SEMI-SUPERVISED TRAINING FOR AUTOMATIC SPEECH RECOGNITION

by

Vimal Manohar

A dissertation submitted to The Johns Hopkins University

in conformity with the requirements for the degree of

Doctor of Philosophy

Baltimore, Maryland

October, 2019

Abstract

State-of-the-art automatic speech recognition (ASR) systems use sequence-level objectives like Connectionist Temporal Classification (CTC) and Lattice-free Maximum Mutual Information (LF-MMI) for training neural network-based acoustic models. These methods are known to be most effective with large size datasets with hundreds or thousands of hours of data. It is difficult to obtain large amounts of supervised data other than in a few major languages like English and Mandarin. It is also difficult to obtain supervised data in a myriad of channel and environmental conditions. On the other hand, large amounts of unsupervised audio can be obtained fairly easily. There are enormous amounts of unsupervised data available in broadcast TV, call centers and YouTube for many different languages and in many environment conditions. The goal of this research is to discover how to best leverage the available unsupervised data for training acoustic models for ASR.

In the first part of this thesis, we extend the Maximum Mutual Information (MMI) training to the semi-supervised training scenario. We show that maximizing Negative Conditional Entropy (NCE) over lattices from unsupervised data, along with state-level Minimum Bayes Risk (sMBR) on supervised data, in a multi-task architecture gives word error rate (WER) improvements

without needing any confidence-based filtering.

In the second part of this thesis, we investigate using lattice-based supervision as numerator graph to incorporate uncertainties in unsupervised data in the LF-MMI training framework. We explore various aspects of creating the numerator graph including splitting lattices for minibatch training, applying tolerance to frame-level alignments, pruning beam sizes, word LM scale and inclusion of pronunciation variants. We show that the WER recovery rate (WRR) of our proposed approach is 5-10% absolute better than that of the baseline of using 1-best transcript as supervision, and is stable in the 40-60% range even on large-scale setups and multiple different languages.

Finally, we explore transfer learning for the scenario where we have unsupervised data in a mismatched domain. First, we look at the teacher-student learning approach for cases where parallel data is available in source and target domains. Here, we train a "student" neural network on the target domain to mimic a "teacher" neural network on the source domain data, but using sequence-level posteriors instead of the traditional approach of using frame-level posteriors. We show that the proposed approach is very effective to deal with acoustic domain mismatch in multiple scenarios of unsupervised domain adaptation – clean to noisy speech, 8kHz to 16kHz speech, close-talk microphone to distant microphone. Second, we investigate approaches to mitigate language domain mismatch, and show that a matched language model significantly improves WRR. We finally show that our proposed semi-supervised transfer learning approach works effectively even on large-scale unsupervised datasets with 2000 hours of audio in natural and realistic conditions.

Thesis Committee

Primary Readers

Sanjeev Khudanpur (Primary Advisor)

Daniel Povey (Co-Advisor)

Alternate Readers

Shinji Watanabe

Hynek Hermansky

Najim Dehak

Acknowledgments

First, I would like to thank my parents, who raised me, educated me, and got me to this place in life. My father, a scientist in India, was my childhood inspiration, and it was through his steady guidance that I got interested in science and math. He instilled in me the curiosity for knowledge, and put me in the quest for always learning new things. My mother never held back from giving her everything for me. As a child, I was lucky enough to have the luxury of her being my personal tutor. My parents have been my support throughout my life; I could always rely on their love and affection to get me through the hardest times. And to that, I am eternally grateful.

I would like to thank my advisor, Sanjeev Khudanpur, for giving me the opportunity to do a PhD at Johns Hopkins. I am grateful to Sanjeev for continuously guiding me on my research and academics. He taught me all the qualities of being a good researcher. His insights on a number of subjects and a broadminded view of all ideas were invaluable throughout my PhD.

I consider myself fortunate for having been an advisee of Daniel Povey. Dan might seem like a difficult person from the outside, but to anyone who has worked with him, it is obvious that he is a very pleasant person. Dan was not only a great advisor, but also cared deeply for the wellbeing of myself and

his other students. I am grateful for his constant hands-on mentoring; he was always available just an email away whenever I had any issues. I thank him for his continuous efforts to maintain the CLSP cluster day and night, without which I would certainly have not been able to complete my research. The field of speech recognition would probably have not been the same without Dan and his open-source Kaldi toolkit; he made the technology available to millions across the world. I am honored to have been involved in the Kaldi development and to have added my own contributions to the toolkit.

I thank Shinji Watanabe, Hynek Hermansky and Najim Dehak for being in my commitee and going through my thesis. I am grateful to have worked with them and their students on many collaborative projects, and to have got their insights about various problems in speech processing. In addition, I thank Hynek for managing CLSP, and putting such a wonderful academic environment for the students. I thank Jan "Yenda" Trmal for his mentoring and his difficult work of maintaining entire speech system pipelines, which made it incredibly easy for me to work on different projects. I thank Leiby Paola Garcia for taking care of us students through our difficult times, and being a liaison for all our issues. I also thank her for her valuable comments on my thesis.

I thank my senior labmates, Pegah Ghahremani, Vijayaditya Peddinti, Guoguo Chen, Xiaohui Zhang, Chunxi Liu, Keith Levin and Harish Mallidi, for their advice, especially during the early years of my PhD. I thank my labmates and my peers, Hainan Xu, David Snyder, Matthew Wiewsner, Matthew Maciejewski, Ke Li, Hossein Hadian, Yiming Wang, Hang Lyu, Jinyi Yang,

Ashish Arora and others, for their support, for being a fantastic company, and creating a productive and supportive environment in the lab. I am grateful for the many discussions I had with my labmates and peers about my research and other topics, via which I was able to learn a lot. I additionally thank my other collaborators and colleagues from JHU, the JSALT workshop, and my internships at Analog Devices and Microsoft Research.

I thank Ruth Scully, Debbie Race and the rest of the CLSP and ECE administration for taking care of all my needs right from the beginning of my PhD. I thank Carl Pupa, the COE IT team and Dan once more for ensuring that I had all the computational resources I needed for my research.

I thank my beloved girlfriend, Meghana Madhyastha, for believing in me and sticking with me through everything. I will always be grateful for her love and support. I also thank my brother, my grand parents and my extended family for their constant love and wishes. I thank my close friends, Shaunak Mukherjee, Abhilash Balachandran and Mukund Madhav Goyal, for being amazing company, for the fun times, and for their support through everything. I thank all my friends at JHU and outside, and well-wishers that helped me get through this PhD. I thank the Indian community that helped me settle when I first moved into the US. Last, but not the least, I thank the Baltimore biking community for giving me a wonderful hobby to pursue that helped me cope with the stress of PhD. The group rides allowed me to see an entirely different side of the city, meet a diverse group of people, and learn a lot about Baltimore and the US.

Table of Contents

Abstract ii

Thesis Committee iv

Acknowledgements v

Table of Contents viii

List of Tables xv

List of Figures xvii

1 Introduction 1

1.1 Motivation 1

1.1.1 Semi-supervised training 2

1.1.2 Lattice supervision 3

1.1.3 Domain mismatch 3

1.2 Outline 4

1.2.1 Lattice-based semi-supervised sequence training 4

1.2.2 Lattice-free semi-supervised sequence training 4
1.2.3 Semi-supervised transfer learning 5

2 Background 7

2.1 Automatic Speech Recognition (ASR) 7
2.2 Supervised training 11
2.3 Weighted Finite State Transducers (FSTs) 12
2.4 Neural network acoustic models 13
2.4.1 Frame-level training 14
2.4.2 Sequence-level training 15
2.5 Sequence discriminative training 16
2.6 Lattice-free MMI 18
2.6.1 Lower frame-rate HMM topology 19
2.6.2 Denominator graph 19
2.6.3 Numerator graphs 20
2.6.4 Forward-backward computation 21
2.6.5 Cross-entropy regularization 21

3 Lattice-based semi-supervised sequence training 23

3.1 Introduction 23
3.2 Negative Conditional Entropy 24
3.2.1 Lattice Entropy 25
3.2.2 Deep Neural Network Acoustic Model 27

3.2.3 Multilingual training architecture 28
3.3 Experimental results 28
3.3.1 Experimental setups 29
3.3.2 System descriptions 30
3.3.3 Results and discussion 33

4 Semi-supervised lattice-free training **36**
4.1 Introduction 36
4.2 Proposed method 37
4.2.1 Naïve splitting 38
4.2.2 No splitting 39
4.2.3 Smart splitting 40
4.2.3.1 Adding tolerances 40
4.3 Preliminary experiments on Fisher English 42
4.3.1 Experimental setup 42
4.3.2 System descriptions 44
4.3.3 Results and discussion 45
4.3.3.1 Effect of frame weights 45
4.3.3.2 Effect of supervision type 46
4.3.3.3 Effect of tolerance 46
4.3.3.4 Effect of LM scale and beam 47
4.3.3.5 Effect of alternative phone sequences 48

4.4 Large-scale data experiments on Fisher English 49
4.4.1 Experimental setups 50
4.4.2 System descriptions 51
4.4.2.1 LM for decoding unsupervised data 52
4.4.3 Results and discussion 53
4.4.3.1 Effect of larger supervised dataset 53
4.4.3.2 Effect of larger unsupervised dataset and extra LM data 55
4.5 Experiments on Babel 56
4.5.1 Experimental setups 56
4.5.2 System descriptions 58
4.5.3 Results and discussion 59

5 Semi-supervised transfer learning **60**
5.1 Introduction 60
5.2 Domain mismatch in unsupervised data 64
5.2.1 Acoustic mismatch 65
5.2.2 Acoustic mismatch with parallel data 65
5.2.3 Language domain mismatch 66
5.2.3.1 Language-specific phone LM 66
5.3 Semi-supervised teacher-student learning 67
5.3.1 Relation to semi-supervised training 67
5.3.2 Relation to frame-level T-S learning 68

5.3.3 Related works . . . 68
5.3.4 Synopsis . . . 69
5.4 Sequence-KL objective . . . 70
5.4.1 Denominator term . . . 71
5.4.2 Numerator term . . . 71
5.4.3 Derivative computation . . . 72
5.4.4 LF-MMI and sequence-KL . . . 72
5.5 Teacher-student learning experiments . . . 73
5.5.1 Experimental setups . . . 73
5.5.1.1 Clean to noisy speech . . . 73
5.5.1.2 8kHz to 16kHz speech . . . 74
5.5.1.3 Close-talk to far-field microphone speech . . . 75
5.5.2 System descriptions . . . 77
5.5.2.1 Clean to noisy speech . . . 77
5.5.2.2 8kHz to 16kHz speech . . . 79
5.5.2.3 Close-talk to far-field microphone speech . . . 80
5.5.3 Results and discussion . . . 82
5.5.3.1 Effect of multitask learning . . . 84
5.5.3.2 Effect of LF-MMI and sequence-KL . . . 86
5.5.3.3 Effect of LM used for numerator computation 87
5.5.3.4 Overall impact of T-S learning . . . 89
5.6 Domain mismatch experiments . . . 90

5.6.1 Experimental setups 91
5.6.1.1 AMI-IHM to Tedlium 91
5.6.1.2 Tedlium to How2 challenge corpus 92
5.6.1.3 Fisher English to *Fearless steps* challenge corpus 92
5.6.2 System descriptions 94
5.6.2.1 AMI-IHM to Tedlium 94
5.6.2.2 Tedlium to *How2 challenge* corpus 95
5.6.2.3 Fisher English to *Fearless steps* challenge corpus 97
5.6.3 Results and Discussion 100
5.6.3.1 Effect of language model mismatch 102
5.6.3.2 Effect of domain-specific phone LM and separate output layer 104
5.6.3.3 Effect of rescoring with strong LM 105
5.6.3.4 Effect of multitask training and amount of unsupervised data 106
5.6.3.5 Overall impact of semi-supervised transfer learning 107

6 Conclusion and future work **109**

6.1 Conclusions 109
6.2 Future work 114
6.2.1 Sequence-training objectives beyond MMI 114
6.2.2 Lightly supervised training 115

Bibliography 117

Vita 127

List of Tables

3.1 Babel data 30

3.2 WER (%) results on Fisher English (100 hrs supervised + 250 hrs unsupervised) for DNN acoustic models 35

3.3 WER (%) results on Fisher English (100 hrs supervised + 250 hrs unsupervised) for GMM acoustic models 35

3.4 WER (%) results on Babel 35

4.1 WER(%) results using different frame weights 45

4.2 WER(%) results using various supervision (15 hours supervised + 250 hours unsupervised data. Combined WRR over the two test sets is also shown.) 47

4.3 WER(%) results using supervision from lattice (15 hours supervised + 250 hours unsupervised) for various tolerances 47

4.4 Preliminary WER(%) results with determinized lattices (15 hours supervised + 250 hours unsupervised data). Combined WRR over the two sets is also shown. 48

4.5 WER(%) results on *dev* and *test* sets and the combined WRR(%) using supervision without and with alternative phone sequences for each word sequence (15 hours supervised + 250 hours unsupervised) 49

4.6 Babel data 58

5.1 WER(%) results for unsupervised adaptation from clean to noisy. The objective is $(1-\beta)\mathcal{F}_{\text{MMI}} + \beta\mathcal{F}_{\text{KL}}$. 83

5.2 WER(%) results for 8kHz Fisher to 16kHz AMI-IHM 84

5.3 WER(%) results for adaptation from close-talk to distant microphone 85

5.4 Language models for decoding unsupervised data for adaptation from AMI-IHM to Tedlium corpus. Their perplexities (PPL) on Tedlium *dev* set are reported. 95

5.5 Perplexities of language models on the manually transcribed *Fearless steps dev* set 100

5.6 Domain adaptation results for adaptation from AMI-IHM to Tedlium showing WER(%) on *Tedlium dev* and *Tedlium test* sets 102

5.7 WER(%) results on *How2 dev* set 103

5.8 Semi-supervised transfer learning results on *Fearless steps dev* set 103

List of Figures

2.1 Flowchart of the generative process of speech 8

2.2 HMM topology used in LF-MMI 19

4.1 Phone topology used in LF-MMI 42

4.2 FST to add a tolerance of ± 1 43

4.3 WER(%) results on Fisher English for various supervised dataset sizes and a fixed unsupervised dataset size 54

4.4 WER(%) results for various LMs and unsupervised dataset sizes 57

4.5 Semi-supervised training results on Babel corpora 59

5.1 Network architecture for teacher-student learning 61

5.2 Transfer learning and how it compares with other similar problems 62

5.3 8kHz Fisher ->16kHz AMI-IHM WER(%) results: Unsupervised vs semi-supervised multitask training. The solid lines show results on *AMI eval* and dashed lines on *AMI dev*. 83

5.4 8kHz Fisher ->16kHZ AMI-IHM WER(%) results: Unigram vs 3-gram for sequence-KL. The solid lines show results on *AMI eval* and dashed lines on *AMI dev*. 84

5.5 Domain adaptation WER(%) results for adaptation from AMI-IHM to Tedlium corpus for different amounts of target-domain unsupervised data 102

Chapter 1

Introduction

1.1 Motivation

State-of-the-art ASR use sequence-level objectives like Connectionist Temporal Classification (CTC) [1] and Lattice-free Maximum Mutual Information (LF-MMI) [2] for training neural network-based acoustic models. However, these methods are known to be most effective with large size datasets with 100s of hours of data. The performance is shown to degrade with small datasets [3]. It is difficult to obtain large amounts of supervised data other than in a few major languages like English and Mandarin. It is also difficult to obtain supervised data in a myriad of channel and environmental conditions. On the other hand, large amounts of unsupervised audio can be obtained fairly easily. There are enormous amounts of unsupervised data available in broadcast TV, call centers and YouTube for many different languages and in many environment conditions. The goal of this research is *to discover how to best leverage the available unsupervised data for training acoustic models for ASR*.

1.1.1 Semi-supervised training

Semi-supervised training is the setting where we use both the supervised data and unsupervised data to estimate the parameters of the acoustic model. The training data in this setting consists of the following sets:

1. Supervised data ($\mathcal{D}_{\mathcal{L}}$) : L utterances with acoustic features $\mathbf{O}^{(1)}, \mathbf{O}^{(2)}, \ldots, \mathbf{O}^{(L)}$ and corresponding transcripts - $W^{(1)}, W^{(2)} \ldots W^{(L)}$

2. Unsupervised data ($\mathcal{D}_{\mathcal{U}}$) : U utterances with only acoustic features

$$\mathbf{O}^{(L+1)}, \mathbf{O}^{(L+2)}, \ldots, \mathbf{O}^{(L+U)}$$

Unsupervised audio has been used successfully in various approaches that predate the modern neural network acoustic models [4]. One of the most common approaches to semi-supervised learning of acoustic models is self-training [5]–[7], where a seed system trained with only supervised data is used to decode the unsupervised data and the predicted hypotheses are selected as the training transcripts, usually based on confidence-based filtering schemes. However, sequence discriminative training methods are very sensitive to the accuracy of the transcripts [8]–[10] and they don't work very well with self-training approaches without good confidence-based filtering [8], [11]–[13]. However, we can directly incorporate the uncertainties into *sequence-level training* by using *lattice-based supervision* as described in the next section. This research focuses on *the best ways to turn the available unsupervised data into lattice supervision for training acoustic models using sequence-level objectives*.

1.1.2 Lattice supervision

Lattices are a natural way to incorporate uncertainties in hypotheses when dealing with unsupervised data. Lattice supervision has been used in [14] with conventional sequence objectives like MMI [15] as well as in [16] for semi-supervised training with lattice entropy minimization. We use lattice supervision and *minimize lattice entropy for semi-supervised training* of neural network acoustic models as proposed in [17] and described in Chapter 3. In [18], we proposed *an extension to LF-MMI training in the semi-supervised setting*. We extend this further and *investigate various issues related to using lattice supervision* in Chapter 4.

1.1.3 Domain mismatch

Often times, we want ASR models to be used in a domain that is mismatched with the domain that it was trained for. Transfer learning is applied in such cases, and there is a rich survey of machine learning methods like [19], [20] including methods specific to neural networks [21], and speech processing [22]. Where supervised data is available in the target domain, teacher-student learning, multi-task or weight transfer approaches have been useful [23]–[26]. However, in many cases, the only data available in the target domain is unsupervised, and we would like to use the unsupervised audio to train better acoustic models. In [27], we proposed *a sequence-level teacher-student learning approach* for this, thereby extending the semi-supervised LF-MMI training from Chapter 4 *to transfer learning*. We further explore various approaches to deal with *the issue of domain-mismatched data* in Chapter 5.

1.2 Outline

1.2.1 Lattice-based semi-supervised sequence training

In the first part of this thesis (Chapter 3), we explore using a lattice of word hypotheses as supervision for semi-supervised training of neural network acoustic models. We look at standard sequence discriminative training objectives like Maximum Mutual Information (MMI) and extend it to the scenario of semi-supervised training. In this scenario, the analogous objective is to maximize Negative Conditional Entropy (NCE), which is computed over a lattice. Sequence discriminative training methods like MMI and sequence Minimum Bayes Risk (sMBR) generally don't work for semi-supervised training without extensive confidence-based filtering. However, *we show that joint training using NCE on unsupervised data and sequence Minimum Bayes Risk (sMBR) on supervised data in a multi-task architecture gives word error rate (WER) improvement over self-training using sMBR objective on unsupervised data.* We evaluate the proposed approach on the Fisher English task as well as on multiple Babel languages.

1.2.2 Lattice-free semi-supervised sequence training

While in Chapter 3, we explore lattice-based training in the traditional frame-level cross-entropy neural network framework, in the second part of this thesis (Chapter 4), we explore using a lattice as supervision in the lattice-free MMI [2] training framework. The lattice-free training framework allows a natural way to incorporate the uncertainties in unsupervised data through the numerator

supervision graph. *We explore the various aspects of creating the numerator graph including splitting lattices for minibatch training, applying tolerance on frame-level alignments, pruning beam sizes, word language model scale and inclusion of pronunciation variants.* We then investigate the proposed approach on *large-scale setups by varying the amounts of supervised and unsupervised data.* We finally show that the proposed method works in *many different scenarios without requiring extensive tuning* by evaluating on multiple Babel languages.

1.2.3 Semi-supervised transfer learning

Finally in Chapter 5, we explore the scenario where the unsupervised data is in a different domain than the supervised data. First, in cases where there is parallel data in the source and target domains, we propose to use a teacher-student learning approach. Here, we train a new "student" network in the target domain that mimics the outputs of the "teacher" network. While traditionally done using a frame-level approach, *we propose to have the student network mimic the sequence-level posteriors of teacher network.* Thereby, we extend semi-supervised sequence training to transfer learning scenario. We investigate the proposed approach for *unsupervised adaption in the following domain mismatch scenarios* – clean to noisy speech in same language domain, 8kHz to 16kHz speech in different microphone and language domains, close-talk microphone to distant microphone in different language domains. Second, we look at cases where there is no parallel data in source and target domains, but just domain mismatched unsupervised data. Here, we look at the *effect*

that mismatched language domain has in the quality of lattice supervision and semi-supervised training performance using large-scale unsupervised datasets in natural and realistic conditions.

