeech*, 2010, pp. 2742–2745.

[74] Z. Karam, W. M. Campbell, and N. Dehak, "Graph Relational Features for Speaker Recognition and Mining," in *IEEE Statistical Signal Processing Workshop*, 2011, pp. 525–528.

[75] J. Carbonell and J. Goldstein, "The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries," in *Proceedings of SIGIR*, 1998.

[76] S. H. Shum, D. Harwath, N. Dehak, and J. R. Glass, "On the Use of Acoustic Unit Discovery for Language Recognition," *IEEE Transactions on Audio, Speech, and Language Processing (to appear)*, May 2016.

[77] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Processing Magazine*, pp. 82–97, November 2012.

[78] Y. Song, B. Jiang, Y. Bao, S. Wei, and L.-R. Dai, "i-vector representation based on bottleneck features for language identification," *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, 2013.

[79] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A Novel Scheme for Speaker Recognition Using a Phonetically-Aware Deep Neural Network," in *Proceedings of ICASSP*, 2014.

[80] L. Ferrer, Y. Lei, M. McLaren, and N. Scheffer, "Spoken Language Recognition Based on Senone Posteriors," in *Proceedings of Interspeech*, 2014.

[81] P. Matejka, L. Zhang, T. Ng, S. H. Mallidi, O. Glembek, J. Ma, and B. Zhang, "Neural Network Bottleneck Features for Language Identification," in *Proceedings of Odyssey*, 2014.

[82] F. Richardson, D. Reynolds, and N. Dehak, “A Unified Deep Neural Network for Speaker and Language Recognition,” in *Proceedings of Interspeech*, 2015.

[83] R. Fer, P. Matejka, F. Grezl, O. Plchot, and J. Cernocky, “Multilingual Bottleneck Features for Language Recognition,” in *Proceedings of Interspeech*, 2015.

[84] L. Ferrer, Y. Lei, M. McLaren, and N. Scheffer, “Study of Senone-Based Deep Neural Network Approaches for Spoken Language Recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 1, pp. 105–116, 2016.

[85] J. J. Godfrey, E. C. Holliman, and J. McDaniel, “Switchboard: Telephone Speech Corpus for Research and Development,” in *Proceedings of ICASSP*, 1992.

[86] M.-H. Siu, H. Gish, A. Chan, W. Belfield, and S. Lowe, “Unsupervised training of an HMM-based self-organizing unit recognizer with applications to topic classification and keyword discovery,” *Computer Speech and Language*, vol. 28, no. 1, 2014.

[87] A. Jansen, E. Dupoux, S. Goldwater, M. Johnson, S. Khudanpur, K. Church, N. Feldman, H. Hermansky, F. Metze, R. Rose, M. Seltzer, P. Clark, I. McGraw, B. Varadarajan, E. Bennett, B. Borschinger, J. Chiu, E. Dunbar, A. Fourtassi, D. Harwath, C. Lee, K. Levin, A. Norouzian, V. Peddinti, R. Richardson, T. Schatz, and S. Thomas, “A Summary of the 2012 JHU CLSP Workshop on Zero Resource Speech Technologies and Models of Early Language Acquisition,” in *Proceedings of ICASSP*, 2013.

[88] A. Jansen, K. Church, and H. Hermansky, “Towards Spoken Term Discovery at Scale with Zero Resources,” in *Proceedings of Interspeech*, 2010.

[89] A. Jansen and K. Church, “Towards Unsupervised Training of Speaker Independent Acoustic Models,” in *Proceedings of Interspeech.*

[90] R. Singh, B. Raj, and R. M. Stern, “Automatic Generation of Subword Units for Speech Recognition Systems,” *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 2, pp. 89–99, 2002.

[91] S. Chaudhuri, M. Harvilla, and B. Raj, “Unsupervised Learning of Acoustic Unit Descriptors for Audio Content Representation and Classification,” in *Proceedings of Interspeech*, 2011.

[92] G. Chollet, J. Cernocky, A. Constantinescu, S. Deligne, and F. Bimbot, *Computational Models of Speech Pattern Processing.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, ch. Toward ALISP: A proposal for Automatic Language Independent Speech Processing, pp. 375–388.

[93] D. Petrovska-Delacretaz, M. Abalo, A. E. Hannani, and G. Chollet, "Data-driven Speech Segmentation for Language Identification and Speaker Verification," in *Proceedings of NOLISP*, 2003.