Chapter 2

Background

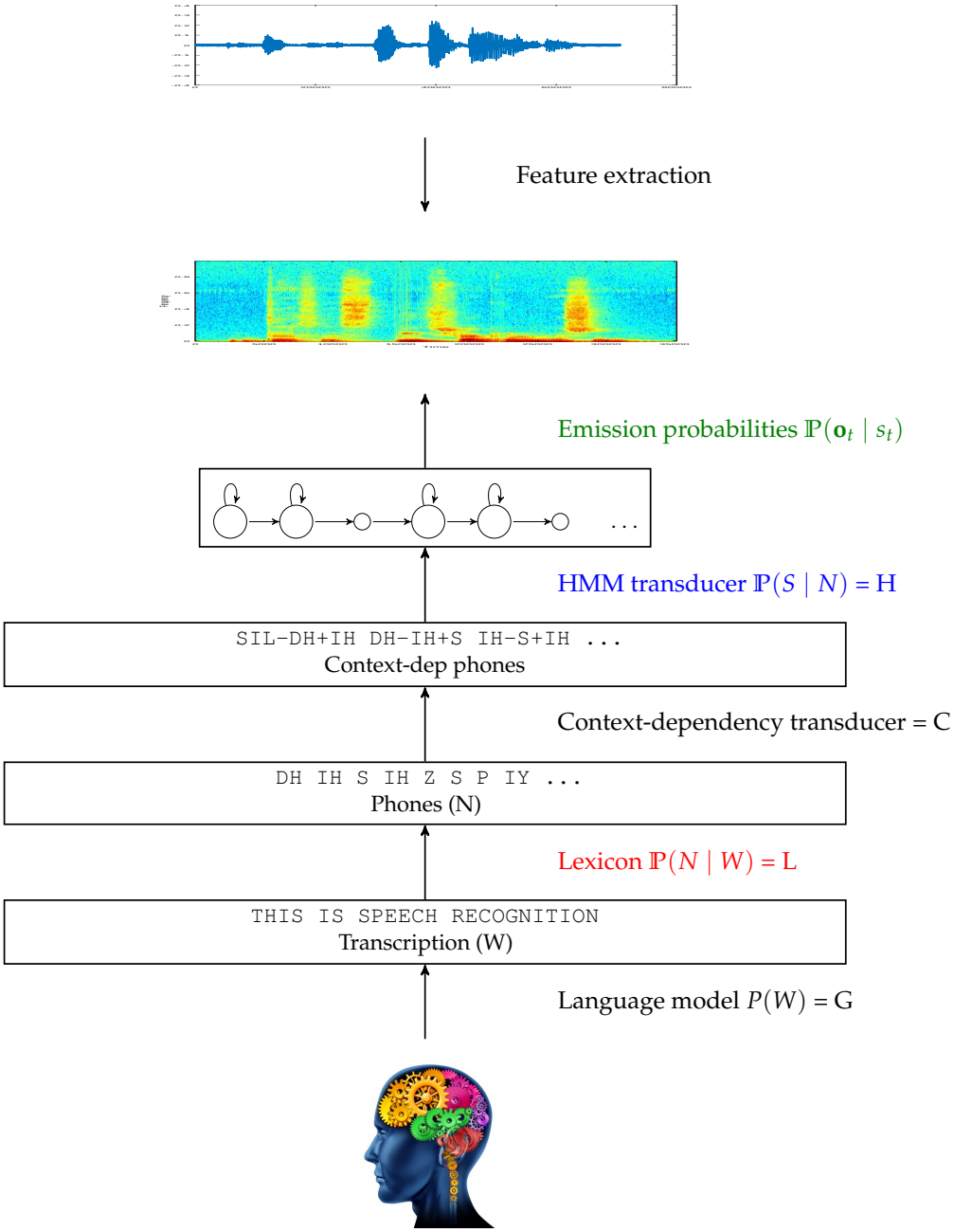
2.1 Automatic Speech Recognition (ASR)

An automatic speech recognition (ASR) system is one that converts an input speech signal into a sequence of words as the text transcription. Mathematically, the problem can be formulated as finding the transcription W that best describes the acoustic feature sequence $\mathbf{O}$ representing the speech signal:

$$\hat{W} = \arg\max_{W} \mathbb{P}(W \mid \mathbf{O}) \tag{2.1}$$

However, it is infeasible to directly model the conditional distribution in (2.1) of the word sequences given the acoustic observations directly. We instead solve this using an inference over a generative process. Using Bayes' rule, some independence assumptions and chain rule with Markov assumptions, we can efficiently do the inference using simpler probabilistic models. We assume the raw speech signal has been converted into a sequence of acoustic features of T time-steps $\mathbf{O} = \mathbf{o}_1\mathbf{o}_2 \cdots \mathbf{o}_T$, where each $\mathbf{o}_t \in R^D$. The generative process (diagrammatically represented in Figure 2.1) is as follows:

Figure 2.1: Flowchart of the generative process of speech



1. A word sequence W is generated using a language model (LM) $\mathbb{P}(W)$. The n-gram LM that we use is represented using a Weighted Finite State Acceptor (WFSA) (G).

2. The word sequence is converted to a sequence of phonemes, which are the simplest linguistic units of sounds using a lexicon model $\mathbb{P}(N \mid W)$. This can be represented using a Weighted Finite State Transducer (L).

3. To account for co-articulation effects of how the phonemes sound in the context of other phonemes, we choose to model context-dependent phonemes. The mapping from phoneme sequence to context-dependent phoneme sequence is done implicitly using a context-dependency transducer (C).

4. The translation of the sequence of context-dependent phonemes into an acoustic feature sequence is modeled by what is known as the acoustic model (AM). Typically this is an Hidden Markov Model (HMM). The mapping from the context-dependency phonemes to the individual transitions in the HMM can be represented using the HMM transducer (H).

5. Each context-dependent phoneme is modeled using its own HMM. The model for the whole sequence is then simply a concatenation the individual HMMs of the context-dependent phones.

6. Each transition through this HMM represents one time-step. In this HMM, we model both the state transition probabilities $\mathbb{P}(s_i \mid s_j)$ and

emission probabilities $\mathbb{P}(\mathbf{o}_t \mid s_i)$, where $s_i, s_j \in \mathcal{S}$ and $\mathcal{S}$ is the state space of the HMM.

7. Because of sparsity issues in modeling all the emission probabilities separately, a decision tree is used to cluster the context-dependent HMM states and all the states in a cluster j are modeled using the same emission probability distribution $\mathbb{P}(\mathbf{o}_t \mid j)$. In ASR parlance, the cluster j is referred to as a "tied state" or by the distribution itself as "p.d.f." (Stands for probability density function) and j is the "p.d.f. label". Typically, these have been estimated using Gaussian Mixture Models (GMMs) and Deep Neural Networks (DNNs).

With this generative process and the probabilistic models within, we can solve the solve the speech recognition problem as:

$$\hat{W} = \arg\max_{W} \mathbb{P}(\mathbf{O} \mid W)\mathbb{P}(W) \tag{2.2}$$

$$\approx \arg\max_{W} \sum_{N} \mathbb{P}(\mathbf{O} \mid N)\mathbb{P}(N \mid W)\mathbb{P}(W) \tag{2.3}$$

$$\approx \arg\max_{W} \sum_{S,N} \mathbb{P}(\mathbf{O} \mid S)\mathbb{P}(S \mid N)\mathbb{P}(N \mid W)\mathbb{P}(W) \tag{2.4}$$

$$\approx \arg\max_{W} \sum_{S,N} \underbrace{\mathbb{P}(\mathbf{O} \mid S)}_{AM} \underbrace{\mathbb{P}(S \mid N)}_{H} \underbrace{\mathbb{P}(N \mid W)}_{L} \underbrace{\mathbb{P}(W)}_{G} \tag{2.5}$$

The advantage of this generative process is that each of the individual models can be trained independently. For training the LM, we need a large text corpus. The lexicon is usually created by an expert. However, there are automatic lexicon learning methods. A simplified approach is to use graphemes

instead of phonemes, which trivializes the lexicon. For training the acoustic model, we typically need a corpus consisting of matched audio and transcription. Both the transition and emission probabilities can be learned jointly using an Expectation-Maximization (EM) algorithm or a gradient descent algorithm without needing an explicit labeling of the hidden variables (HMM states) at each time-step [28].

2.2 Supervised training

The training of acoustic model (parametrized by λ) involves finding a λ that is "good" according to some criteria like Maximum Likelihood (ML), Maximum Mutual Information (MMI) [15] etc. In general, the training data for estimating the AM consists of the following data $\mathcal{D}_{\mathcal{L}}$: L utterances with acoustic feature sequences $\mathbf{O}^{(1)}, \mathbf{O}^{(2)}, \ldots, \mathbf{O}^{(L)}$ and corresponding transcripts $W^{(1)}, W^{(2)} \ldots W^{(L)}$ Since the training data includes transcripts for each utterance that can be used as *supervision* for training the acoustic models, the data is referred to as *supervised data*.

In a supervised training setting, we use only the supervised data to estimate the parameters of the acoustic model λ. The ML criterion for supervised training is as in (2.6). As described in Section 2.1, using a generative process and models for LM, lexicon, context-dependency and HMM, we can expand this into (2.7). [1]

[1] Note that this is very similar to how (2.5) is obtained.

$$\mathcal{F}_{\mathrm{ML}}(\lambda) = \frac{1}{L}\sum_{r=1}^{L}\log\left\{\mathbb{P}(\mathbf{O}^{(r)}, W^{(r)}; \lambda)\right\} \tag{2.6}$$

$$= \frac{1}{L}\sum_{r=1}^{L}\log\left\{\sum_{S,N}\mathbb{P}(\mathbf{O}^{(r)} \mid S; \lambda)\mathbb{P}(S \mid N; \lambda)\mathbb{P}(N \mid W^{(r)})\mathbb{P}(W^{(r)})\right\} \tag{2.7}$$

2.3 Weighted Finite State Transducers (FSTs)

As described in section 2.1, the components in the generative process for speech viz. the language model, the lexicon, the context-dependency and the HMM transition model can be represented using WFSTs – G, L, C and H. The joint probabilities from these models can be obtained by composing the probabilities from the individual models. In terms of FSTs, this is a composition of the H, C, L and G transducers into a composite "HCLG" graph. The probabilities are in general stored in log-domain and are interpreted as "scores" or in a negated version "costs". The scores or costs can in general be relaxed to be non-probabilistic, with the addition of a normalization term.

An acoustic features sequence $\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, \ldots \mathbf{o}_T$ is generated from a path π in this composite HCLG graph with probability $\mathbb{P}(\mathbf{O} \mid \pi)$. Let the sequence of transitions in this path, π, be $a_1, a_2, \ldots a_T$ and $s(a_t)$ be the state corresponding to the transition a_t. The likelihood of the path and acoustic feature sequence

can be computed using the chain rule and HMM assumptions:

$$\mathbb{P}(\mathbf{O}, \pi) = \mathbb{P}(\mathbf{O} \mid \pi)\mathbb{P}(\pi) \tag{2.8}$$

$$= \prod_{t=1}^{T} \mathbb{P}\left(\mathbf{o}_t \mid s(a_t)\right) \mathbb{P}\left(a_t \mid s(a_t)\right), \tag{2.9}$$

where each term consists of two terms:

- an acoustic probability representing the emission of an acoustic feature vector $\mathbf{o}_t$ from state $s(a_t)$, $\mathbb{P}(\mathbf{o}_t \mid s(a_t))$
- a graph probability representing the transition a_t from the state $s(a_t)$, $\mathbb{P}(a_t \mid s(a_t)$.

As noted previously, we use scores more generally and thus we have acoustic scores and graph scores. This can be seen as a transformation of the FST from a probabilistic semiring to a log semiring.

Using this FST framework, we can simplify the expression in (2.7) to (2.10).

$$\mathcal{F}_{\mathrm{ML}}(\lambda) = \frac{1}{L} \sum_{r=1}^{L} \log \left\{ \frac{\sum_{\pi \in \mathcal{G}(W^{(r)})} \mathbb{P}(\mathbf{O}^{(r)} \mid \pi; \lambda)\mathbb{P}(\pi; \lambda)}{\sum_{\pi \in \mathcal{G}(W^{(r)})} \mathbb{P}(\pi; \lambda)} \right\}, \tag{2.10}$$

where $\mathcal{G}(W^{(r)})$ is a graph created specific to the utterance r (using a linear FST created from $W^{(r)}$ in place of a trained language model).

2.4 Neural network acoustic models

Conventionally GMMs were used to model the emission probabilities of the HMM states. GMMs can in theory model any distribution to arbitrary

accuracy given sufficient number of components. In practice, this is infeasible due to the blow up of the number of parameters. Additional assumptions such as diagonal covariance are added to ensure better parameter estimation. This adds a limitation that the acoustic features cannot be augmented with context of nearby frames because the acoustic features in nearby frames are correlated and breaks the diagonal covariance assumption. To overcome these limitations, deep neural networks were proposed for acoustic modeling as summarized in [29]. An HMM with a DNN to model emission probabilities is referred to as HMM-DNN hybrid acoustic model.

2.4.1 Frame-level training

The DNNs proposed for acoustic modeling in [29] were designed to directly predict the HMM tied state or p.d.f. label j given the acoustic feature vector. These were trained to minimize the cross-entropy between the true p.d.f. labels and predicted p.d.f. labels as in (2.11).

$$\mathcal{F}_{\text{CE}}(\lambda) = -\sum_{r=1}^{L}\sum_{t=1}^{T^{(r)}} \log \mathbb{P}(j_t^* \mid \mathbf{o}_t^{(r)}; \lambda), \tag{2.11}$$

where $\left[j_1^*, j_2^* \dots j_{T^{(r)}}^*\right]$ is the p.d.f. label sequence corresponding to the best alignment π^* obtained using a previously trained HMM-GMM acoustic model:

$$\pi^* = \underset{\pi \in \mathcal{G}(W^{(r)})}{\arg\max}\, \mathbb{P}(\pi \mid \mathbf{O}^{(r)}, W^{(r)}; \lambda') \tag{2.12}$$

The HMM state emission probabilities can be obtained from the DNN

using the Bayes' rule:

$$\mathbb{P}(\mathbf{o}_t \mid j_t) = \frac{\mathbb{P}(j_t \mid \mathbf{o}_t)\mathbb{P}(\mathbf{o}_t)}{\mathbb{P}(j_t)}$$

$\mathbb{P}(\mathbf{o}_t)$ is the same for all the HMM states and can be safely ignored during decoding. The resulting term $\mathbb{P}(\mathbf{o}_t \mid j_t) = \frac{\mathbb{P}(j_t|\mathbf{o}_t)}{\mathbb{P}(j_t)}$ is known as a pseudo-likelihood [30]. $\mathbb{P}(j_t)$ is known as the "prior". The prior distribution is conventionally estimated as in [29] using the counts of the p.d.f. labels in all of the training data. The effect of this division by priors is to "remove" the priors implicitly learned by the neural network during training. We found an alternate approach to be useful in many scenarios in [17]. In this approach, we estimate the priors by marginalizing the posterior distribution of the p.d.f. labels using empirical distribution of the acoustic feature vectors in the dataset as shown in (2.13).

$$\begin{aligned}\tilde{\mathbb{P}}(j) =& \sum_{i=1}^{N} \mathbb{P}(j \mid \mathbf{o}_i)\hat{\mathbb{P}}(\mathbf{o}_i) \\ =& \frac{1}{N}\sum_{i=1}^{N} \mathbb{P}(j \mid \mathbf{o}_i)\end{aligned} \tag{2.13}$$

2.4.2 Sequence-level training

An alternative to frame-level training is to train the neural network to predict the whole sequence well. This is similar to the approach of sequence discriminative training that has been used with HMM-GMM acoustic models. This is described in detail in Section 2.5.

Recently, sequence-level training objectives like Connectionist Temporal

Classification (CTC) [1] and Lattice-free Maximum Mutual Information (LF-MMI) [2] have been shown to out-perform traditional frame-level objectives like cross-entropy for sequence tasks like ASR. Neural networks trained using these form state-of-the-art acoustic models in many of the ASR tasks. LF-MMI training is described in greater detail in Section 2.6.

2.5 Sequence discriminative training

Maximum Likelihood training of model is asymptotically optimal (i.e. when there is infinite data for training) and when the model assumptions match the data (e.g. the conditional independence assumptions). However, in practice discriminative training is generally shown to give better estimates than ML training. A discriminative criterion encourages the model to be maximally discriminative of the reference transcript against the competing hypotheses. A number of discriminative criteria such as MMI [15], MCE[31], MPE [32], sMBR [33], [34] and bMMI [35] have been developed and used in HMM-based speech recognition [36]–[39]. Most of these can be generalized and unified into a single framework [40].

The MMI criterion for training acoustic model is as in (2.14). This is a ratio of "numerator" likelihood and "denominator" likelihood. The numerator likelihood is the joint likelihood of the acoustic feature sequence $\mathbf{O}^{(r)}$ and transcription $W^{(r)}$. The denominator term is the likelihood of the acoustic feature sequence alone $\mathbf{O}^{(r)}$. It is computed by marginalizing over all word sequences. In practice, it may have to be approximated for e.g. using a lattice generated by using a weak LM. Composing the probabilities from the

LM, lexicon and the HMM transition model into graph scores, the objective can be expressed as in (2.15). We can see from this that the numerator and denominator likelihoods are computed over numerator and denominator graphs, both of them being WFSTs. The numerator graph is specific to the utterance and is obtained by using a linear FST created from $W^{(r)}$ as the language model. The denominator graph is "free" and uses a pre-trained language model.

$$\mathcal{F}_{\text{MMI}}(\lambda) = \frac{1}{L}\sum_{r=1}^{L}\log\frac{\mathbb{P}(\mathbf{O}^{(r)} \mid W^{(r)};\lambda)\mathbb{P}(W^{(r)})}{\sum_{W'}\mathbb{P}(\mathbf{O}^{(r)} \mid W';\lambda)\mathbb{P}(W')} \tag{2.14}$$

$$= \frac{1}{L}\sum_{r=1}^{L}\log\frac{\sum_{\pi\in\mathcal{G}_{\text{Num}}(W^{(r)})}\mathbb{P}(\mathbf{O}^{(r)} \mid \pi;\lambda)\mathbb{P}(\pi)}{\sum_{\pi'\in\mathcal{G}_{\text{Den}}}\mathbb{P}(\mathbf{O}^{(r)} \mid \pi';\lambda)\mathbb{P}(\pi')} \tag{2.15}$$

In general, it is not feasible to compute the denominator likelihood in (2.14) over all word sequences. The summation is restricted to hypotheses contained in a lattice generated using a weak language model (Usually a unigram or a bigram). But the generation of this lattice (called a denominator lattice) assumes a pre-trained acoustic model. Conventionally, this is an acoustic model trained using an ML objective (in case of HMM-GMM) or a CE objective (in case of HMM-DNN). This adds the limitation that such acoustic models cannot be trained from scratch using sequence-level objectives, but only updated from a pre-trained one. Furthermore, the denominator lattices once generated may not be appropriate once the acoustic model is updated and might need to be regenerated. Without this regeneration of lattices, the acoustic model can get overtrained as seen in [41]. Paths in the numerator

graph that are missing in the denominator graph can also cause problems as seen in [41], [42]. Some of these problems are addressed in approaches like CTC and LF-MMI, which allow the model to be trained from scratch without needing to first generate denominator lattices. LF-MMI is described in detail in the next section.

2.6 Lattice-free MMI

This section summarizes the lattice-free MMI training of neural network acoustic models. For a more detailed presentation, the readers are directed to [2].

Lattice-free MMI (LF-MMI) is a particular implementation of MMI training using the objective (2.15) where the denominator term is computed using a forward-backward over a full HMM graph instead of over a lattice. This graph is fixed and unlike the lattice does not have to be generated for each utterance separately. This makes it possible for the neural network to be trained from scratch without needing to be pre-trained using a frame-level cross-entropy objective. Furthermore, this computation of denominator term is done efficiently using a GPU. Both the denominator graph as well as the utterance-specific numerator graphs are represented as finite state transducers (FSTs). The basic features of LF-MMI system are described in the following sections.

2.6.1 Lower frame-rate HMM topology

LF-MMI system uses a simpler HMM topology than the conventional system. While the conventional system uses a 3-state topology, the LF-MMI system uses a topology that can be traversed in one frame. The frame rate is also 3x lower and thus a single frame corresponds to 30ms. The HMM topology can be represented using an FST as shown in Figure 2.2. A single HMM (corresponding to a context-dependent phone) can emit symbols (p.d.f.s) corresponding to the regular expression ba^* i.e. b, ba, baa etc. Here a is analogous to the blank symbol in CTC, but there is a distinct "blank" symbol for each context-dependent phone.

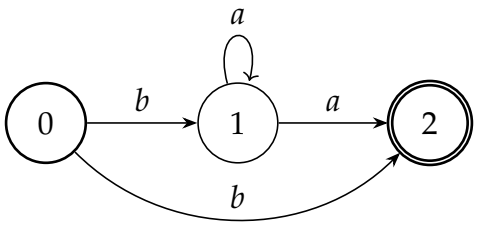


Figure 2.2: HMM topology used in LF-MMI

2.6.2 Denominator graph

The denominator graph is generated using a language model and composing that with the context-dependency (C) and HMM transducer (H). In traditional MMI as described in 2.5, a weak word-level language model like a unigram is used. But in order to make the graph compact and efficiently computable using a GPU, the LF-MMI system uses a 4-gram phone-level language model. There is no pruning or backoff below the trigram level and hence it can

account for only triphones seen in the training data. This LM is represented as a phone graph (G). This is composed on the left with C to add phonetic-context dependency and then with H to add the HMM topology. The composed FST is projected on the input-side to keep only the p.d.f. labels, and followed by epsilon removal and optimized to reduce the size of the graph. The details of this procedure can be found in [2].

During LF-MMI training, the utterances are split into fixed-size chunks of around 1.5s long (or a few different sizes). Due to this, the initial and final probabilities of the graph created above are inappropriate (because they reflect the sentence start and end probabilities). So the initial and final probabilities are modified in the following way. The initial probabilities are set to a new distribution which is obtained by running the HMM for 100 time steps starting from the original initial state and averaging the distribution of the states over those 100 time steps. The final probabilities are all set to one.

2.6.3 Numerator graphs

Unlike the denominator graph, the numerator graph is utterance-specific. Prior to training the neural network, a GMM-based system is used to dump lattice representing alternative pronunciations of the training utterances. These lattices are processed into phone graphs and then compiled into utterance-specific FSTs as for HMM-based conventional training. But, similar to [43], a constraint is added to allow the phones to appear only slightly before or after ($\pm 20ms$ by default) where it appeared in the lattice. This process also allows each state to be identified with a time-index allowing the graph to be split into

chunks for training.

Since the graphs created above don't have any scores on them, the scores are added by composing with a version of denominator FST (known as normalization FST) which includes the initial and final scores as described in Section 2.6.2. This ensures that the objectives are always negative, which helps in debugging. This also ensures that numerator graph does not have any paths not seen in denominator graph, which can cause the training to diverge.

2.6.4 Forward-backward computation

The forward-backward computation to compute the LF-MMI objective and derivates w.r.t. the neural network outputs (pdfs) are implemented on the CPU and GPU for the numerator part and the denominator part respectively. The computation is done in the real-space and not the log-space with special care taken to avoid underflow or overflow. This is detailed in [2]. The LF-MMI training is also regularized by allowing during the denominator computation, transitions from any state to any state with a small probability. This is referred to as the "leaky-hmm". This also allows the forward and backward scores to be kept in a reasonable range during the computation.

2.6.5 Cross-entropy regularization

In [2], it is shown that adding cross-entropy regularization helps the LF-MMI training. A separate output is added to the neural network with just a separate affine component or a separate hidden layer. Thus the cross-entropy part of the neural network has "it's own" separate weight parameters while it

shares most of the parameters with the LF-MMI objective. The supervision for the cross-entropy objective is obtained from the posteriors in the numerator forward-backward computation. The cross-entropy objective is scaled by a small value (usually 0.1) to balance it with the LF-MMI objective.

Chapter 3

Lattice-based semi-supervised sequence training

3.1 Introduction

In this chapter, we investigate sequence discriminative objectives for semi-supervised training of ASR. Discriminative training has been a popular alternative to Maximum Likelihood (ML) training of acoustic models for speech recognition as described in Section 2.5.

Standard discriminative training is very sensitive to the accuracy of the transcripts [8]–[10]. If the reference transcript is incorrect, it discriminates the incorrect reference transcript against the other hypotheses, which includes the true transcript. So sequence discriminative self-training methods do not work very well without some form of confidence-based filtering, as used in [8], [11]–[13]. However, we show that we can directly incorporate the uncertainties by using lattice-based objective functions. In [17], we proposed to use Negative Conditional Entropy (NCE) for semi-supervised training of DNN acoustic models. Entropy minimization has previously been used as

an objective for semi-supervised learning in a facial recognition problem [44] and for sequence-discriminative training of GMM acoustic models [16]. In this chapter, we describe our proposed NCE objective in Section 3.2 and look at experimental results in Section 3.3.

3.2 Negative Conditional Entropy

This section describes the approach we proposed in [17] for using NCE objective for semi-supervised training of DNN acoustic models. Our extension to semi-supervised training can be obtained using the definition of the Maximum Mutual Information Estimation (MMIE) as in (3.1). In the traditional supervised training setting with only supervised data $\mathcal{D}_{\mathcal{L}}$, the expectation in (3.1) is taken over the empirical distribution, $\hat{\mathcal{P}}$. This leads to the standard MMI objective as in (2.14), which is described in Section 2.5.

$$\begin{aligned}\hat{\lambda} &= \arg\max_{\lambda} \mathop{\mathbb{E}}_{(\mathbf{O},W)\sim\hat{\mathcal{P}}} \left[\log \frac{\mathbb{P}(\mathbf{O},W;\lambda)}{\mathbb{P}(\mathbf{O};\lambda)\mathbb{P}(W)}\right] \\ &= \arg\max_{\lambda} \mathop{\mathbb{E}}_{(\mathbf{O},W)\sim\hat{\mathcal{P}}} \left[\log \frac{\mathbb{P}(\mathbf{O} \mid W;\lambda)\mathbb{P}(W)}{\sum_{W'}\mathbb{P}(\mathbf{O} \mid W';\lambda)\mathbb{P}(W')}\right],\end{aligned} \tag{3.1}$$

Similarly for unsupervised data $\mathcal{D}_{\mathcal{U}}$, the expectation is taken over a distribution that is consistent with the empirical distribution over acoustic features. This leads to following criterion:

$$\begin{aligned}\mathcal{F}_{\text{NCE}}(\lambda) &\triangleq \frac{1}{U}\sum_{r=L+1}^{L+U}\sum_{W}\mathbb{P}(W \mid \mathbf{O}^{(r)};\lambda)\log\mathbb{P}(W \mid \mathbf{O}^{(r)};\lambda) \\ &= -\frac{1}{U}\sum_{r=L+1}^{L+U}\mathbb{H}(W \mid \mathbf{O}^{(r)};\lambda),\end{aligned} \tag{3.2}$$

where $\mathbb{H}(W \mid \mathbf{O}; \lambda)$ is the conditional entropy of the word sequence W conditioned on the acoustic feature sequence $\mathbf{O}^{(r)}$ under the posterior distribution with acoustic model parameterized by λ. This criterion was defined as "Negative Conditional Entropy (NCE)" in [16]. Another way of looking at this is that we maximize the weighted average of conditional log-likelihood of the paths in the lattice weighted by their respective probabilities in the lattice.

In the following sections, we describe efficient ways of optimization for the objective (Section 3.2.1), describe how the procedure can be used with DNN acoustic models (Section 3.2.2) and propose a multilingual architecture for DNN training (Section 3.2.3).

3.2.1 Lattice Entropy

It is difficult to directly optimize for the conditional entropy in (3.2). One issue is the summation overall word sequences. In practice, we can approximate that to a summation over word sequences in a lattice $\mathcal{L}_r$. A further approximation, which is exact if the lattices are generated according to [45], is to minimize the entropy of the paths in the lattice $\mathcal{L}_r$:

$$\begin{aligned} \mathcal{F}_{\text{NCE}}(\lambda) &\triangleq \frac{1}{U} \sum_{r=L+1}^{L+U} \sum_{\pi \in \mathcal{L}_r} \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda) \log \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda) \\ &= -\frac{1}{U} \sum_{r=L+1}^{L+U} \mathbb{H}_{\mathcal{L}_r}(\pi \mid \mathbf{O}^{(r)}; \lambda), \end{aligned} \tag{3.3}$$

where π is a HMM state sequence in the lattice as described in .

Lattice-based methods for discriminative training have been developed for many discriminative objective functions including MMI [36], [37]. The lattice

entropy in (3.3) and its gradients can be computed using an Inside-Outside[1] algorithm [46] or equivalently a backpropagation algorithm [47] over the decoded lattice $\mathcal{L}_r$. It is easiest to interpret this algorithm as doing semiring summations over all paths in the lattice under the first-order and second-order expectation semirings respectively [46].

It can be seen that the entropy of the lattice $H_{\mathcal{L}} = \mathbb{H}(\pi \mid \mathbf{O}; \lambda)$ can be computed as follows:

$$\begin{aligned} H_{\mathcal{L}} &= -\sum_{\pi \in \mathcal{L}} \frac{\mathbb{P}(\mathbf{O} \mid \pi; \lambda)\mathbb{P}(\pi)}{Z} \log \left\{ \frac{\mathbb{P}(\mathbf{O} \mid \pi; \lambda)\mathbb{P}(\pi)}{Z} \right\} \\ &= \log Z - \frac{\bar{r}}{Z} \end{aligned} \tag{3.4}$$

where $Z = \sum_{\pi \in \mathcal{L}} \mathbb{P}(\mathbf{O} \mid \pi; \lambda)\mathbb{P}(\pi)$ is the marginal probability and $\bar{r} = \sum_{\pi \in \mathcal{L}} \mathbb{P}(\mathbf{O} \mid \pi; \lambda)\mathbb{P}(\pi) \log [\mathbb{P}(\mathbf{O} \mid \pi; \lambda)\mathbb{P}(\pi)]$. $\langle Z, \bar{r} \rangle$ can be computed as the semiring sum under the first-order expectation semiring with arc weight $\langle p_a, p_a \log p_a \rangle$, where $p_a = \mathbb{P}(o_t \mid s(a); \lambda)\mathbb{P}(a \mid s(a))$ is the probability of the transition and consists of the acoustic and graph probabilities. The gradient $\nabla H_{\mathcal{L}}$ can be computed using the second-order expectation semiring sum $\langle Z, \bar{r}, \nabla Z, \nabla \bar{r} \rangle$ with arc weights $\langle p_a, p_a \log p_a, \nabla p_a, (1 + \log p_a) \nabla p_a \rangle$ as

$$\nabla H_{\mathcal{L}} = \frac{\nabla Z}{Z} - \frac{\nabla \bar{r}}{Z} + \frac{\bar{r} \nabla Z}{Z^2}. \tag{3.5}$$

The derivative of the objective function (3.2) with respect to the probability p_a of an arc a in lattice $\mathcal{L}_r$ simply uses one component in the gradient vector

[1] effectively a Forward-Backward algorithm for a 2-uniform hypergraph like the decoded lattice

(3.5):

$$\frac{\partial \mathcal{F}_{\text{NCE}}}{\partial p_a} = -\frac{1}{U}\frac{\partial H_{\mathcal{L}_r}}{\partial p_a} \tag{3.6}$$

3.2.2 Deep Neural Network Acoustic Model

In a HMM-DNN hybrid system, the DNN is used to provide the emission probability or the pseudo-likelihood [30] of an acoustic feature vector $\mathbf{o}_t$ at time t from a pdf j as described in Section 2.4.

For the neural network acoustic model, the parameters λ are set of all the weights of the connections between the nodes of the neural network. But since the objective function value depends on the neural network weights only through DNN outputs (here the outputs are the log of the pdf posteriors) $\log y_t(j)$, it is enough to find the gradients of the objective function with respect to the DNN outputs.

It should also be noted that in the lattice framework described in Section 3.2.1, each lattice arc a maps to a unique pdf label j.

$$\begin{aligned}\frac{\partial \mathcal{F}_{\text{NCE}}}{\partial \log y_t^{(r)}(j)} &= \sum_{a \in \mathcal{A}_t^{(r)}(j)} \frac{\partial \mathcal{F}_{\text{NCE}}}{\partial \log p_a} \frac{\partial \log p_a}{\partial \log p(\mathbf{o}_t \mid s(a); \lambda)} \\ &= \sum_{a \in \mathcal{A}_t^{(r)}(j)} \frac{\partial \mathcal{F}_{\text{NCE}}}{\partial \log p_a} \qquad (3.7)\end{aligned}$$

$$\implies \frac{\partial \mathcal{F}_{\text{NCE}}}{\partial \log y_t^{(r)}(j)} = \gamma_t^{NCE}(j), \tag{3.8}$$

where $\mathcal{A}_t^{(r)}(j)$ is the set of arcs in the lattice at time t with p.d.f. j and $\gamma_t^{NCE}(j)$

can be termed as the NCE "posterior" of p.d.f. j and is analogous to the MBR "posteriors" defined in [41]. The derivatives w.r.t. to the output can be backpropagated [48] to find the gradients for all the parameters of the neural network. These gradients can be used to optimize the neural network parameters using stochastic gradient descent (SGD).

3.2.3 Multilingual training architecture

In the multilingual training architecture [49], [50], two (or more) DNNs are trained sharing all the layers except the last one. An alternate view is of a single neural network with two or more separate output layers. We can use this idea for semi-supervised training, by viewing the unsupervised data as the "second language". So in this case, the first neural network $nnet_{\mathrm{L}}$ sees only supervised data and the second neural network $nnet_{\mathrm{U}}$ sees only the unsupervised data. The primary motivation is to use the unsupervised data to only learn representations, but use the final layer trained using the supervised data for acoustic classification or decoding. The gradients from the different neural networks can be scaled appropriately to give a higher weight to gradients from the neural network $nnet_{\mathrm{L}}$ that sees supervised data. In addition, the filtering of unsupervised data frames using say, frame-level confidence scores [5], can be incorporated easily.

3.3 Experimental results

We evaluated the performance of our proposed semi-supervised training approach on subsets of Fisher English and on 4 Babel languages. Since in

practice, sMBR objective works better than MMI for ASR tasks, we used sMBR as the objective on supervised data in all our experiments.

3.3.1 Experimental setups

We report experiment results on a subset of Fisher English corpus [51] and several Babel [52] languages in the *limitedLP* condition. The Fisher English corpus has a total of 1600 hours of telephone speech. The first 5000 utterances (about 3.3 hours) in the corpus was selected as the *dev* set for tuning hyperparameters and the next 5000 utterances (about 3.2 hours) was selected as the *test* set for evaluation. Out of remaining data, 100 hours was selected as supervised data and the remaining part was selected as unsupervised data by ignoring the corresponding transcripts. Here, we show results with only 250 hours subset of unsupervised data. We use a language model that is trained using the transcripts of the 100 hours supervised data subset for all experiments.

The Babel languages under *limitedLP* condition have 10 hours of supervised data and around 50-60 hours of unsupervised data (Table 3.1) after automatic segmentation. We show results on four Babel languages – Assamese, Bengali, Zulu and Tamil. We use the fixed lexicon provided under the *limitedLP* condition. We evaluate our systems on the 10 hours *dev10h* set, while tuning on a 2 hours subset *dev2h*. But we don't tune hyperparameters for different languages separately.

For sMBR training, we use a weak language model (unigram) to increase the number of alternative hypotheses for discrimination. But for the NCE training, we use a trigram language model to produce a compact lattice with

Table 3.1: Babel data

Language	$\mathcal{D}_{\mathcal{L}}$ (hrs)	$\mathcal{D}_{\mathcal{U}}$ (hrs)	# Context-dependent states
Assamese	10	50	1984
Bengali	10	57	2005
Zulu	10	59	1913
Tamil	10	65	1902

only the most likely hypotheses. This is in line with the empirical results in [47] that show that a stronger model is better for semi-supervised learning. The decoding of the test sets is also done using the same trigram language model.

3.3.2 System descriptions

All the DNNs used in our experiments have p-norm non-linearity with $p = 2$ and the same basic architecture as in [53]. The features used are the Type IV acoustic features defined in [54]. For the experiments on Fisher English, we use MFCC as the base features. For the Babel experiments, we use PLP as the base features, but we additionally append pitch features [55]. The neural networks are trained using Natural Gradient SGD [56]. The alignments and tree for the DNN training are obtained using a HMM-GMM model trained using only supervised data. The number of context-dependent triphone states is 3915 for Fisher English and is given in Table 4.6 for Babel languages.

We found the prior estimation method by averaging neural network posteriors as described in Section 2.4 to improve performance of the semi-supervised trained neural networks over the traditional method of prior estimation from alignments [30]. We used a subset (3 hours) of supervised

data for the estimation of priors.

Supervised baseline systems

The baseline DNN system *nnet2_CE* is trained with Cross-Entropy as objective with the targets as alignments of $\mathcal{D}_{\mathcal{L}}$ obtained from the HMM-GMM system. The *nnet2_CE* system for Fisher English has 4 hidden layers of p-norm input and output dimensions $(3000, 300)$. The network is mixed-up [53] to 8000 components in the middle of training. The network is trained for 10 epochs using an exponentially decreasing learning rate varying from 0.08 to 0.008. The *nnet2_CE* system for Babel languages have 3 hidden layers of p-norm input and output dimensions $(2000, 200)$. The network is mixed-up to 5000 components. The baseline discriminative system *nnet2_sMBR* is initialized with *nnet2_CE* and trained with sMBR as objective for 4 epochs with a learning rate of 9×10^{-5} and 4 parallel jobs[3].

Self-training systems

For the self-training systems, the unsupervised data was decoded using the *nnet2_CE* system and the best paths through the lattices were chosen as the transcripts. The systems *nnet2_CE_semisup[:conf]* have the same architecture as *nnet2_CE* and are trained from scratch using Cross-Entropy objective using supervised and unsupervised data frames combined together. If $conf$ is specified, then only unsupervised data frames with confidences [5] greater than $conf$ are selected.

The systems *multilang2_CE[:conf]* are DNNs in the multilingual architecture (Section 3.2.3). $nnet_{\mathrm{L}}$ is initialized from a partially trained (after the

mix-up stage) *nnet2_CE* neural network. $nnet_{\mathrm{U}}$ is also initialized from the same *nnet2_CE* neural network, but the final affine layer before softmax is randomized and the mix-up components are removed. The system is trained for 20 epochs as measured on supervised data[2] with learning rate decreasing from 0.04 to 0.004 and 4 parallel jobs[3]. If $conf$ is specified, then frame-confidence-based selection is done as before.

The system *multilang2_sMBR* is the discriminative self-training system in the multilingual architecture. $nnet_{\mathrm{L}}$ and $nnet_{\mathrm{U}}$ are both initialized with the same final *nnet2_CE* model. The system is trained for 4 epochs as measured on the unsupervised data[2] with a learning rate of 9×10^{-5} and 2 parallel jobs[3] for $nnet_{\mathrm{L}}$. The learning rate for the DNN corresponding to unsupervised data was reduced by a factor of 10 in order to give less weight to the gradients from unsupervised data. Using equal learning rate worsened the results.

Proposed system

The system *multinnet2_sMBR+NCE* is similar to *multilang2_sMBR*, but $nnet_{\mathrm{U}}$ trained with NCE. The training is done for 4 epochs[2] with the learning rate in Fisher English setup being 18×10^{-5} for $nnet_{\mathrm{L}}$ with 2 parallel jobs. The learning rate for $nnet_{\mathrm{U}}$ was reduced by a factor of 3. The resulting parameter updates were found to be about 10 times smaller than for $nnet_{\mathrm{L}}$ due to NCE gradients being smaller than sMBR gradients in general. The learning rate in Babel was 9×10^{-5} for $nnet_{\mathrm{L}}$ with 1 DNN job and reduced by a factor of 3 for

[2] It roughly corresponds to the same number of epochs on the both DNNs because the number of parallel jobs are for each DNN is varied in proportion to the amount of data available for the respective DNN

[3] `--num-jobs-nnet` parameter in Kaldi

$nnet_{\mathrm{U}}$.

We also show an oracle system *nnet2_sMBR_oracle* as an upper bound on the performance of the semi-supervised systems. This oracle system is similar to the *nnet2_sMBR* system, but does sMBR training by using true transcripts of the unsupervised data. For comparision with *multinnet2_sMBR+NCE*, the language model for the oracle system is trained using only the transcripts of the supervised data.

3.3.3 Results and discussion

The results on Fisher English with 250 hours of unsupervised data are given in Table 3.2. The self-learning CE system *nnet2_CE_semisup* has a WER worse than the baseline CE system *nnet2_CE* even with frame-filtering. This might be because we did not add multiple copies of supervised data as suggested in [5]. In contrast, self-learning in the multilingual architecture *multilang2_CE* gives nearly 1.4% absolute improvement over supervised CE system *nnet2_CE* with and without frame filtering. This suggests that the multilingual architecture is an effective framework for doing semi-supervised training of DNN.

But the best CE system (*multilang2_CE:0.8*) is still more than 1% worse than the system with supervised discriminative training, *nnet2_sMBR*. This shows that in order to compete with a discriminatively trained system, the semi-supervised learning must involve discriminative training. The discriminatively self-trained system, *multilang2_sMBR*, in the multilingual architecture is shown to be slightly worse than the supervised baseline *nnet2_sMBR* even though the learning rate of $nnet_{\mathrm{U}}$ was reduced by a factor of 10. So we can

conclude that discriminative self-training requires filtering of unsupervised data as was suggested in several works in the literature. But our proposed system *multinnet2_sMBR+NCE* gives 0.16% and 0.38% absolute improvements on *dev* and *test* sets respectively without any explicit filtering of data. Comparing with the oracle system results (*nnet2_sMBR_oracle*) we see that these results of the proposed system (*multinnet2_sMBR+NCE*) correspond to a recovery of 15% and 37% of the possible improvements over *nnet2_sMBR* on *dev* and *test* sets respectively. We believe that the loss in accuracy is due to a combination of inaccuracy in the decoding, mismatch in features because of using unsupervised speaker adaptation for unsupervised data and the choice of MMI as the criterion over sMBR.