[94] H. Khemiri, "Unified Data-driven Approach for Audio Indexing, Retrieval, and Recognition," Ph.D. dissertation, TELECOM ParisTech, 2013.

[95] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi Speech Recognition Toolkit," in *Proceedings of ASRU*, 2011.

[96] A. Garcia and H. Gish, "Keyword Spotting of Arbitrary Words Using Minimal Speech Resources," in *Proceedings of ICASSP*, 2006.

[97] C. Lee, "Discovering Linguistic Structures in Speech: Models and Applications," Ph.D. dissertation, Massachusetts Institute of Technology, 2014.

[98] J. Glass, "A probabilistic framework for segment-based speech recognition," *Computer Speech and Language*, vol. 17, pp. 137–152, 2003.

[99] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallet, N. L. Dahlgren, and V. Zue, "TIMIT Acoustic-Phonetic Continuous Speech Corpus," 1993.

[100] A. Ihler and D. Newman, "Understanding Errors in Approximate Distributed Latent Dirichlet Allocation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 952–960, May 2012.

[101] M. J. Johnson, J. Saunderson, and A. S. Willsky, "Analyzing Hogwild Parallel Gaussian Gibbs Sampling," in *Advances in Neural Information Processing Systems*, 2013.

[102] J. R. Glass and V. W. Zue, "Multi-Level Acoustic Segmentation of Continuous Speech," in *Proceedings of ICASSP*, 1988.

[103] S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based State Tying for High Accuracy Acoustic Modelling," in *Proceedings of the Workshop on Human Language Technology*, 1994.

[104] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank Matrix Factorization for Deep Neural Network Training with High-dimensional Output Targets," in *Proceedings of ICASSP*, 2013.

[105] Y. Zhang, E. Chuangsuwanich, and J. Glass, "Extracting Deep Neural Network Bottleneck Features Using Low-rank Matrix Factorization," in *Proceedings of ICASSP*, 2014.

[106] P. Torres-Carrasquillo, N. Dehak, E. Godoy, D. Reynolds, F. Richardson, S. Shum, E. Singer, and D. Sturim, "The MITLL NIST LRE 2015 Language Recognition System," in *Proceedings of Odyssey (submitted)*, 2016.

[107] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, and J. Deller, Jr., "Approaches to Language Identification using Gaussian Mixture Models and Shifted Delta Cepstral Features," in *Proceedings of ICSLP*, 2002.

[108] B. Bielefeld, "Language Identification Using Shifted Delta Cepstrum," in *Proceedings of the 14th Annual Speech Research Symposium*, 1994.

[109] D. A. Reynolds, W. M. Campbell, W. Shen, and E. Singer, "Automatic Language Recognition Via Spectral and Token Based Approaches," in *Springer Handbook of Speech Processing and Communication*, 2007.

[110] M. Zissman, "Comparison of Four Approaches to Automatic Language Identification of Telephone Speech," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 1, pp. 31–44, January 1996.

[111] P. Matejka, P. Schwarz, J. Cernocky, and P. Chytil, "Phonotactic Language Recognition using High Quality Phoneme Recognition," in *Proceedings of Interspeech*, 2005.

[112] A. Stolcke, M. Akbacak, L. Ferrer, S. Kajarekar, C. Richey, N. Scheffer, and E. Shriberg, "Improving Language Recognition with Multilingual Phone Recognition and Speaker Adaptation Transforms," in *Proceedings of Odyssey*, 2010.

[113] H. Li, B. Ma, and C. Lee, "A Vector Space Modeling Approach to Spoken Language Identification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 271–284, January 2007.

[114] W. M. Campbell, F. Richardson, and D. A. Reynolds, "Language Recognition with Word Lattices and Support Vector Machines," in *Proceedings of ICASSP*, 2007.

[115] T. Mikolov, O. Plchot, O. Glembek, P. Matejka, L. Burget, and J. Cernocky, "PCA-based Feature Extraction for Phonotactic Language Recognition," in *Proceedings of Odyssey*, 2010.

[116] F. S. Richardson and W. M. Campbell, "NAP for High Level Language Identification," in *Proceedings of ICASSP*, 2011.

[117] M. Soufifar, M. Kockmann, L. Burget, O. Plchot, O. Glembek, and T. Svendsen, "iVector Approach to Phonotactic Language Recognition," in *Proceedings of Interspeech*, 2011.