We measure the WER recovery rate, which is a metric to measure the performance of semi-supervised trained system. This is defined as:

$$\text{WRR} = \frac{\text{BaselineWER} - \text{SemisupWER}}{\text{BaselineWER} - \text{OracleWER}} \tag{3.9}$$

These results of the proposed system correspond to a WRR of 24% (averaged over the test sets) of the possible improvements if we had oracle transcripts.

Table 3.3 presents analogous results with GMM acoustic models, which demonstrates that the method is not restricted to only DNN acoustic models. Here we use MMI as the supervised training criterion.

We got similar WER improvements from 0.1% absolute on Zulu to 0.6% absolute on Assamese. Improvements in Bengali and Tamil are also in this range as detailed in Table 3.4.

Table 3.2: WER (%) results on Fisher English (100 hrs supervised + 250 hrs unsupervised) for DNN acoustic models

System	*dev*	*test*
nnet2_CE	31.98	31.18
nnet2_sMBR	29.58	28.49
nnet2_CE_semisup	32.40	–
nnet2_CE_semisup:0.8	32.46	–
multilang2_CE	30.61	29.84
multilang2_CE:0.8	30.53	29.81
multilang2_sMBR	29.87	28.77
multinnet2_sMBR+NCE	29.44	28.11
nnet2_sMBR_oracle	28.50	27.46

Table 3.3: WER (%) results on Fisher English (100 hrs supervised + 250 hrs unsupervised) for GMM acoustic models

System	*dev*	*test*
gmm_ML	39.58	38.33
gmm_MMI	38.97	36.88
gmm_ML_semisup	38.67	37.33
gmm_MMI+NCE	38.15	35.84
gmm_MMI_oracle	37.47	35.47

Table 3.4: WER (%) results on Babel

Language	System	*dev2h*	*dev10h*
Assamese	*nnet2_sMBR*	63.9	62.2
Assamese	*multinnet2_sMBR+NCE*	63.4	61.6
Bengali	*nnet2_sMBR*	66.3	64.1
Bengali	*multinnet2_sMBR+NCE*	65.8	63.8
Zulu	*nnet2_sMBR*	65.9	67.3
Zulu	*multinnet2_sMBR+NCE*	65.7	67.2
Tamil	*nnet2_sMBR*	76.3	74.8
Tamil	*multinnet2_sMBR+NCE*	76.1	74.6

Chapter 4

Semi-supervised lattice-free training

4.1 Introduction

Sequence-level objectives like Connectionist Temporal Classification (CTC) [1] and Lattice-free Maximum Mutual Information (LF-MMI) [2] out-perform traditional frame-level objectives like cross-entropy and are the state-of-the-art systems for ASR. However, these methods are known to be data hungry and are more effective with large size datasets with 100s of hours of data. The performance is shown to degrade with smaller datasets [3]. So, extending these methods to unsupervised data is of essence.

As shown in the previous chapter, lattice supervision is a natural way to incorporate uncertainties when dealing with unsupervised data. In this chapter, we use lattice-based supervision to extend standard LF-MMI training to semi-supervised training scenario as proposed in [18]. We describe this method in Section 4.2 and describe the experiments and results in Section 4.3.

4.2 Proposed method

The basic LF-MMI training is described in Section 2.6. For more details, the readers are directed to [2]. In [18], we proposed a modification to the standard LF-MMI training for the semi-supervised scenario. The proposed approach is described in this section.

The standard method of numerator graph creation as described in Section 2.6.3 is applicable only when there is a single transcript (word sequence) for an utterance. We modify this to be applicable to the semi-supervised training scenario by changing the supervision used. In particular, we modify (2.15) to use the numerator graphs created from lattices dumped by decoding unsupervised data using a seed LF-MMI system. In this case, using the same notations as in (2.15) and using $\mathcal{L}^{(r)}$ to correspond to the decoded lattice of utterance r, the training objective can be written as in (4.1). We looked at several approaches to train using this lattice-based supervision. These are described in Sections 4.2.1, 4.2.2 and 4.2.3.

$$\mathcal{F}_{\text{MMI}}(\lambda) = \frac{1}{U}\sum_{r=1}^{U} \log \frac{\sum_{\pi \in \mathcal{G}_{\text{Num}}(\mathcal{L}^{(r)})} \mathbb{P}(\mathbf{O}^{(r)} \mid \pi; \lambda)\mathbb{P}(\pi)}{\sum_{\pi' \in \mathcal{G}_{\text{Den}}} \mathbb{P}(\mathbf{O}^{(r)} \mid \pi'; \lambda)\mathbb{P}(\pi')} \tag{4.1}$$

We then train a new neural network using both supervised data and unsupervised data in a multitask training approach with the two datasets in different minibatches. Our experimental results show that it is better to share all the layers of the network instead of having separate output layers for supervised and unsupervised data. This was the case even when the

unsupervised data was from a mismatched domain.

In the standard method, the denominator graph is generated using a 4-graph phone LM estimated from phone sequences in the training data as described in Section 2.6.2. But in the proposed method, we also get phone sequences from the unsupervised data by taking the best path in the decoded lattice. This ensures that the denominator graph is a good match to the supervised as well as the unsupervised data and can account for all phone sequences in computing the denominator likelihood part of the LF-MMI objective. This is especially important when the supervised data is very small compared to the unsupervised data or the unsupervised data is mismatched with the supervised data.

4.2.1 Naïve splitting

In this approach, we convert a lattice obtained from the decoding of an utterance into phone graphs and then to numerator supervision as in standard LF-MMI as described in Section 2.6.3. But the difference here is in how we get the lattice and the numerator graph. Here, we get the lattice by decoding the unsupervised data using a seed LF-MMI system.

Unlike in standard LF-MMI, these lattices also have word LM scores from the word language model used for decoding the unsupervised data. The standard LF-MMI does not add these scores to the numerator graph, but only adds phone LM scores from the denominator FST. But for semi-supervised training, these word LM scores are added to the numerator graph with an LM scale, $lm\text{-}scale = \alpha$, while the phone LM scores from the denominator FST

are added with a scale of $1 - \alpha$. Thus, we replace the numerator graph path probability as:

$$\mathbb{P}(\pi) = [\mathbb{P}_{\text{word}}(\pi)]^{\alpha} \left[\mathbb{P}_{\text{phone}}(\pi)\right]^{1-\alpha} \tag{4.2}$$

Note that this is a hack as the denominator term in (4.1) uses only the phone LM scores. But in practice, as shown by experimental results, a positive *lm-scale* on the word LM scores help in improving semi-supervised training results. The word LM scores can help upweight "good" paths in the lattice and downweight "bad" paths in the lattice.

The addition of phone LM scores is done by composition with the normalization FST (a version of denominator FST with initial and final scores) after adding time-constraints and splitting into chunks of around 150 frames as described in Section 2.6.3. We call this "naïve" approach because when we split the supervision, we do not add appropriate scores from the lattice to the beginning and end of the splits.

4.2.2 No splitting

In this approach, we do not split the FSTs at the end, but rely on having appropriate chunk sizes of audio before decoding to get lattices. We use the approach in [57] to modify the utterances of the entire unsupervised training corpus to be around 20 distinct lengths. This modification is done using speed perturbation to alter the length of the utterance to the nearest of the distinct lengths. Alternatively, we pad each utterance with silence to get one of the distinct lengths. We do this for the purpose of diagnosis, but this is not the preferred approach as it makes it difficult to use minibatch training or cannot

effectively make use of long history language models for decoding.

4.2.3 Smart splitting

In this approach, we split the lattices directly to get chunks for training (1.5s long i.e. 150 frames) while also adding appropriate initial and final scores to the states at the beginning and end of the chunks. The appropriate initial and final scores are obtained by running a forward-backward [18], [28] on the original unsplit lattice. This ensures that the forward-backward scores for the chunk (after splitting) and the per-frame posteriors are exactly the same as for the original lattice before splitting. (Note that this is the case only at the beginning of the training. Once the model gets updated, the scores will not be the same as that for the unsplit lattice.)

We project this split lattice FST to the input labels to get an FSA with HMM *transition-ids* as labels. We compose this FSA on the right with a special purpose FST to incorporate tolerances. i.e. we allow the phones to occur a few frames behind or ahead of what is defined in the lattice. This is described in the Section 4.2.3.1. The FSA, thus incorporated with tolerance is composed with the normalization FST to add phone LM scores, but with the appropriate scales as in (4.2).

4.2.3.1 Adding tolerances

While we solve the issue of initial and final scores, the smart splitting described above results in "half" phones i.e. for some paths, we might have split in the middle of a phone. It is not possible to convert this split lattice into a

phone graph and compile it into an utterance-specific FST as done with naive splitting. Instead, we directly convert this lattice into an FSA and apply the tolerance. The tolerance application can be formulated as a right composition with a special purpose FST that simulates taking extra self-loops or taking fewer self-loops in the transition model. For a tolerance of ± 1 frame and a transition model from Figure 4.1 with only one phone, this FST is as shown in Figure 4.2. Here b is the *forward transition-id* and a is the *self-loop transition-id*, which is like the blank-symbol in CTC.

Our "tolerance FST" can be trivially extended to multiple phones by making copies of appropriate states and arcs for each phone.

The FST in Figure 4.2 shows three *offset states* for $o \in \{-1, 0, 1\}$. The offset represents the number of *extra* self-loop transition-ids added. $o = +1$ means there is one extra self-loop transition-id, and there must be a deletion down the path in order to accept only valid paths. Offset state 0 is the only final state. This ensures that the input label path and the output label path are of the same length, and the number of frames in the supervision remains the same. We also ensure that there are no duplicate paths created after composition with the lattice:

First, we allow a deletion or insertion of self-loop transition-ids only at the end of the longest ba^* sequence. e.g. the FST can transduce sub-sequence $ba^n a$ into ba^n and $ba^n aa$, but not into $ba^{n-1}a$. In order to remember which forward transition-id has been "seen", for each offset o, we add a set of states of, where f is the forward transition-id. We call these *forward states*. These are the only states with self-loops accepting a sequence of self-loop transition-ids like a^*

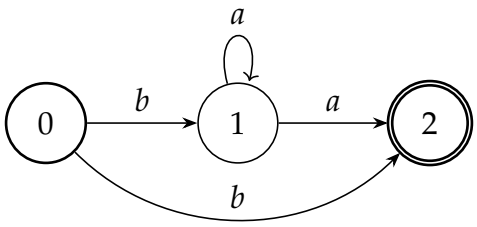
before possibly inserting or deleting a's.

Second, we do not allow paths that mix insertion and deletion. This is done by adding for each state of, a pair of states ofs for $s \in \{i, d\}$ called *insertion states* and *deletion states* corresponding to insertion and deletion respectively.

An insertion arc from a forward state of takes us to an insertion state $(o+1)fi$. From here, we can continue inserting the same transition-id to $(o+2)fi$ and so on or end the process of insertion by taking an ϵ arc to an offset state $o+1$. The deletion process is analogous, but with decreasing offset.

To allow this FST to work with partial phones at the beginning of a split lattice, we add a special *start state* S with an ϵ arc to the offset state 0. We add a self-loop on the start state accepting all the self-loop transition-ids corresponding to transitions of the partial phones. We also add arcs from S to the deletion states corresponding to the offset 0 that delete the self-loop transition-id.

Figure 4.1: Phone topology used in LF-MMI

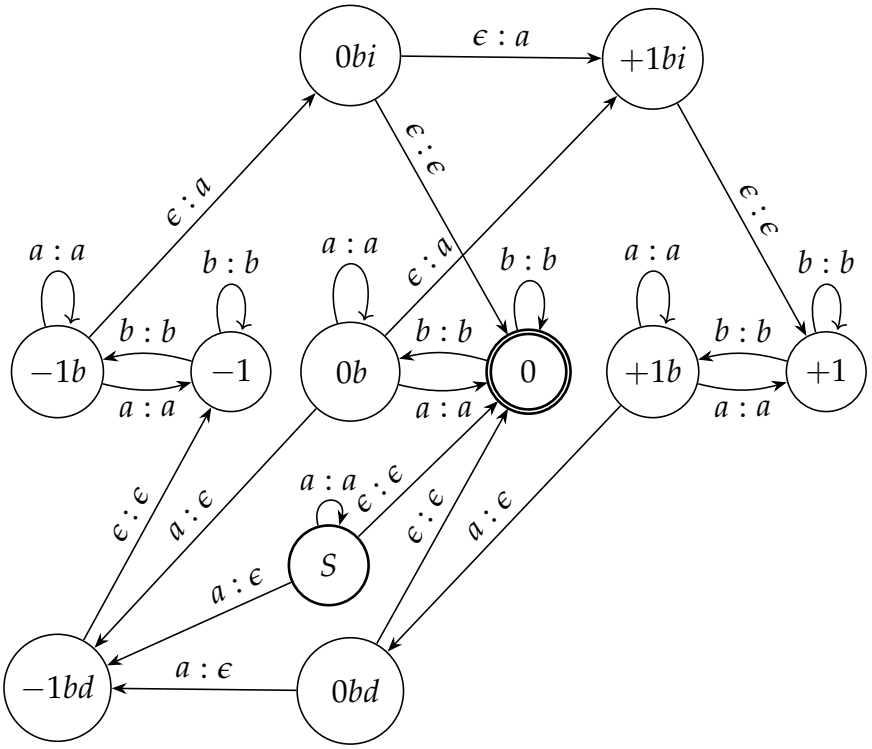


4.3 Preliminary experiments on Fisher English

4.3.1 Experimental setup

In this section, we conduct preliminary experiments using a subset of the Fisher English corpus. We work on small-sized subsets to investigate the effect

Figure 4.2: FST to add a tolerance of ± 1



of various configuration settings in proposed approach. So we use a subset of speakers corresponding to ~250 hours from the corpus as unsupervised data. We used the transcripts from the remaining subset to train the language models for decoding unsupervised data. We used a ~15 hour subset of this as the supervised data to train the acoustic models. The results are reported on separately held-out *dev* and *test* sets (about 3 hours each), which are part of the standard Kaldi [58] recipe for Fisher English[1]. We use WER Recovery

[1] https://github.com/kaldi-asr/kaldi/tree/master/egs/fisher_english/s5

Rate (WRR) [59] as a metric to evaluate the WER improvements from semi-supervised training:

$$\text{WRR} = \frac{\text{BaselineWER} - \text{SemisupWER}}{\text{BaselineWER} - \text{OracleWER}}. \tag{4.3}$$

When evaluating on multiple test sets, the WER is averaged over the test sets and the WRR is computed using the averaged WERs.

4.3.2 System descriptions

Our basic recipe is to first train a GMM system using only the supervised data and use this to get supervision to train a seed LF-MMI time-delay neural network (TDNN) [60] system. The training details are as in [2]. We use i-vector [61] for speaker adaptation of the neural network. To exclude any effect of i-vector extractor, we train the i-vector extractor using the combined supervised and unsupervised datasets. So in these experiments with 15 hours of supervised data, the i-vector extractor was trained using 275 hours including both the supervised and unsupervised datasets. Also, for comparison purposes, we use statistics from only the supervised data to train the context-dependency decision tree for all the systems.

The phone LM used for creating the denominator FST is estimated using phone sequences from both supervised and unsupervised data as in [26], [62]. We give a higher weight of 2.5 to the phone sequences from supervised data to balance out the imbalance (15 hours vs 250 hours) between supervised and unsupervised datasets. We did not tune this weight factor. But it is usually important to have phone sequences from unsupervised data especially if the

amount of supervised data is very small.

4.3.3 Results and discussion

4.3.3.1 Effect of frame weights

We use lattice posteriors of the senones (pdfs) in the best path of the lattice as the frame-weights as done in [5]. The per-frame derivatives of our objective are scaled by these weights. The results are as shown in Table 4.1 for 15 hours supervised and 250 hours unsupervised data (with *lm-scale* 0.5, *beam* 4.0 and tolerance ± 1 frame). Using best path posteriors as frame-weights is significantly better with both naïve and smart splitting of lattices. We also found similar results when using other values of *lm-scale*, *beam* and tolerance, and with 50 hour supervised set. Therefore, all subsequent results are reported using the best path posteriors as frame-weights.

We also experimented using a different confidence measure – per-frame MBR posteriors (obtained from the sausages intermediate to MBR decoding [63]). However, this did not yield any significant gain over using lattice posteriors as confidences. Since lattices posteriors is computationally simpler to compute, we resort to using that instead of MBR posteriors.

Table 4.1: WER(%) results using different frame weights

	Naïve split		Smart split	
Weights	*dev*	*test*	*dev*	*test*
No weights	22.63	22.49	22.14	22.67
Best path posteriors	**22.37**	**22.14**	**22.02**	**21.89**

4.3.3.2 Effect of supervision type

Table 4.2 shows results for various supervision types with 15 hours supervised and 250 hours unsupervised data. The first row shows baseline results with supervised training using only 15 hours data. The last row shows supervised training results using oracle transcripts for the unsupervised data portion. Using only the 1-best phone sequence from the lattice i.e. a *beam* of 0.0 gives 6% absolute improvement over the supervised baseline. In this section, the lattices used are word determinized i.e. for each word sequence in the lattice, we retain a unique HMM state sequence and hence a unique phone sequence. We show later in Section 4.3.3.5 that it is important for the supervision to have all the alternate pronunciation variations. This also includes alternate paths that vary only in the presence or absence of optional silence.

For lattice supervision results shown in Table 4.2, we use a *beam* of 4.0 and *lm-scale* of 0.5. Lattice supervision is clearly better than 1-best phone sequence, and the best results are with smart splitting, although the differences from naïve splitting and no splitting are not very great. We do not recommend no splitting method as we cannot do minibatch training effectively if the lengths of the utterances vary a lot. The smart splitting method is more generalizable.

4.3.3.3 Effect of tolerance

Table 4.3 shows results with various tolerances with *lm-scale* 0.5 and *beam* 4.0. The results with naïve splitting using tolerance ± 1 frame (60ms) and tolerance ± 2 (120ms) frames are both significantly better ($\sim$1% absolute) than using a tolerance of 0. There is no significant difference in WER between using

Table 4.2: WER(%) results using various supervision (15 hours supervised + 250 hours unsupervised data. Combined WRR over the two test sets is also shown.)

Supervision	*lm-scale*	*beam*	tol	*dev*	*test*	WRR(%)
Supervised baseline				29.4	29.2	0
1-best phone seq	0.0	0.0	1	23.0	23.2	54
Naïve split	0.5	4.0	1	22.4	22.1	62
No split + sil	0.5	4.0	1	22.0	22.3	63
No split + speed	0.5	4.0	1	22.1	22.3	62
Smart split	0.5	4.0	1	**22.0**	**21.9**	**64**
Oracle	-	-	-	17.9	18.0	100

tolerance ± 1 and ± 2 and we henceforth use ± 1 frame of tolerance.

Table 4.3: WER(%) results using supervision from lattice (15 hours supervised + 250 hours unsupervised) for various tolerances

			Naïve split		Smart split	
lm-scale	*beam*	tol	*dev*	*test*	*dev*	*test*
0.5	4.0	0	23.48	23.53	23.64	23.85
0.5	4.0	1	**22.37**	**22.14**	**22.02**	**21.89**
0.5	4.0	2	**22.31**	**22.52**	**21.82**	**22.03**

4.3.3.4 Effect of LM scale and beam

Experiments with various LM scales and beams are shown in Table 4.4. As described in Section 4.2.1, *lm-scale* is the interpolation factor between word LM scores from the lattice and phone LM scores from the normalization FST. The lattices used in these experiments are word determinized i.e. it does not include any alternative phone sequences for a word sequence in the lattice. In this case, we see that an *lm-scale* of 0.5 and a *beam* of 4.0 works the best. However, we find that in large-scale setups in Section 4.4 a smaller *beam* of 2.0 is more effective.

Our intuition is that as the beam size is increased, more paths are included

in the lattice and hence the oracle path is more likely included in the lattice. But wider beam also includes more incorrect paths in the lattice, and hence the WER degrades on increasing the beam further.

Table 4.4: Preliminary WER(%) results with determinized lattices (15 hours supervised + 250 hours unsupervised data). Combined WRR over the two sets is also shown.

			Naïve split			Smart split		
lm-scale	*beam*	tol	*dev*	*test*	WRR(%)	*dev*	*test*	WRR(%)
Supervised baseline			29.4	29.2	0.0	-	-	-
0.0	0.0	1	23.0	23.2	54.6	23.0	23.2	54.6
0.0	2.0	1	22.4	22.9	58.6	22.5	22.5	59.9
0.0	4.0	1	22.3	22.3	61.7	22.4	22.6	59.9
0.0	8.0	1	23.5	23.7	50.2	-	-	-
0.5	2.0	1	22.4	22.7	59.5	22.5	22.4	60.4
0.5	4.0	1	**22.4**	**22.1**	**62.1**	**22.0**	**21.9**	**64.8**
0.5	8.0	1	22.4	22.5	60.3	22.1	22.2	63.0
Oracle			17.9	18.0	100.0	-	-	-

4.3.3.5 Effect of alternative phone sequences

Table 4.5 shows results comparing supervision without and with alternative phone sequences. The standard approach of generating lattices for decoding as in [45] creates lattices with only one HMM state sequence and hence one phone sequence for each word sequence. So a supervision created from such a lattice does not have alternative phone sequences. But, as shown in Table 4.5, using supervision with alternative phone sequences for each word sequence is significantly better. The effect is particularly dominant when using only 1-best word sequence as supervision. We believe the inclusion of paths differing in silence is the main reason for the better performance when including alternative phone sequences.

Table 4.5: WER(%) results on *dev* and *test* sets and the combined WRR(%) using supervision without and with alternative phone sequences for each word sequence (15 hours supervised + 250 hours unsupervised)

Supervision	*lm-scale*	*beam*	tol	Without *dev*	Without *test*	Without WRR	With *dev*	With *test*	With WRR
1-best word seq	0.0	8.0	1	23.0	23.2	54.6	22.5	22.3	**60.8**
Naïve split	0.5	4.0	1	22.4	22.1	62.1	21.9	21.7	**66.1**
Smart split	0.5	4.0	1	22.0	21.9	64.8	21.8	21.6	**67.0**

4.4 Large-scale data experiments on Fisher English

In this section, we experiment using different sized subsets of Fisher English corpus as supervised and unsupervised datasets for semi-supervised LF-MMI training. We investigate

1. effect of different sizes of supervised and unsupervised datasets (Sections 4.4.3.1 and 4.4.3.2)
2. different proportions of unsupervised dataset relative to supervised dataset (Sections 4.4.3.1 and 4.4.3.2)
3. amount of language modeling data (Section 4.4.3.2)

The results here are reported on separately held-out *dev* and *test* sets (about 3 hours each) from Fisher English corpus, which are part of the standard Kaldi recipe for Fisher English, as well as standard National Institute of Standards and Technology (NIST) evaluation sets *eval2000* (LDC2002S23[2]) and *rt03* (LDC2007S10[3]).

[2] https://catalog.ldc.upenn.edu/LDC2002S23
[3] https://catalog.ldc.upenn.edu/LDC2007S10

4.4.1 Experimental setups

We consider several different experimental setups varying the amount of supervised data, the amount of unsupervised data and the amount of data for language modeling. The setups are denoted with the notation $supervised\_size + unsupervised\_size$, where the $supervised\_size$ and $unsupervised\_size$ are the sizes (in hours) of subsets used as supervised and unsupervised datasets respectively. For fair comparison, we built the context-dependency tree using only the supervised data and used the same for baseline, semi-supervised trained and oracle systems. For the results in this section, we use TDNN+LSTM acoustic model.

For the first part, we fix the amount of unsupervised data to 250 hours and vary the amount of supervised data (15h, 50h and 100h). In these setups, to estimate the LM used for decoding the unsupervised data, we use the transcripts corresponding to the whole of Fisher English training set but excluding the 250 hours of unsupervised data. In the experimental setup with 100 hours of supervised data, the i-vector extractor is trained using only the supervised data. On the other hand in the setups corresponding to 15 and 50 hours of supervised data, even the 250h of unsupervised data is included for training the i-vector extractor.

For the second part, we fix the amount of supervised data to 100 hours and vary the amount of unsupervised data (250h, 500h, 1000h and 1700h). We investigate two sizes of language modeling data used to estimate the LM for the purpose of decoding unsupervised data in semi-supervised training experiments. The setups where the semi-supervised training uses a small

LM trained on only 100h of supervised data are denoted by suffixing with $+smallLM$, while the setups that use larger LM trained on additional transcripts (from the portion of the corpus excluding the unsupervised set) are denoted without at any suffix. In all these experimental setups, the same i-vector extractor is used – the one trained on 100h of supervised dataset.

4.4.2 System descriptions

For each setup, we consider the following systems:

1. Baseline system trained only on supervised data
2. Semi-supervised trained system using 1-best word as supervision
3. Semi-supervised trained system using lattice supervision
4. Oracle system trained using reference transcripts for all the data including for the portion used as the unsupervised set.

The semi-supervised trained systems here use a hidden layer size that is equal to that used in the oracle system. This is appropriate because both use the same amount of data for training (albeit data is unsupervised in one of them). This is larger than the hidden layer size used for the baseline system, which uses a smaller amount of data for training. Using a larger network for the baseline did not yield any additional improvements.

For the semi-supervised trained system using lattice supervision, we use smart splitting of lattices and include alternative phone sequences in the hypotheses. The supervision here used a *beam* of 2.0 and *lm-scale* of 0.0. This

was the best when using 100 hours of supervised data, although a *beam* of 4.0 and *lm-scale* of 0.5 gave slightly better results when using only 15 or 50 hours of supervised data.

4.4.2.1 LM for decoding unsupervised data

In all the experimental setups, the LM for decoding unsupervised data is a pruned 4-gram modified Kneser-Ney LM trained using the pocolm [64] toolkit. When only supervised data (100h) transcription is available for LM estimation, the constructed LM is small enough to be efficiently compiled into a decoding graph (HCLG). However, when larger amount of LM data is available, we create a smaller pruned LM for decoding to generate lattices and use the larger LM only to rescore the lattices using a memory-efficient lattice rescoring approach [65]. Since the lattice rescoring approach in Kaldi [58] is only directly applicable to determinized lattices (when there is only one path corresponding to each word sequence), a small modification is required to make it applicable to lattices here that have multiple phone sequences for each word sequence.

Given an undeterminized lattice $\mathcal{L}$, and the LM G_{old} used to generate $\mathcal{L}$, the algorithm for rescoring with a new LM G_{new} is as shown in Algorithm 1. In both steps 4 and 5, only costs from the best path in LM are added to the lattice. This is achieved by an intermediate determinization operation to take only the best path in LM. This particular operation only works when the lattice is determinized, and hence the need to determinize the original lattice in step 2. Finally, this determinized lattice projected to word labels to get an

FSA is composed with the original lattice in step 7. This effectively subtracts the costs from the old LM and adds costs from the new LM, thereby rescoring the lattice.

Algorithm 1 Lattice rescoring of undeterminized lattices with new LM G_{new}

1: **function** LATTICE-RESCORE($\mathcal{L}$, G_{old}, G_{new})
2: $\mathcal{L}_{\text{det}} \leftarrow$ Determinize($\mathcal{L}$)
3: Remove the costs from $\mathcal{L}_{\text{det}}$.
4: Add to $\mathcal{L}_{\text{det}}$, the negated costs from the old LM G_{old}
5: Add to $\mathcal{L}_{\text{det}}$, the costs from the new LM G_{new}
6: Project $\mathcal{L}_{\text{det}}$ to output labels (words).
7: $\mathcal{L}_{\text{out}} \leftarrow \mathcal{L} \circ \mathcal{L}_{\text{det}}$
8: **return** $\mathcal{L}_{\text{out}}$
9: **end function**

4.4.3 Results and discussion

4.4.3.1 Effect of larger supervised dataset

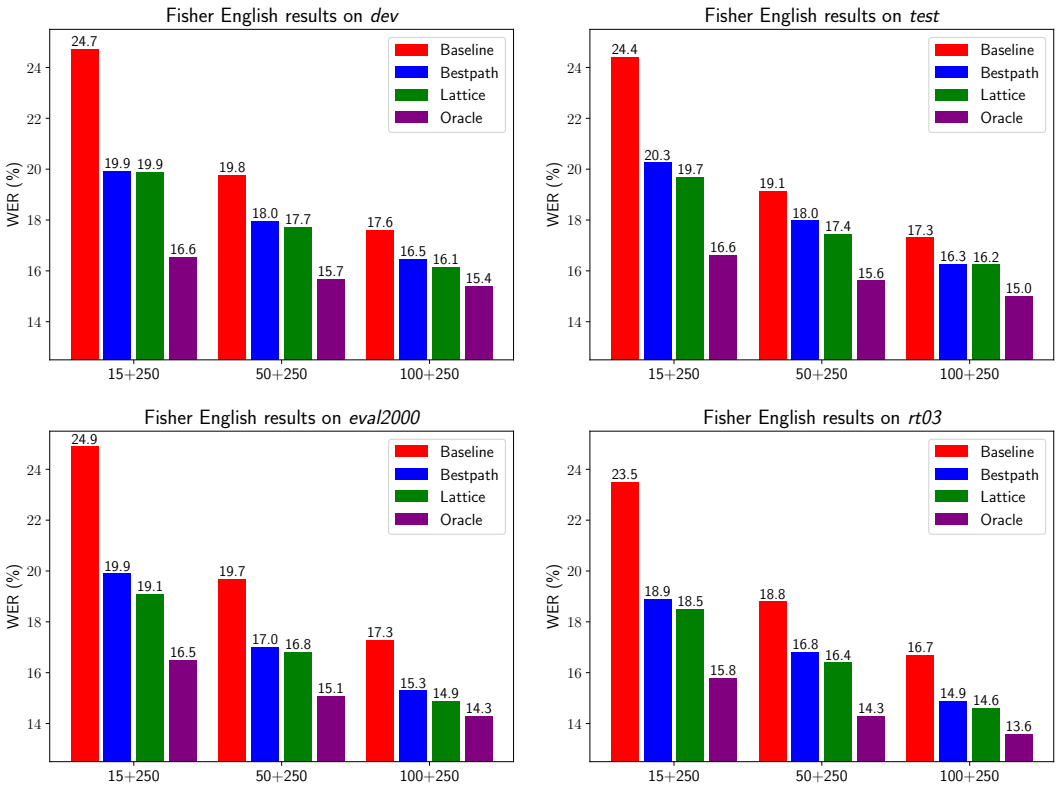
In this section, we report results using fixed amount of unsupervised data (250h) and varying amount of supervised data (15h, 50h and 100h). The experimental setup is described in the previous section (Section 4.4.1).

We can see from the WER(%) results in Figure 4.3 that on all four test sets – *dev*, *test*, *eval2000* and *rt03* – increasing the amount of supervised data (and hence the amount of data available even for semi-supervised and oracle systems) improves the system performance for all the systems – baseline, semi-supervised and oracle. The WER recovery rates (WRR) combined[4] over the four test sets for these TDNN+LSTM acoustic model based systems are in the 50-60% range. But there is no pattern in the variation of WRR with supervised

[4]The WRR is computed using the WER averaged over the four test sets

dataset size. The WRR is seen to decrease moving from 15h to 50h supervised data, but then increases on moving to 100h supervised data. However, these WRRs are in the same range as those seen with TDNN acoustic model based system in the preliminary experiments (Section 4.3). This shows that the proposed semi-supervised training method works well even recurrent neural network-based models like TDNN+LSTM. Further, by comparing the results of the best path (1-best word sequence) supervision and lattice supervision, we see that lattice supervision gives a small consistent gain across all the experimental setups. The WRR when using the lattice supervision is 5-10% absolute higher than when using the best path supervision.

Figure 4.3: WER(%) results on Fisher English for various supervised dataset sizes and a fixed unsupervised dataset size



4.4.3.2 Effect of larger unsupervised dataset and extra LM data

In this section, we report results fixing the supervised dataset to a 100 hours subset of Fisher English and varying the amount of unsupervised data (250h, 500h, 1000h and 1700h). Each setup has two variations – one where the semi-supervised training uses a *small LM* trained on only the supervised data text and the other that uses a *large LM* trained on additional transcripts. For these experiments, we used only a *pruned* version of the *large LM* to generate lattices by building a decoding graph from the pruned version (This pruned version of *large LM* is still larger than *small LM* because it is trained on a much larger amount of text). We did not rescore the lattices using the larger unpruned version. Doing such additional rescoring as described in Section 4.4.2.1 is computationally taxing and did not give any significant improvement in the setups we tried. The experimental setups are described in Section 4.4.1, and are denoted with the notation $\{supervised\_size + unsupervised\_data\_size\}$ with an additional suffix for those using the *small LM*.

The results are in Figure 4.4. We see that using the stronger *large LM* trained on extra transcripts gives a significant improvement over using the *small LM* trained on only the supervised data. This suggests that semi-supervised LF-MMI training is most effecting when using a strong LM for the numerator computation.

By comparing the results of setups using *small LM* for different amounts of unsupervised data in Figure 4.4, we see that as the amount of unsupervised data is increased the WER gets better for both the semi-supervised trained system as well as the oracle system. However, the WERs start saturating and

increasing the amount of data by a large amount shows only a small gain in WER. Because of this, the WER of semi-supervised systems only change by a tiny amount after increasing by a large amount of data. Since the oracle system performance is also saturating, the WER recovery rates (computed using WERs averaged over the four test sets) remain in a consistent range of 40-50%.

Furthermore, for larger setups, we found it more efficient to decrease the number of epochs on unsupervised data, while keeping the number of epochs supervised data the same. We achieve this by making duplicate copies of the supervised data and reduce the overall number of epochs. For e.g., if we make 2 copies of supervised data and decrease the number of epochs from 4 to 2, we are effectively training on supervised data for 4 epochs and on unsupervised data for 2 epochs.

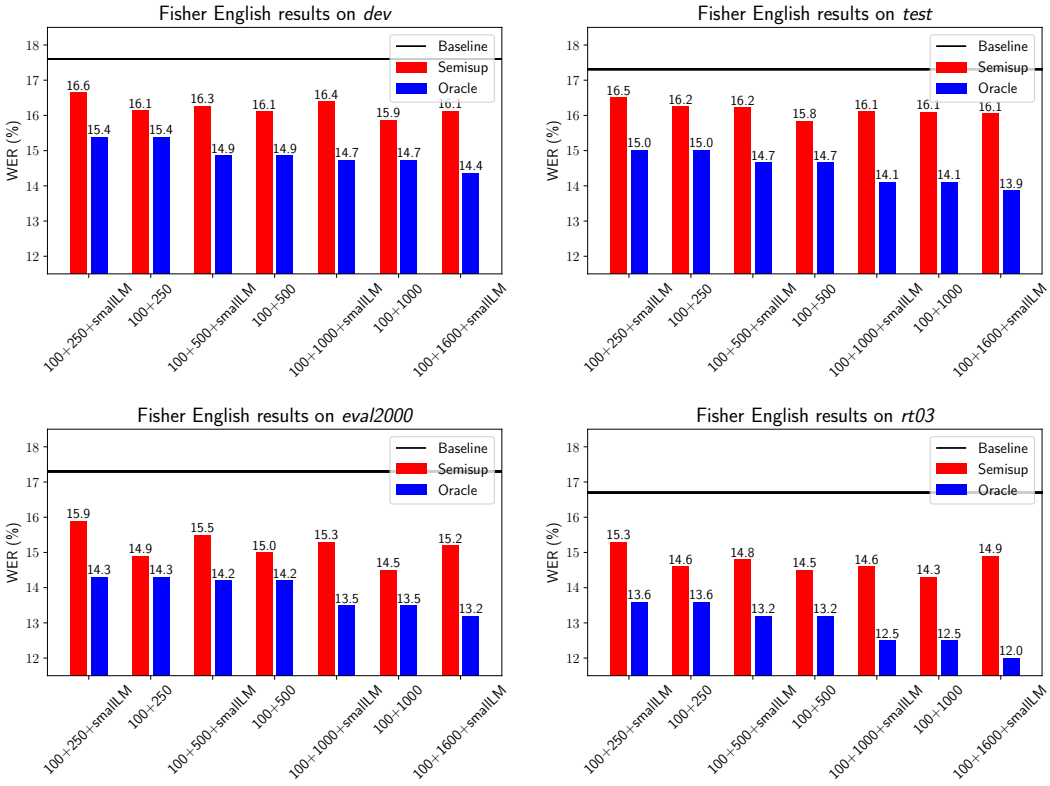
4.5 Experiments on Babel

In this section, we investigate semi-supervised training on many different languages from the Babel corpora.

4.5.1 Experimental setups

The Babel corpora have around 40-80 hours of transcribed speech. 10 hours of data in each language is set as the supervised dataset under the *limitedLP* condition, with the remaining 30-60 hours being the unsupervised dataset. Note that we still use the transcripts for these for the purpose of training the oracle systems, but we do not use them when training the semi-supervised

Figure 4.4: WER(%) results for various LMs and unsupervised dataset sizes



systems.

Table 4.6 shows the amount of data in each Babel language used in the experiments after automatic segmentation. For automatic segmentation, we use a neural network based SAD with a HMM Viterbi decoding for automatically creating segments with a maximum duration of 10s. The neural network SAD has a TDNN architecture with a statistics pooling layer, and was trained on the out-of-domain Fisher data reverberated with synthetic RIRs and additive noise added from MUSAN corpus.

We show results on 5 Babel languages – Assamese, Javanese, Tamil, Turkish and Vietnamese. We evaluate our systems on the 10 hours *dev10h* set.

4.5.2 System descriptions

The supervised systems are trained using lattice-free MMI. For the baseline system, we train only on the 10 hours supervised data. For the oracle system, we train by including the unsupervised portion of the data using the oracle (manual) transcripts.

For semi-supervised training, we use lattice-supervision with smart splitting with a lattice beam of 4.0 and an LM scale of 0.5 each on word LM and phone LM. The lattice supervision was generated by decoding using the baseline system. We fix the tree obtained from the baseline system for the semi-supervised trained system.

For the oracle system we generate numerator lattices using the HMM-GMM system trained only on 10 hours supervised data (same as the system used to generate lattices for the baseline system). However, we build the tree using alignments from all the training data, including the portion that was heldout as unsupervised data when training the baseline and semi-supervised systems. We also investigated fixing the tree from the baseline system and using it for the oracle system, but this did not have any significant impact on the performance.

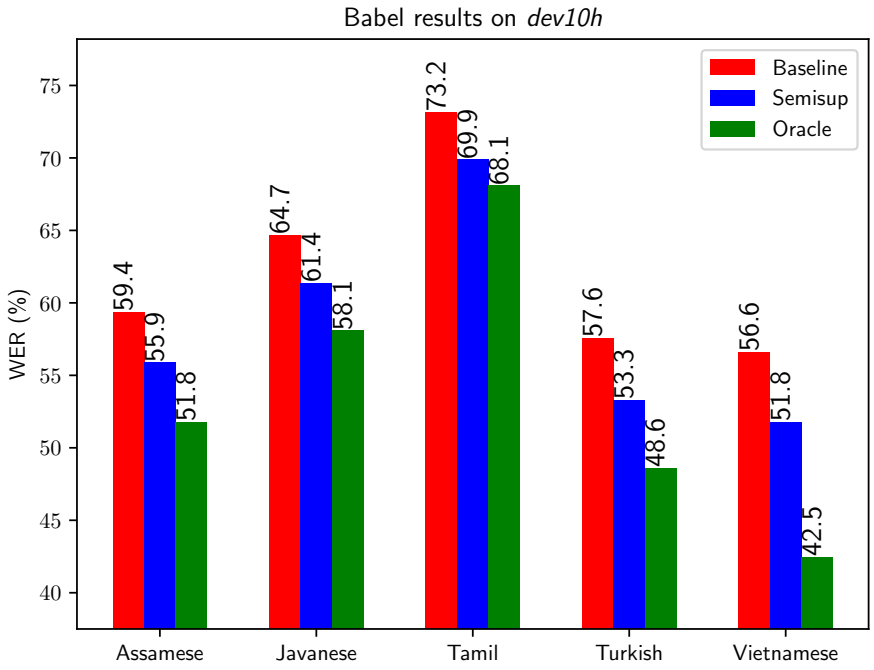
Table 4.6: Babel data

Language	$\mathcal{D}_{\mathcal{L}}$ (hrs)	$\mathcal{D}_{\mathcal{U}}$ (hrs)
Assamese	10	41
Javanese	10	25
Tamil	10	43
Turkish	10	62
Vietnamese	10	62

4.5.3 Results and discussion

The results are as shown in Figure 4.5. We see from the graph that semi-supervised trained system recovers around 50% of the WER improvement from baseline to oracle system in most of the languages; the only exception to this was on the Vietnamese corpus. We suspect that this is because the unsupervised portion of the data in the Vietnamese corpus is significantly different from the supervised data. Further investigation is required to deal with possible vocabulary and lexical differences in order to mitigate the gap in performance.

Figure 4.5: Semi-supervised training results on Babel corpora



Chapter 5

Semi-supervised transfer learning

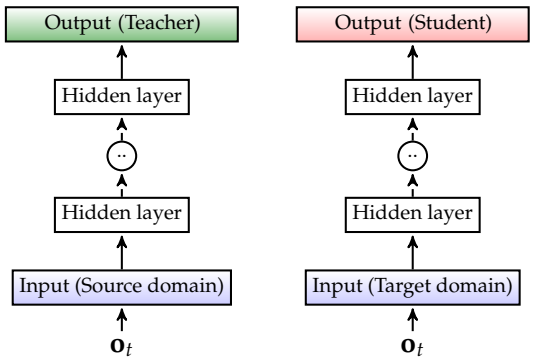
5.1 Introduction

Transfer learning is the general machine learning approach of transferring knowledge from one model to another. Depending on which context it is used, it is called different things. The semi-supervised training in Chapters 3 and 4 is a special case of transfer learning, where we have additional unsupervised data in the same domain. In the case where we have to learn a model in a different domain, the approach is called "domain adaptation". In the case where we have to learn a smaller model on the same domain, the approach is called "model compression". There is a rich survey of transfer learning methods in the literature [19]–[21]. Transfer learning methods have been applied to speech processing in various settings. Wang et. al [22] gives a good overall survey of methods used in speech processing.