[118] M. H. Bahari, N. Dehak, H. Van hamme, L. Burget, A. M. Ali, and J. Glass, "Non-Negative Factor Analysis of Gaussian Mixture Model Weight Adaptation for Language and Dialect Recognition," *IEEE/ACM Transactions of Audio, Speech, and Language Processing*, vol. 22, no. 7, pp. 1117–1129, July 2014.

[119] M. Soufifar, L. Burget, O. Plchot, S. Cumani, and J. Cernocky, "Regularized Subspace n-Gram Model for Phonotactic iVector Extraction," in *Proceedings of Interspeech*, 2013.

[120] M. Kockmann, L. Burget, O. Glembek, L. Ferrer, and J. Cernocky, "Prosodic Speaker Verification using Subspace Multinomial Models with Intersession Compensation," in *Proceedings of Interspeech*, 2010.

[121] P. Schwarz, P. Matejka, and J. Cernocky, "Hierarchical Structures of Neural Networks for Phoneme Recognition," in *Proceedings of ICASSP*, 2006.

[122] J. Gauvain, A. Messaoudi, and H. Schwenk, "Language Recognition Using Phone Lattices," in *Proceedings of ICSLP*, 2004.

[123] M. Versteegh, R. Thiolliere, T. Schatz, X. N. Cao, X. Anguera, A. Jansen, and E. Dupoux, "The Zero Resource Speech Challenge 2015," in *Proceedings of Interspeech*, 2015.

[124] L. Badino, C. Canevari, L. Fadiga, and G. Metta, "An Auto-encoder Based Approach to Unsupervised Learning of Subword Units," in *Proceedings of ICASSP*, 2014.

[125] F. Grezl, M. Karafiat, and K. Vesely, "Adaptation of Multilingual Stacked Bottleneck Neural Network Structure for New Language," in *Proceedings of ICASSP*, 2014.

[126] Y. Zhang, E. Chuangsuwanich, and J. Glass, "Language ID-based training of multilingual stacked bottleneck features," in *Proceedings of Interspeech*, 2014.

[127] A. Jansen, S. Thomas, and H. Hermansky, "Weak Top-Down Constraints for Unsupervised Acoustic Model Training," in *Proceedings of ICASSP*, 2013.

[128] H. Kamper, M. Elsner, A. Jansen, and S. Goldwater, "Unsupervised Neural Network Based Feature Extraction Using Weak Top-Down Constraints," in *Proceedings of ICASSP*, 2015.

[129] H. Kamper, W. Wang, and K. Livescu, "Deep Convolutional Acoustic Word Embeddings Using Word-Pair Side Information," in *Proceedings of ICASSP*, 2016.

[130] C. Lee, T. J. O'Donnell, and J. Glass, "Unsupervised Lexicon Discovery from Acoustic Input," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 389–403, July 2015.

[131] M. Johnson, T. L. Griffiths, and S. Goldwater, "Adaptor Grammars: A Framework for Specifying Compositional Nonparametric Bayesian Models," in *Proceedings of NIPS*, 2006.

[132] A. Park and J. R. Glass, "Unsupervised Pattern Discovery in Speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 186–197, 2008.

[133] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, February 1978.

[134] M. Senoussaoui, P. Kenny, N. Dehak, and P. Dumouchel, "An i-vector Extractor Suitable for Speaker Recognition with Both Microphone and Telephone Speech," in *Proceedings of IEEE Odyssey*, 2010.

[135] N. Dehak, Z. N. Karam, D. A. Reynolds, R. Dehak, W. M. Campbell, and J. R. Glass, "A Channel-Blind System for Speaker Verification," in *Proceedings of ICASSP*, 2011.

[136] V. J. Hodge and J. Austin, "A Survey of Outlier Detection Methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[137] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," University of Minnesota, Tech. Rep. TR 07-017.

[138] P. Kenny, T. Stafylakis, P. Ouellet, M. J. Alam, and P. Dumouchel, "PLDA for Speaker Verification with Utterances of Arbitrary Duration," in *Proceedings of ICASSP*, 2013.

[139] S. Cumani, O. Plchot, and P. Laface, "Probabilistic Linear Discriminant Analysis of i-vector Posterior Distributions," in *Proceedings of ICASSP*, 2013.