One of the methods for transfer learning is the teacher-student (T-S) approach where a teacher network is used to "teach" a student network to make

the same predictions as the teacher. It is traditionally used for model compression [66] as in [67], [68]. It has also been applied in context of domain adaptation [23], where the teacher network is trained on the source domain and the student network is trained on the target domain. It is particularly effective when parallel data is available in source and target domains [69].

Figure 5.1: Network architecture for teacher-student learning

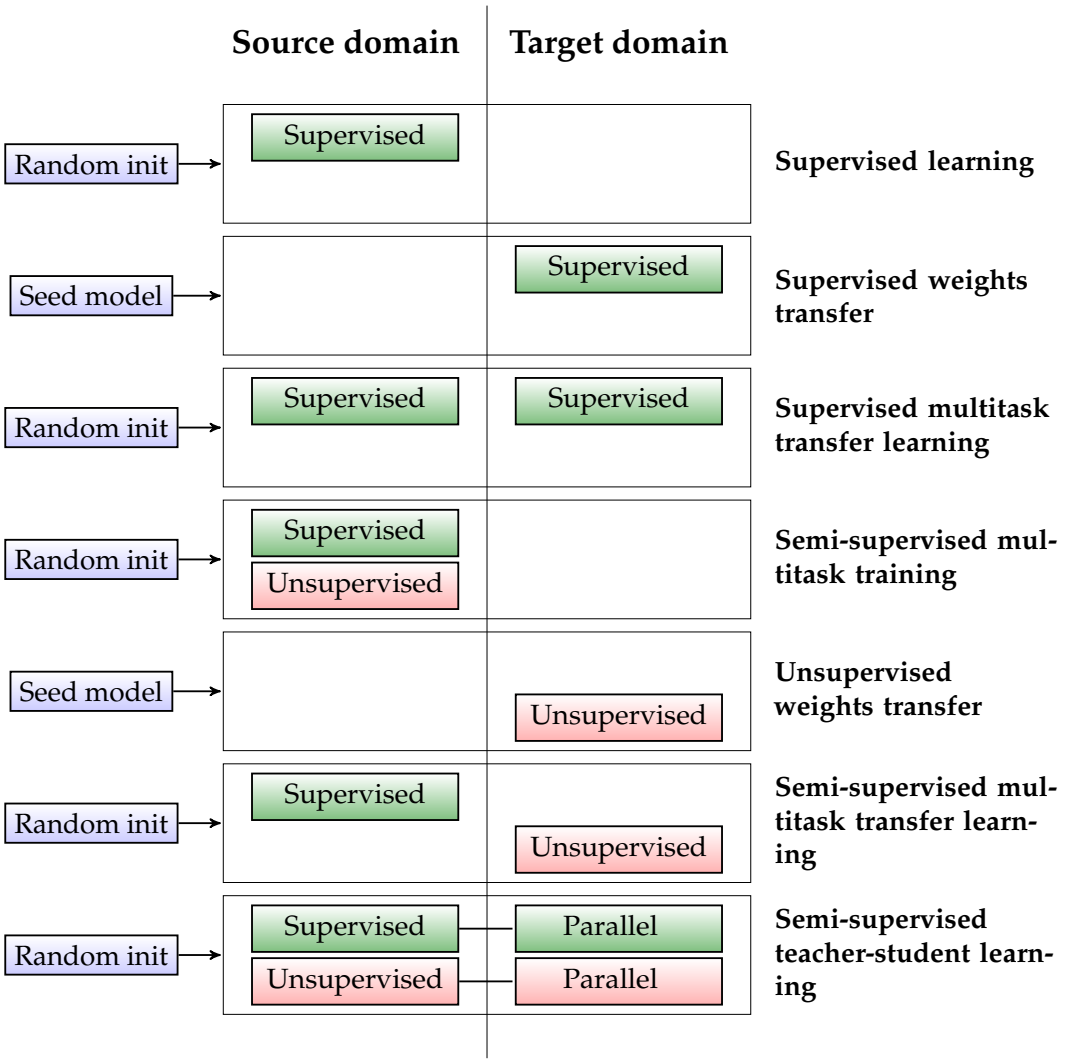


In this chapter, we focus on the problem of unsupervised domain adaptation, where we have a large amount of unsupervised data in the target domain. Figure 5.2 can help to contrast this with other similar problems.

Supervised learning Figure 5.2 shows the traditional case of supervised learning where there is supervised data and the test data are from only a single domain.

Supervised weights transfer Two approaches to supervised transfer learning are shown in the figure. In the first, a seed model trained initially on source domain data is adapted to target domain using a small amount of supervised data in target domain. This is supervised weights transfer

Figure 5.2: Transfer learning and how it compares with other similar problems



approach [26].

Supervised multitask transfer learning The second approach is to jointly train from scratch using supervised data from both source domain and target domain in a multitask training approach [26] and thus creating a model usable on test data from both source and target domains.

Semi-supervised multitask training The scenario in Chapters 3 and 4 is the traditional semi-supervised training, where the unsupervised data is assumed to be from the same domain as the supervised data. Here, the model is trained on both supervised and unsupervised data in a multitask training approach.

Unsupervised weights transfer Unsupervised weights transfer is one case of semi-supervised transfer learning where a seed model is adapted to the target domain, but using only unsupervised data in the target domain.

Semi-supervised multitask transfer learning Another approach to semi-supervised transfer learning is a generalization of the standard semi-supervised training, but where the unsupervised data is from a different target domain. In such a scenario, the network is trained in a multitask approach on supervised data in the source domain and unsupervised data in the target domain. An example of this is to adapt a seed model trained on switchboard corpus to AMI-IHM corpus and test on AMI-IHM. In this case, the model can be trained on supervised switchboard and unsupervised AMI-IHM data in a multitask approach.

Semi-supervised teacher-student learning A special case of this is the semi-supervised teacher-student learning, where parallel data is available in source and target domains. This is described in Section 5.3.

5.2 Domain mismatch in unsupervised data

In the general case of semi-supervised transfer learning, most or all of the data in the target domain of interest is unsupervised. Here, we have a seed acoustic model trained on source domain data that is mismatched with the target domain. We use this to train a new model using the unsupervised target domain data using the semi-supervised sequence-training approaches proposed in Chapter 4.

To summarize, the proposed semi-supervised training approach in Chapter 4 involves using the seed model to decode the unsupervised data to create lattice-based supervision that is used to train a new model in a multitask training approach – using both supervised data and unsupervised data. The supervised data portion uses regular LF-MMI with GMM-generated lattices of alternative pronunciations of training transcripts. The unsupervised data portion uses semi-supervised LF-MMI with lattices obtained by decoding using the seed acoustic model and a language model. This assumes that the unsupervised data is matched with the seed model and the supervised data that is used to train the seed model. But when the target domain unsupervised data is mismatched, we have to add a few additional techniques depending on the kind of acoustic or language domain mismatch. These are described in the following sections. Often times, we need to deal with both acoustic and language domain mismatch. But some of the proposed techniques are orthogonal and complementary.

5.2.1 Acoustic mismatch

If the target data is from a different acoustic condition, multitask learning and weights transfer techniques (Section 5.1) have been shown to be helpful in mitigating the mismatch during LF-MMI training [26]. We apply the same, but using unsupervised data from target domain. In this case, we use lattices generated by decoding the unsupervised data using the seed model to create numerator supervision for LF-MMI training. We note that since the unsupervised data is mismatched, the lattices might not be that good. Nevertheless, experimental results show that with minor mismatch in acoustic conditions, we can still get WER improvements by doing semi-supervised transfer learning. Data augmentation methods like simulating reverberation using synthetic RIRs [70], adding background and foreground noises, speed and volume perturbation [71] to the training data improve the robustness of the seed model, thereby reducing its mismatch with the target domain unsupervised data and making the subsequent semi-supervised training more effective.

5.2.2 Acoustic mismatch with parallel data

In some special cases, there is parallel data available in acoustically mismatched conditions. For e.g., noisy speech can be created by artificially corrupting clean speech, low-resolution audio can be created by downsampling high-resolution audio, speech can be simultaneously recorded using a far-field microphone and a close-talk microphone. In these cases, the acoustic mismatch can be dealt with effectively using a teacher-student (T-S) learning

approach [69]. While the standard T-S learning approach uses a frame-level KL-divergence objective, we propose to use a sequence-level counterpart [24], [25] in lattice-free training framework in Section 5.3.

5.2.3 Language domain mismatch

If the language domain of the unsupervised data is different from that of the supervised data used to train the seed acoustic model, then we simply train a new language model for the target domain. It is usually easier to obtain text data from the target domain of interest than transcribed speech data in the target domain. We then use the mismatched seed acoustic model along with the in-domain language model to decode the target-domain unsupervised data. This ensures that the lattices generated, and hence the numerator supervision for LF-MMI training are better.

We also note that since the language domain is mismatched, the phone sequences used to train phone-level LM for denominator graph creation are also mismatched. But as noted in Section 2.6.2, we interpolate phone n-gram counts from supervised and unsupervised data, and this mitigates the mismatch.

5.2.3.1 Language-specific phone LM

A further extension to deal with this mismatch is to use domain-specific or language-specific phone LMs and use them to create language-specific denominator FSTs. In this case, for the supervised source-domain data, we simply use the phone LM (and hence denominator FST) that was used to train

the seed acoustic model. But for the unsupervised target-domain data, we use a target-specific phone LM that is estimated by including phone n-gram counts obtained from the unsupervised data (by taking the best paths from the decoded lattices).

5.3 Semi-supervised teacher-student learning

Semi-supervised teacher-student learning is a special case of semi-supervised transfer learning where parallel data is available in source and target domains, and a lot of this data is unsupervised. Here, we have a seed network trained on the source domain, which is referred to as the "teacher" network. We use this teacher network and an appropriate LM to decode unsupervised data in the source domain to obtain lattices. These lattices are used as the supervision to train a "student" network, but using the target domain data, which is parallel to the source domain data. The network architecture for this is as shown in Figure 5.1.

5.3.1 Relation to semi-supervised training

One approach to deal with the unsupervised data is to naively apply the semi-supervised training methods of Chapters 3 and 4, while ignoring the mismatch between source and target domains. But since we have parallel data in source and target domains, we can be smarter and decode the unsupervised data in the source domain and use the supervision with parallel target domain data. This is standard practice and is generally shown to be helpful in many scenario. For e.g. in [72], we see that supervision generated using the headset

microphone data improves the performance of acoustic model trained using distant microphone audio. In our case, we use lattices generated by decoding the unsupervised source domain data and use these as supervision to train on the parallel target domain data. For this training, we can use the LF-MMI objective as described in Chapter 4 or a sequence-KL objective, which is proposed in [24], [25]

5.3.2 Relation to frame-level T-S learning

The standard approach for T-S learning uses an objective of minimizing the KL divergence between the frame-level posteriors of the teacher and student networks. However, this is not applicable to LF-MMI trained networks, which do not output frame-level posteriors. Instead, we propose to use KL divergence between sequence-level posteriors [24], [25] from the teacher and student networks as the training objective. Furthermore, we compare this objective to just using LF-MMI objective for training.

5.3.3 Related works

A sequence-KL objective for T-S learning was introduced in [24] for model compression from an ensemble. Unlike that work which used lattice-based discriminative training, here we apply sequence-level KL divergence in the lattice-free training framework for unsupervised domain adaptation. In [25], a lattice-free sequence-KL objective was introduced for model compression and speaker adaptation. Our method here differs in how the supervision for training the student is generated. In particular, we propose a simpler

way to get the supervision using the lattice supervision approach used for semi-supervised LF-MMI training in [18]. We also investigate the effect of using different LMs both when creating the numerator supervision and the denominator graph.

KL divergence objective is also viewed as a regularizer, which prevents the model from diverging too much from what the original model predicts [23]. A sequence-level KL version of this idea was used to regularize LF-MMI based DNN adaptation in [25], [73] to small adaptation sets. On the other hand, our work in this chapter focuses on unsupervised domain adaptation when we have large unsupervised target-domain dataset. We also train our neural networks from scratch since the input features to the student network might be different from that of the teacher network (e.g. 16kHz vs 8kHz). In this context, we can view the sequence-level KL objective to be regularizing semi-supervised LF-MMI training to prevent to the model from over-fitting to the unsupervised data.

5.3.4 Synopsis

In this chapter, we investigate two sequence objectives for teacher-student type transfer learning for unsupervised adaptation – semi-supervised LF-MMI and sequence-level KL divergence (Section 5.4). In Section 5.5, we describe experiments to evaluate our proposed method in the scenario of domain adaptation. We look at three scenario for adaptation – clean to noisy speech, 8kHz to 16kHz audio, and headset microphone to distant microphone.

5.4 Sequence-KL objective

Sequence-KL objective was proposed for T-S learning in [24]. The objective here is to make the student network mimic the teacher network by maximizing the negative KL divergence between sequence-level posteriors from the teacher and student networks as shown in (5.1). We describe in this section our implementation in the lattice-free training framework and how it differs from those in other similar works in [24], [25].

$$\mathcal{F}_{\mathrm{KL}} = -\sum_{r} \sum_{\pi \in \mathcal{G}_{\mathrm{Num}}} \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda^*) \log \left[\frac{\mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda^*)}{\mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda)} \right], \tag{5.1}$$

$$\propto \sum_{r} \Bigg(\sum_{\pi \in \mathcal{G}_{\mathrm{Num}}} \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda^*) \log \mathbb{P}(\mathbf{O}^{(r)} \mid \pi; \lambda) - \log \mathbb{P}(\mathbf{O}^{(r)}; \lambda) \Bigg), \tag{5.2}$$

where $\mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda^*)$ and $\mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda)$ are posterior probabilities of the HMM state sequence π obtained from the teacher network (parameterized by λ^*) and the student network (parameterized by λ) respectively. The former quantity is a constant since the teacher network is fixed when training the student. The simplification[1] to (5.2) makes it clear that the objective consists of numerator and denominator terms.

[1] using Bayes rule and removing the constant additive terms. Also $\sum_{\pi \in \mathcal{G}_{\mathrm{Num}}} \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda^*) = 1$

5.4.1 Denominator term

The denominator term $\log \mathbb{P}(\mathbf{O}^{(r)}; \lambda)$ i.e. the log-likelihood under the student network is independent of the teacher network. In [24], this term was computed using a denominator lattice generated using a unigram LM. However, in lattice-free training, we compute this over a fixed denominator graph, $\mathcal{G}_{\text{Den}}$, just as in the case of LF-MMI. The reader is directed to [2] for details of this forward-backward [28] computation on a GPU. As in [2], the denominator graph is created using a 4-gram phone LM. To bias it to the target domain, we use interpolated counts from source and target domains as in [62].

5.4.2 Numerator term

We compute the first term in (5.2) i.e. the numerator term as a summation over HMM state sequences $\pi = s_1 \dots s_T$ in the numerator graph $\mathcal{G}_{\text{Num}}$ created by decoding the utterance using the teacher network. This is the same numerator graph that is generated for the semi-supervised LF-MMI training described in Section 2.6. This is also where we differ from [25]. In [25], this summation is done over the weak denominator graph $\mathcal{G}_{\text{Den}}$. However, we are doing this summation over a lattice-based supervision that is generated using a strong 3-gram or 4-gram word LM. Our results in Section 5.5 show that using a strong LM here is generally better. This is also easier to implement since the lattice-based supervision is already generated for semi-supervised LF-MMI training.

5.4.3 Derivative computation

Since teacher network is fixed, the derivative of $\mathcal{F}_{\text{KL}}$ w.r.t. the student network output of utterance r at time t, $y^{(rt)}(j;\lambda)$, is:

$$\frac{\partial \mathcal{F}_{\text{KL}}}{\partial y^{(rt)}(j;\lambda)} = \gamma_{rt}^{NUM}(j;\lambda^*) - \gamma_{rt}^{DEN}(j;\lambda), \tag{5.3}$$

where $\gamma_{rt}^{NUM}(j;\lambda^*)$, the numerator posterior, is the posterior probability of senone j at time t computed over the numerator graph $\mathcal{G}_{\text{Num}}$ using the teacher network and $\gamma_{rt}^{DEN}(j;\lambda)$, the denominator posterior, is the posterior probability of senone j at time t computed over the denominator graph $\mathcal{G}_{\text{Den}}$ using the student network. These are computed as:

$$\gamma_{rt}^{NUM}(j;\lambda^*) = \frac{\sum_{\pi \in \mathcal{G}_{\text{Num}}} \delta_{rt}(j)\mathbb{P}(\mathbf{O}^{(r)} \mid \pi;\lambda^*)\mathbb{P}(\pi)}{\sum_{\pi' \in \mathcal{G}_{\text{Num}}} \mathbb{P}(\mathbf{O}^{(r)} \mid \pi';\lambda^*)\mathbb{P}(\pi')}, \tag{5.4}$$

$$\gamma_{rt}^{DEN}(j;\lambda) = \frac{\sum_{\pi \in \mathcal{G}_{\text{Den}}} \delta_{rt}(j)\mathbb{P}(\mathbf{O}^{(r)} \mid \pi;\lambda)\mathbb{P}(\pi)}{\sum_{\pi' \in \mathcal{G}_{\text{Den}}} \mathbb{P}(\mathbf{O}^{(r)} \mid \pi';\lambda)\mathbb{P}(\pi')}, \tag{5.5}$$

where $\delta_{rt}(j)$ is 1 iff HMM state s_t in sequence π corresponds to senone j and 0 otherwise. Both the numerator and denominator posteriors are computed over their respective graphs using forward-backward algorithm [28].

5.4.4 LF-MMI and sequence-KL

From (5.3), we can see that the derivative is the difference of the numerator posterior computed using the *teacher network* and the denominator posterior computed using the student network. Note that this differs only in the first

term from the derivative of the MMI objective, which is as follows:

$$\frac{\partial \mathcal{F}_{\text{MMI}}}{\partial y^{(rt)}(j;\lambda)} = \gamma_{rt}^{NUM}(j;\lambda) - \gamma_{rt}^{DEN}(j;\lambda), \tag{5.6}$$

where the first term is the numerator posterior computed using the *student network* i.e. with λ instead of λ^* in (5.4).

To use an interpolation of the two objectives, we can simply interpolate the numerator posteriors from teacher and student networks. This is a sequence-level analogue to the knowledge distillation idea [68], and this was also explored in [74]. In our work, we always compute the numerator of the LF-MMI objective using a supervision lattice generated using a strong 3-gram word LM. But in Section 5.5.3.3, we investigate computing the numerator of the sequence-KL objective using a different supervision lattice such as one generated using a weak LM like a unigram LM.

5.5 Teacher-student learning experiments

We compare semi-supervised LF-MMI and sequence-level KL divergence for domain adaptation in the following scenario – Clean to noisy speech, 8kHz to 16kHz speech, and close-talk to far-field microphone speech.

5.5.1 Experimental setups

5.5.1.1 Clean to noisy speech

In this section, we describe the experimental setups for teacher-student learning for domain adaptation from clean to noisy speech.

The training data consists of 1800 hours of Fisher English [51]. Of this, we

used 300 hours (by randomly choosing a subset of speakers) as supervised data with transcription, and all the 1800 hours as unsupervised data without transcription [2]

To create the parallel noisy speech corpus, we augment the clean speech data with reverberation generated using synthetic RIRs and additive noise from the MUSAN corpus as described in [70]. We create parallel noisy speech for both supervised data and unsupervised data. For each clean utterance, we create 3 augmented versions by randomly picking RIRs and additive noises.

The systems are evaluated using WER and WER Recovery rate (WRR) on *dev* and *test* sets, which are 3 hour subsets heldout from the Fisher English corpus, but reverberated and corrupted with noise. These are part of Kaldi [58] Aspire recipe. We also report results on the official *aspire* development set [75].

5.5.1.2 8kHz to 16kHz speech

In this section, we describe the experimental setups for teacher-student learning for domain adaptation from 8kHz to 16kHz speech.

Supervised data We use Fisher English corpus for the supervised data. This consists of ~1800 hours of data in the training set with audio sampled at 8kHz rate. Of this, we selected a subset of speakers corresponding 300 hours of data as the supervised data. We use a 3x augmented version with RIRs and

[2]Note that a part of the data is used both as supervised data with transcription and unsupervised data without transcription. This is similar to using "soft labels" in standard teacher-student training. This was done for computation benefits since we could dump the features and lattices for all the 1800 hours together as required for some of the experiments that used only unsupervised data for training.

noises. This is same as the noisy version of the supervised data described in the previous section. We create a parallel corpus of 16kHz speech from this by upsampling the audio to 16kHz rate.

Unsupervised data We use AMI-IHM corpus [76] for the unsupervised data. It consists of ~80 hours of audio. The audio is recorder using a headset microphone at 16kHz rate. We create parallel corpus of 8kHz speech from this by downsampling the audio recordings to 8kHz rate.

There are multiple sources of mismatch here including bandwidth, channel and language domain. However, we only have parallel data to deal with the bandwidth mismatch (8kHz vs. 16kHz).

5.5.1.3 Close-talk to far-field microphone speech

In this section, we describe the experimental setups for teacher-student learning for domain adaptation from close-talk to far-field microphone speech.

As supervised dataset, we use the AMI corpus [76], which has parallel data in individual headset microphone (IHM) and single distant microphone (SDM) conditions. As unsupervised datasets, we have two – Mixer 6 [77] and ICSI [78]. The datasets for training and test are described below.

AMI corpus: Augmented multi-party interaction (AMI) corpus [76] collected meeting recordings in multiple locations. Each meeting had 4-7 speakers, each fit with a headset microphone. This data corresponds to the Individual Headset Microphone (IHM) condition. The meeting was also simultaneously recorded using a far-field array microphone. The data

from the first microphone corresponds to the Single Distant Microphone (SDM) condition. We use the data from IHM and SDM conditions as parallel data while training the ASR systems. We evaluate our systems on the AMI data by reporting the performance on the official *dev* and *test* heldout meetings for ASR [76].

ICSI corpus: ICSI corpus is also a meeting corpus similar to AMI, but collected at ICSI in Berkeley. The setting for this is similar to that of AMI. Each meeting involved 3-10 speakers with each speaker fit with a lapel or headset microphone. This data corresponds to the Individual Headset Microphone (IHM) condition. The meeting was also simultaneously recorded using far-field desktop microphone. The data from the 4 individual channels of this far-field microphone corresponds to the Single Distant Microphone (SDM) condition. We use the data from IHM and SDM conditions as parallel data, but we use only the audio and discard the transcripts. We evaluate our systems on the ICSI data by reporting the performance on the official *dev* and *test* heldout meetings for ASR [78].

Mixer-6 corpus: Mixer 6 consists of audio recordings of interviews, transcript readings and conversational telephone speech from native American English speakers. We only use the conversational telephone speech part of the corpus. In particular, we use only one side of it, which was recorded simultaneously using 14 microphone channels. We use data from the MIC02 channel as the close-talk microphone data. We use data from MIC04-14 as the parallel far-field microphone data. Since the

same speech is recorded in multiple channels, we randomly sample the far-field data to be around 3x the amount of close-talk speech data. The Mixer-6 corpus is not transcribed. So we evaluate our systems on the Mixer-6 data by reporting the performance on the 10 hours IARPA *Aspire dev* set.

5.5.2 System descriptions

All the neural networks in our experiments have an architecture with time-delay neural network (TDNN) [60], [79] layers interleaved with LSTM [80] layers. We use per-frame dropout on the LSTM layers [81]. The reader is directed to [81] for training details. To avoid over-fitting, we apply the regularization methods suggested in [2] for both LF-MMI and sequence-KL training. We use online i-vectors [61], [82], [83] for speaker adaptation. Our method for creating lattice-based supervisions is described in Sections 2.6 and 5.4.

The teacher and the student networks use i-vectors extracted from different i-vector extractors trained on their respective domains.

5.5.2.1 Clean to noisy speech

In this section, we describe the systems used for teacher-student learning for adaptation from clean to noisy speech. The corresponding experimental setup is described in Section 5.5.1.1.

Baseline system The "Baseline" system is trained with LF-MMI objective on 300 hours supervised data from Fisher English that is augmented 3x with

reverberation and noise [70] and 3x with speed and volume perturbation [71] (Hence 9x300 hours).

Oracle system The "Oracle" system is trained with LF-MMI objective using as supervised data all 1800 hours augmented 3x with reverberation and noise.

Teacher network The teacher network is trained on "clean" 300 hours supervised data with only 3x speed perturbation, but with no reverberation or noise addition.

Student network The teacher network is used to decode the whole 1800 hours[3] of "clean" Fisher data. For this decoding, we use a 3-gram LM trained on transcripts from the 300 hours supervised set. These lattices are also used to create supervision for the parallel noisy corpus used to train the student network. The student network is trained either only on the unsupervised noisy data or in a multitask framework on both supervised and unsupervised data. The training on supervised data uses LF-MMI with supervision from a GMM system, while the training on unsupervised data uses an interpolated objective $(1-\beta)\mathcal{F}_{\text{MMI}} + \beta\mathcal{F}_{\text{KL}}$. The denominator graph for both is generated using 4-gram phone LM created by averaging counts from supervised data phone transcription and 1-best phone hypotheses from unsupervised data. For LF-MMI, we convert the lattice into numerator supervision as described in 4.2 using smart splitting with a pruning beam of 4 and *lm-scale* of 0.5 interpolating

[3]Note that this includes the 300 hours of audio from supervised dataset, but we are only using the audio and not the transcripts. This is like using soft posteriors for labeled data in conventional T-S learning [68].

word LM and the phone LM scores. For sequence-KL, we use the posteriors obtained from the decoded lattices interpolated with the phone LM using a weight of 0.5.

We use the same i-vector extractor for baseline, oracle and all the student networks. This is trained on 1800 hours of Fisher data augmented 3x with reverberation and noise.

5.5.2.2 8kHz to 16kHz speech

In this section, we describe the systems used for teacher-student learning for domain adaptation from 8kHz to 16kHz speech using Fisher English as the supervised dataset and AMI-IHM as the unsupervised dataset. The corresponding experimental setups is described in Section 5.5.1.2.

Baseline network and Teacher network The “Baseline" network here is same as the one in Section 5.5.2.1. This is also the teacher network for T-S learning.

Student network The teacher network is used to decode the target AMI-IHM data (downsampled to 8kHz to use with Fisher’s teacher network) to generate lattices. A 3-gram Fisher LM is used for this decoding. While the lattices are generated using the 8kHz AMI-IHM data, we use the same with the parallel 16kHz AMI-IHM data for training the student network. The student network is trained either only on the unsupervised AMI-IHM data or in a multitask framework on both supervised Fisher and unsupervised AMI-IHM data. For multitask training, we share all the layers of the network

for both the supervised and unsupervised data. The i-vector extractor used with the networks is also trained on the corresponding data. The training on supervised Fisher English data uses LF-MMI with supervision from a Fisher English GMM system, while the training on unsupervised data uses an interpolated objective $(1-\beta)\mathcal{F}_{\text{MMI}} + \beta\mathcal{F}_{\text{KL}}$. The denominator graph for both is generated using 4-gram phone LM created by averaging counts from Fisher English phone transcription and 1-best phone hypotheses from AMI-IHM. For LF-MMI, we convert the lattice into numerator supervision as described in 4.2 using smart splitting with a pruning beam of 4 and *lm-scale* of 0.5 interpolating word LM and the phone LM scores. For sequence-KL, we use the posteriors obtained from the decoded lattices interpolated with the phone LM.

5.5.2.3 Close-talk to far-field microphone speech

In this section, we describe the systems for teacher-student learning for adaptation from close-talk to far-field microphone speech. The experimental setup for this is described in Section 5.5.1.3.

AMI baseline system For training the baseline system, we use AMI-SDM data mixed with AMI-IHM data augmented with reverberation and noise. For supervision, we generate lattices using a GMM system for AMI-IHM data and use it for the parallel reverberated AMI-IHM data and AMI-SDM data as done in [72].

Oracle systems We have two "oracle" systems – one for ICSI corpus and one for Mixer-6 corpus, all in the far-field microphone condition. We use these

to compare with the performance of unsupervised adaptation systems. The first oracle system is trained only on the ICSI corpus in a supervised fashion analogous to the AMI baseline system described in the previous section. The second oracle system is trained only on 300 hours of Fisher English augmented with synthetic RIRs and additive noise from the MUSAN corpus. We could not train an oracle system on the Mixer-6 corpus itself since we do not have the transcripts for that. Also, the Fisher English corpus was the official training corpus for the IARPA Aspire challenge, which is the test set we are evaluating on.

Teacher network As teacher network, we use a TDNN-LSTM network trained on AMI-IHM data that is mixed with reverberated and noise augmented version of the same. This teacher network was selected as it gave the best performance on AMI-IHM *dev* and *eval* sets.

Mixer 6 student network For adaptation using mixer 6 data, we decode the mixer 6 headset microphone (MIC02) data using the teacher network and a 3-gram Fisher word LM to generate lattices. These lattices are converted into supervision for data from the parallel far-field microphones (MIC04-MIC13). Since the same data was recorded in multiple microphones, we kept a subset of only 30% of the parallel far-field data.

ICSI student network For adaptation using ICSI data, we decode the ICSI-IHM data using the teacher network and the 3-gram AMI word LM to generate lattices. These lattices are converted into supervision for data from the parallel

ICSI-SDM data. We used all 4 available distant microphones, but adjusted the training to train on these for one-fourth the number of epochs as the rest of the data.

In both the baseline and T-S learning networks, we use the same i-vector extractor, which is trained on the same AMI data used to train the baseline. For both the adaptation experiments, we use semi-supervised training in a multitask architecture with supervised training with LF-MMI on the same AMI data as the baseline network and unsupervised training also with LF-MMI on a mix of augmented (using reverberation and noise addition) headset microphone data and distant microphone data from Mixer 6 or ICSI corpora.

5.5.3 Results and discussion

The results for adaptation from clean to noisy speech are in Table 5.1. The experimental setup and the description of the systems are in Sections 5.5.1.1 and 5.5.2.1 respectively. The columns "sup" and "unsup" show the amount of supervised and unsupervised data (prior to augmentation) respectively used in training the student network.

The results for adaptation from 8kHz to 16kHz speech are plotted in Figures 5.3 and 5.4 for various configurations and various values of β, the weight of interpolation of sequence-KL objective with LF-MMI objective. The best configuration is compared with the baseline and the oracle systems in 5.2.

The results for adaptation from close-talk microphone speech to single distant microphone speech are shown in Table 5.3. The table shows the supervised and unsupervised data used for training of the systems.

Table 5.1: WER(%) results for unsupervised adaptation from clean to noisy. The objective is $(1-\beta)\mathcal{F}_{\text{MMI}} + \beta\mathcal{F}_{\text{KL}}$.

System	β	hrs		WER (%)			WRR
		sup	unsup	*dev*	*test*	*aspire*	(%)
Baseline	0	300	0	23.6	22.5	26.6	0
Unsup only	0	0	1800	23.0	22.0	27.0	6
Unsup only	1	0	1800	21.8	21.0	25.9	34
Semisup multitask	0	300	1800	21.6	21.0	25.1	42
Semisup multitask	1	300	1800	21.0	20.3	24.4	59
Semisup multitask	0.5	300	1800	**21.0**	**20.2**	**24.2**	61
+ unigram LM	0.5	300	1800	21.2	20.6	24.5	54
Oracle	0	1800	0	19.1	18.4	23.3	100

Figure 5.3: 8kHz Fisher ->16kHz AMI-IHM WER(%) results: Unsupervised vs semi-supervised multitask training. The solid lines show results on *AMI eval* and dashed lines on *AMI dev*.

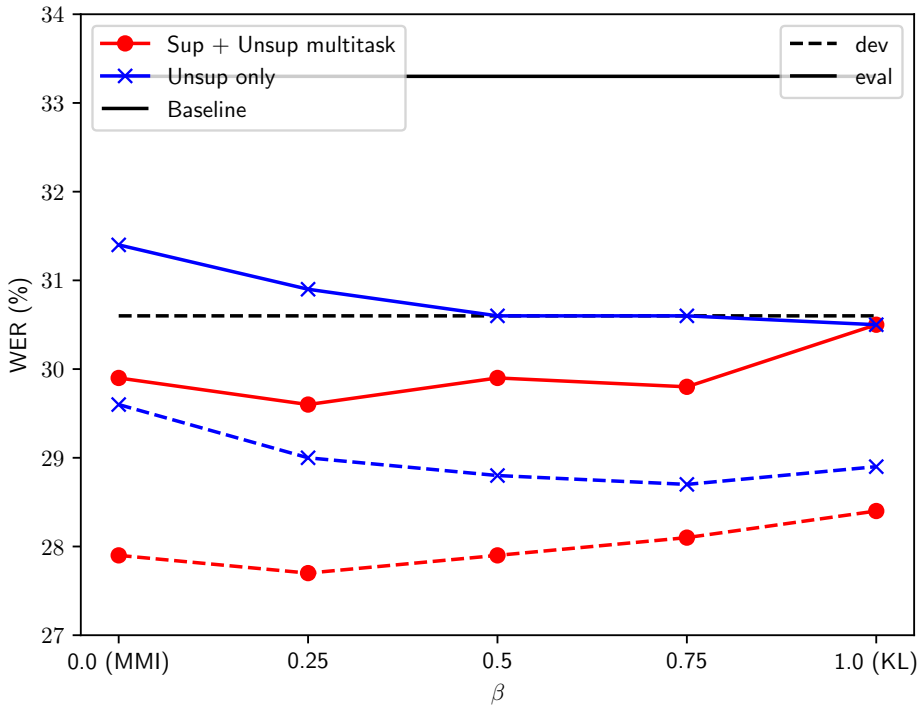


Figure 5.4: 8kHz Fisher ->16kHZ AMI-IHM WER(%) results: Unigram vs 3-gram for sequence-KL. The solid lines show results on *AMI eval* and dashed lines on *AMI dev*.

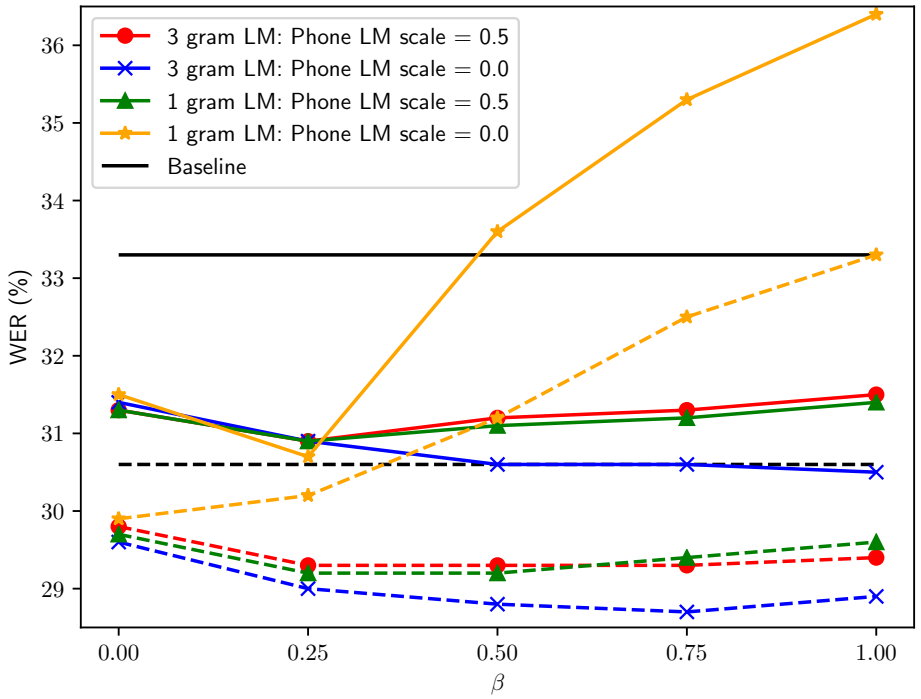


Table 5.2: WER(%) results for 8kHz Fisher to 16kHz AMI-IHM

System	Source domain		Target domain			WER		WRR
	sup (hrs)	Rate (kHz)	sup (hrs)	unsup (hrs)	Rate (kHz)	*dev* (%)	*eval* (%)	(%)
Baseline	300	8	0	0	8	30.6	33.3	0
T-S	300	16	0	80	16	27.7	29.6	24
Oracle	0	-	80	0	16	18.7	18.6	100
Oracle	0	-	80	0	8	20.4	19.9	89

5.5.3.1 Effect of multitask learning

In Table 5.1, rows 4-6 show results with semi-supervised multitask learning using LF-MMI on supervised data and an interpolation of LF-MMI and

Table 5.3: WER(%) results for adaptation from close-talk to distant microphone

	Training data		AMI-SDM		ICSI-SDM		Aspire
System	sup	unsup	*dev*	*eval*	*dev*	*eval*	*dev*
Baseline	AMI	-	33.8	37.0	43.9	42.9	41.4
T-S	AMI	ICSI	32.9	36.5	36.0	31.5	-
+ domain den-fst	AMI	ICSI	32.9	36.9	36.1	31.4	-
T-S	AMI	Mx6	34.0	37.1	-	-	33.2
+ domain den-fst	AMI	Mx6	**33.3**	**36.8**	-	-	**32.0**
Oracle	ICSI	-	-	-	30.2	27.9	-
Oracle	Fsh300h	-	-	-	-	-	26.6

sequence-KL on unsupervised data. We see that any of these multitask training systems is better than the systems that train only on the unsupervised data.

As described in Section 5.2, multitask training can effectively mitigate mismatch in acoustic and language domains. We see this in the results for domain adaptation from Fisher English to AMI-IHM shown in Figure 5.3. WERs on *AMI dev* and *AMI eval* are shown for various interpolation factors. We see that semi-supervised multitask training (Red $\circ$) is better than training only on the unsupervised data (Blue $\times$) in the target domain. The former gives an improvement of around 3% absolute over the "Baseline" in the best configuration of $\beta = 0.25$. It is possible that training on a larger amount of data and also regularizing with supervised Fisher data (even if out-of-domain) is helping the cause here [4].

From these results, we conclude that semi-supervised multitask training is significantly better than training only on unsupervised data. This is the

[4]In addition, we have an i-vector extractor that is also trained on a larger dataset (including Fisher), but based on the results in [26] for a similar adaptation experiment from Switchboard to AMI, we do not think this itself gives a significant improvement over just training i-vector extractor on the AMI data as done for the results in the previous sections.

case even when there is a big domain mismatch between supervised and unsupervised data as in the case of Fisher vs AMI.

5.5.3.2 Effect of LF-MMI and sequence-KL

From rows 2 and 3 of Table 5.1 showing results from training *only* on the unsupervised data, we see that sequence-KL is significantly better than using LF-MMI. The WER with LF-MMI is worse than even the baseline on the *aspire* set. However, when using multitask learning, the difference between the systems is much smaller. Using either sequence-KL (Row 5) or an interpolation of LF-MMI and sequence-KL (Row 6) is slightly better than using LF-MMI (Row 3).

However, this does not seem to hold in generic cases where there is a domain mismatch between supervised and unsupervised data as in the case of adaptation from 8kHz to 16kHz speech. We see this from the results in Figure 5.3. The red lines in the figure show WER results for different interpolation weights, β, between sequence-KL and LF-MMI. The WER is better for LF-MMI and the optimal is at $\beta = 0.25$. We believe that since the domains of the data used to train the teacher and student networks are different (Fisher English vs. AMI), the numerator posteriors from the teacher are not very good for training the student using sequence-KL. But, we get better posteriors from the student network by training using LF-MMI.

From these results, we conclude that sequence-KL is more robust than LF-MMI when training only on unsupervised data. But, when multitask training is used, the difference between sequence-KL and LF-MMI is less significant.

Since, multitask training is quite effective in handling domain mismatch, we recommend using LF-MMI or an interpolation of LF-MMI and sequence-KL with a higher weight on LF-MMI.

5.5.3.3 Effect of LM used for numerator computation

For semi-supervised training, it is generally better to use a strong LM for decoding unsupervised data to generate lattices. From [18], the best phone LM scale for interpolating normalization FST's phone LM scores and lattice's word LM scores when generating numerator supervision is 0.5.

In this section, we try to find:

1. the best phone LM scale (0.0 vs 0.5) for interpolating phone LM and word LM scores to get numerator posteriors for sequence-KL
2. the best LM (3-gram vs 1-gram) to use for generating lattices to get numerator posteriors for sequence-KL

The legend in Figure 5.4 shows the LM used for decoding and the phone LM scale when generting supervision for sequence-KL objective. In these experiments, we use the interpolated objective $(1-\beta)\mathcal{F}_{\text{MMI}} + \beta\mathcal{F}_{\text{KL}}$. Note that the LF-MMI objective in all cases uses a 3-gram word LM for decoding and a phone LM scale of 0.5 for interpolating phone LM and word LM scores.

From Figure 5.4, when using a 3-gram word LM for decoding, using a phone LM scale of 0.0 (Blue $\times$) works better than a phone LM scale of 0.5 (Red $\circ$). However, when using a 1-gram LM for decoding, the WER degrades with a phone LM scale of 0.0 (Orange $*$) and gets even worse than baseline for

large β. This problem is alleviated if a phone LM scale of 0.5 is used (Green $\triangle$), but is still worse than using 3-gram word LM for decoding.

We observe the same effect even for adaptation from clean to noisy speech as can be seen in Table 5.1. Using a unigram LM (Row 7) is worse than using a 3-gram LM (Row 6) for decoding when generating numerator posteriors for sequence-KL objective (while still using 3-gram for generating lattices for MMI training),

From these results, we conclude that for unsupervised domain adaptation, it is better to use a strong LM like 3-gram for generating numerator supervision. This is also computationally advantageous because using strong 3-gram LM requires only a single generation of lattices for both MMI and sequence-KL, while using 1-gram LM requires regeneration of lattices for sequence-KL. Further, when using a strong word LM, interpolating the LM scores with phone LM scores is not required and using a phone LM scale of 0.0 works the best.

The performance degradation when using a weak LM was also reported in [25] for unsupervised speaker adaptation. But we believe the degradation is larger in our case because we are training the student network from scratch instead of initializing from the teacher network. However, initializing from teacher network is not straight-forward in our case since the input features are different (16kHz vs 8kHz).

5.5.3.4 Overall impact of T-S learning

We discuss in this section the impact of semi-supervised teacher-student learning for domain adaptation based on the results in Tables 5.1, 5.2 and 5.3

As can be seen in Table 5.1, for adaptation from clean to noisy speech, we see that our proposed approach multitask learning approach training on supervised data using LF-MMI and on unsupervised data using interpolation of LF-MMI and sequence-KL gives a WER recovery of 61%. Using only LF-MMI on both types of data gives a lesser 42% WER recovery.

Similarly, we get good results for adaptation from close-talk to distant microphone as can be seen in Table 5.3. For the T-S learning systems, we used just the LF-MMI objective, which is quite effective especially in cases of language domain mismatch. The results show that using parallel data in Mixer 6 or ICSI for adaptation from IHM to SDM improves WER in the distance microphone condition. The performance improvement is small on the AMI-SDM test sets since the baseline here is already state-of-the-art on that task. But we see significant improvement in the mismatched condition when evaluating on ICSI-SDM or Aspire test sets. In these cases, the baseline system is mismatched to the test data and its performance is very poor compared to the corresponding oracle systems. But using T-S learning, we are able to get a WER recovery rate of 50% or more. As can be seen in the rows with "+ domain den-fst", we improve WER slightly by using a domain-specific phone LM and denominator FST described in Section 5.2.3.1 But this shows a significant difference only when using Mixer 6 data, which is mismatched with the source AMI-IHM data.

However, the WER recovery rate is smaller (at 24%) for adaptation from 8kHz Fisher English to 16kHz AMI-IHM as can be seen in Table 5.2. The best multitask training system here is quite behind the "Oracle" system (Row 3) in Table 5.2. This is the case even when considering an "Oracle" system trained on 8kHz data (Row 6), which only degrades performance by less than 2% over using 16kHz data. This suggests that bandwidth mismatch by itself is not a major issue in these experiments, but that other forms of domain mismatch such as language mismatch (dialect, topics etc.) between Fisher and AMI are more prominent. One prominient issue is that we decoded the unsupervised AMI data using a Fisher LM because we did not have text matching the AMI corpus (other than the training transcripts, but using that would be cheating).

From these results, we conclude that semi-supervised teacher-student learning is very effective when parallel data is available in source and target domains. It shows good WER recovery rate for adaptation from clean to noisy speech and close-talk to distant microphone speech. It is not effective by itself in dealing the adaptation from 8kHz Fisher to 16kHz AMI, because in addition to bandwidth mismatch for which parallel data is available, there are also other mismatches in acoustic and language domains. A better target-domain matched LM is required as suggested in Section 5.2.3. This is explored in the next set of experiments.

5.6 Domain mismatch experiments

We explore semi-supervised transfer learning across mismatch domains in the following scenario – AMI-IHM to Tedlium corpus, Tedlium to *How2 challenge*

corpus and Fisher English to *Fearless steps* corpus.

5.6.1 Experimental setups

5.6.1.1 AMI-IHM to Tedlium

In this section, we describe the experimental setups for semi-supervised transfer learning experiments from the AMI-IHM corpus to Tedlium corpus. As supervised dataset, we use the AMI-IHM corpus and as unsupervised dataset we use Tedlium corpus [84], [85].

AMI-IHM corpus is described in detail in Section 5.5.1.3. This is the training data for the source domain and is supervised consisting of 80 hours of audio and corresponding transcripts.

The Tedlium corpus is made from TED talks and their transcriptions available in the TED website [5]. These were filtered and cleaned for the purposes speech processing tasks as described in [84], [85]. Here, we use the Release 3 of this corpus, which includes 2351 TED talks and transcriptions corresponding to 452 hours of audio data. We used the dictionary as well as the language modeling data released with this to get the pronunciation lexicon and the language model for decoding. We followed the Kaldi recipe [6] for this purpose.

The target domain data is 452 hours of data from the *train* set of the Tedlium release 3 corpus. We only use the audio from this data, and use its transcripts for neither acoustic model training nor language model training. The *dev* and *test* sets from the Tedlium corpus are the datasets that we evaluate the system performance on.

[5] http://www.ted.com/

[6] https://github.com/kaldi-asr/kaldi/tree/master/egs/tedlium/s5_r3

5.6.1.2 Tedlium to How2 challenge corpus

In this section, we describe the experimental setups for semi-supervised transfer learning from the Tedlium corpus to the *How2 challenge* corpus [86].

Source domain We use as supervised data the Release 3 of the Tedlium corpus [85]. This is as described in Section 5.6.1.1, and consists of 452 hours of audio data with corresponding transcriptions.

Target domain The target domain is the instructional videos from YouTube collected in the *How2 challenge* corpus. This corpus was designed to have 80,000 instructional videos (about 2,000 hours) with associated English sub-titles and summaries. A 300 hour subset of this with cleaned sub-titles was released for use during the *JSALT 2018 Workshop*. For our work, we use this 300 hours set for a supervised domain adaptation baseline. For unsupervised domain adaptation experiments, as unsupervised data, we use a separately collected set of videos from the `expertvillage` YouTube channel [7]. This consists of about 100,000 videos (about 2200 hours). We excluded the videos that were contained in the 300 hours supervised set.

5.6.1.3 Fisher English to *Fearless steps* challenge corpus

In this section, we describe the experimental setups for semi-supervised transfer learning from Fisher English to the *Fearless steps* challenge corpus [87].

[7] `https://www.youtube.com/expertvillage`

Source domain The training data for the source domain is the supervised training data from the Fisher English corpus.

Target domain *Fearless steps* corpus is an effort by CRSS-UTDallas to digitize the audio from the Apollo missions into a corpus for various spoken language technology tasks. Around 19,000 hours of data from Apollo 11, 13 and 1 missions was released with the name "Fearless Steps". The corpus was released to the community along with a set of 5 challenge tasks. In this work, we focus on only the ASR task. The challenge conditions allow us to utilize any speech and language data external to the *Fearless steps* corpus, but only a subset of channels and recording from the *Fearless steps* corpus. This subset amounts to around 9000 hours. Since the data is conversational and some of the audio is very noisy, we chose the augmented Fisher English corpus as the training data for seed acoustic model training to apply transfer learning to the *Fearless steps* domain.

But the tricky part is to build an in-domain vocabulary and language model that is matched to the *Fearless steps* corpus because we did have any matched language modeling data. For this reason, we resort to collecting data from the web. We used several sources of data as suggested in [88], [89] related to Apollo missions and NASA. It included materials from books [8], mission reports [90] and transcriptions [91], [92].

All the data available for training is unsupervised (~9000 hours) and is used for semi-supervised transfer learning experiments. A small amount of ~20 hours data (~4 hours after segmentation) was manually transcribed

[8] https://www.nasa.gov/connect/ebooks/index.html

and was released by the challenge organizers as the *Fearless steps dev* set. We evaluate our systems on this dataset. Another ~60 hours of data (~20 hours after SAD and segmentation) was also released by the organizers as the *Fearless steps train* set. But this is not manually transcribed but rather contains ASR-generated transcripts. So we do not use this set for acoustic model training, but only use it for language model training.

5.6.2 System descriptions

5.6.2.1 AMI-IHM to Tedlium

In this section, we describe the systems used for semi-supervised transfer learning from AMI-IHM corpus to Tedlium corpus.

AMI-IHM baseline We use as baseline a system completely trained on the source domain, AMI-IHM corpus. This is a TDNN+LSTM neural network acoustic model trained using clean AMI-IHM data along with a reverberated copy of AMI-IHM data. This system was chosen to give the best performance on the AMI-IHM evalutation sets. For supervision, we use lattices generated from a GMM system for AMI-IHM data and use it also for the reverberated AMI-IHM data as done in [70], [72].

Semi-supervised systems The unsupervised audio from the Tedlium release 3 *train* set is decoded using the "AMI-IHM baseline" acoustic model to generate lattices for semi-supervised training. We investigate using two language models as described in Table 5.4.

Table 5.4: Language models for decoding unsupervised data for adaptation from AMI-IHM to Tedlium corpus. Their perplexities (PPL) on Tedlium *dev* set are reported.

Domain	Data source	PPL	
AMI	Out-of-domain	AMI + Fisher transcripts	423
Ted	In-domain	Selected data from WMT12 corpus [93]	219

We use multitask training for semi-supervised transfer learning from AMI-IHM to Tedlium using supervised data from AMI-IHM and only unsupervised data from the Tedlium corpus. We generate lattice supervision using smart splitting with a lattice *beam* of 4 and *lm-scale* of 0.5 on word and phone LMs. The wider beam of 4 was more useful when decoding unsupervised data with AMI LM, and it did not make much difference with in-domain Tedlium LM.

Oracle system For the oracle system, we use multitask training for supervised transfer learning from AMI-IHM to Tedlium.

We use the same i-vector extractor as in the "AMI-IHM baseline" system for all the systems. This i-vector extractor is trained only on the AMI-IHM data. We did not see any significant gain on training i-vector extractor by including the Tedlium data. So for fair comparison, we used the same i-vector extractor the baseline, semi-supervised and oracle systems.

5.6.2.2 Tedlium to *How2 challenge* corpus

In this section, we describe the systems used for semi-supervised transfer learning from Tedlium corpus to *How2 challenge* corpus. For the corresponding experimental setup, the reader is directed to the description in a previous section (Section 5.6.1.2).

Tedlium baseline We use as baseline a system that is trained entirely on the source domain, the *train* set of the Tedlium 3 corpus. This is a TDNN-F neural network acoustic model. For supervision, we use lattices generated from a HMM-GMM system trained on the same Tedlium corpus. Note that these lattices are constrained to the transcripts, and only include variations of pronunciations.

Semi-supervised system The unsupervised data from the *How2 challenge* corpus is decoded using the "Tedlium baseline" system acoustic model to generate lattices. We investigate using different language models for decoding – n-gram LM as well as RNNLM:

Ted 4-gram LM The first is a baseline Tedlium n-gram LM used in the "Tedlium baseline" system, which was estimated on the Tedlium transcripts and a selection of monolingual data from WMT12 corpus [93] [9] and the Tedlium acoustic transripts. This has a perplexity of 181 on the *How2 dev* set.

How2 4-gram LM The second is a How2 n-gram LM that is in-domain to the How2 corpus and was estimated by including in the LM training data, the sub-titles from the 300 hours of videos released as part of the How2 challenge during the JSALT 2018 Workshop [86]. This has a perplexity of 101 on the *How2 dev* set.

How2 RNNLM The last is an in-domain How2 RNNLM. This was trained using the same data sources used for the How2 n-gram LM estimation.

[9] https://www.openslr.org/51/

For n-gram LM estimation, we use the pocolm toolkit [64]. For RNNLM training, we use Kaldi[94], [95]. For efficient rescoring of lattices, we use the pruned rescoring proposed in [95] with a lattice pruning beam of 4. But since the lattices are undeterminized and can have multiple phone sequeunces per word sequence, we use the modified rescoring approach described in Algorithm 1. For semi-supervised multitask learning, we use lattice-based supervision with smart splitting, pruning with a beam of 2.0, only phone LM scores ($lm\text{-}scale = 0$) on the supervision.

Oracle system As an oracle system, we train a supervised system on only the 300 hours subset of the How2 corpus that included the cleaned sub-titles. We generate lattices for training using the "Tedlium baseline" neural network acoustic model. As in standard LF-MMI, these are constrained to the word-level transcripts (sub-titles) but include pronunciation variants.

For fair comparison, we use the same i-vector extractor as for the "Tedlium baseline" system, which is trained on the Tedlium 3 training data.

5.6.2.3 Fisher English to *Fearless steps* challenge corpus

In this section, we describe the systems used for semi-supervised transfer learning from Fisher English to *Fearless steps* challenge corpus.

Aspire baseline As baseline system, we use a pre-trained Aspire LF-MMI trained TDNN+LSTM neural network acoustic model [10] as the baseline. This

[10] http://kaldi-asr.org/models/m1

model was trained using the Kaldi Aspire recipe [11].

Semi-supervised systems The unsupervised data (~9000 hours) from the *Fearless steps* corpus is unsegmented. We first segment it into 10s or smaller segments using a neural network-based speech activity detection (SAD) system. We used a pre-trained SAD system [12], which was also trained on the Aspire corpus. The segmented unsupervised data, amounting to around 2400 hours, is decoded using Aspire baseline acoustic model. To get a reasonable performance, we use an in-domain LM for decoding that was estimated by including NASA related text sources from the web mentioned in the previous section (Section 5.6.1.3). Aside from all the in-domain text from the web, transcripts from the Fisher English corpus were also included to model spontaneous conversations. A small amount of data amounting to 20 hours was released as the *Fearless steps train* set with ASR-generated transcripts. These transcripts are also used when estimating the language model. We used the same text sources to extract a vocabulary that included scientific terms and abbreviations used throughout the *Fearless steps* corpus. A multi-stage process was used to normalize the NASA text sources based on existing vocabulary, and update the vocabulary to include new words in the text sources that occur at least twice. The language model estimation and lexicon preparation steps are described in the following section.

[11] https://github.com/kaldi-asr/kaldi/tree/master/egs/aspire/s5
[12] http://kaldi-asr.org/models/m4

Language modeling We used a modified Kneser-Ney 4-gram LM [96] as the in-domain language model for the *Fearless steps* corpus. This was estimated on the n-gram counts from all the mentioned sources using the pocolm [64] toolkit. The counts from different sources were weighted in such a way as to minimize the perplexity on the tuning set. We considered two different sets for tuning:

1. ASR-generated transcripts released as the *Fearless steps train* by the *Fearless steps* challenge organizers

2. Normalized Apollo 11 transcripts obtained from Apollo Flight Journal (AFJ) [91] and Apollo Lunar Surface Journal (ALSJ) [92].

When one of the sets was used for tuning, the other was included for training. The perplexity results on the *Fearless steps dev* set as shown in Table 5.5 show that tuning on the Apollo 11 transcripts from AFJ and ALSJ gives 4 points improvement over tuning on *Fearless steps train*. This is reasonable because these transcripts are the closest to the content of the *Fearless steps* corpus, for which we do not have any manual transcription other than the *Fearless steps dev* set. The ASR-generated transcripts from the *Fearless steps train* set released by the challenge organizers are not good enough for this purpose. Even with these improvements to the LM, the perplexity on the *Fearless steps dev* was still low (best number is 114) as shown in Table 5.5, but a considerable improvement over an LM trained just on Fisher English. The results also show that excluding NASA eBooks from the LM sources gives better perplexity numbers as the eBooks turn out to be not that good a match to the spontaneous conversations and the domain of the *Fearless steps* corpus.

Table 5.5: Perplexities of language models on the manually transcribed *Fearless steps dev* set

LM Sources	Tuned on	Perplexity
Fisher English	Fisher heldout	451
+ NASA	*Fearless steps train*	122
+ NASA	Apollo 11 (AFJ, ALSJ)	118
- eBooks	*Fearless steps train*	118
- eBooks	Apollo 11 (AFJ, ALSJ)	114

Lexicon Since we need a pronunciation lexicon for ASR training, we also get pronunciations for all the words. We use the CMU pronouncing dictionary [97] as the seed lexicon, and use it to train a grapheme-to-phoneme (G2P) [98] model using phonetisaurus [99] toolkit. The G2P model is used to generate pronunciations for the new non-abbreviation words in the expanded vocabulary.

Since the text sources from the web are all normalized differently it is not possible to easily figure out whether something is an abbreviation or not. We used an adhoc method to consider a capitalized sequence of letters as an abbreviation if that is not in the lexicon, and if it was already in the lexicon we assume that it's a regular non-abbreviation word. For the abbreviations in the expanded vocabulary, we add the pronunciation by spelling the individual letters in the abbreviation. The pronunciation of the individual letters are obtained from the CMU dictionary.

5.6.3 Results and Discussion

In this section, we report experimental results of semi-supervised transfer learning in the following scenario – AMI-IHM to Tedlium, Tedlium to *How2*

challenge corpus, Fisher English to *Fearless steps* corpus.

The results for adaptation from AMI-IHM to Tedlium are shown in Table 5.6. The LM used for decoding unsupervised data to generate lattice-based supervision for semi-supervised training is shown in the column "LM". The results comparing performance for different amounts of unsupervised data is shown in Figure 5.5. The LM used for decoding unsupervised data is indicated in the legend. The figure also shows the performance using three different training strategies:

init from baseline Here, we train only on unsupervised data by updating the seed model. This corresponds to unsupervised weights transfer in Figure 5.2.

train from scratch Here, we train only on unsupervised data but we train from scratch instead of initializing the model from a seed model.

multitask training This is semi-supervised multitask transfer learning on source-domain supervised data and target-domain unsupervised data.

The results for adaptation from Tedlium to How2 corpus are shown in Table 5.7. The LM used for decoding unsupervised data to generate lattice-based supervision for semi-supervised training is shown in the column "LM". WER results are shown on the *How2 dev* using the *How2 4-gram* LM for decoding as well as the corresponding numbers after rescoring with *How2 RNN* LM.

The results for adaptation from Fisher English to *Fearless steps* corpus are shown in Table 5.8. The amount of supervised source-domain and unsupervised target-domain data used for training each system is shown in the

Table 5.6: Domain adaptation results for adaptation from AMI-IHM to Tedlium showing WER(%) on *Tedlium dev* and *Tedlium test* sets

System	Type	LM	*dev*	*test*	WRR(%)
AMI-IHM baseline	-	-	18.8	19.4	0
Semisup multitask	shared den-fst	*AMI*	14.8	13.8	46
	domain den-fst	*AMI*	14.8	13.8	46
	separate outputs	*AMI*	15.3	14.2	42
	shared den-fst	*Ted*	12.9	12.2	63
	domain den-fst	*Ted*	**12.6**	**12.2**	**64**
	separate outputs	*Ted*	13.2	12.3	61
Tedlium oracle	shared den-fst	-	8.7	8.6	100
	separate outputs	-	9.2	9.0	96

Figure 5.5: Domain adaptation WER(%) results for adaptation from AMI-IHM to Tedlium corpus for different amounts of target-domain unsupervised data

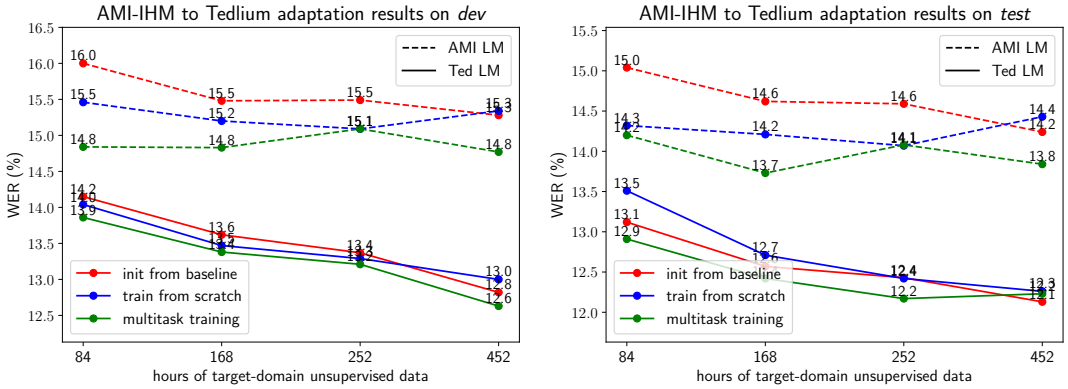


table.

5.6.3.1 Effect of language model mismatch

In this section, we discuss the effect of the mismatch between the domains of the language model used for decoding and the unsupervised target-domain data.

In Table 5.6, rows 2-4 and rows 5-7 respectively show results for adaptation

Table 5.7: WER(%) results on *How2 dev* set

System	Ted (hrs)	How2 (hrs) sup	How2 (hrs) unsup	LM	*how2 dev* 4-gm	*how2 dev* RNNLM
Tedlium baseline	452	0	0	-	18.7	16.8
Semisup multitask	452	0	2200	*Ted 4-gram*	17.0	16.0
Semisup multitask	452	0	2200	*How2 4-gram*	16.4	15.4
Semisup multitask	452	0	2200	*How2 RNN*	**16.2**	**15.2**
Supervised How2	0	300	0	-	15.9	14.8

Table 5.8: Semi-supervised transfer learning results on *Fearless steps dev* set

System	Type	Data (hrs) sup	Data (hrs) unsup	WER (%)
Aspire baseline	-	1800	0	38.8
Only unsup	train from scratch	0	180	36.6
	init from baseline	0	180	34.1
Semisup multitask	-	300	180	34.2
Only unsup	train from scratch	0	2400	33.9
	init from baseline	0	2400	34.2
Semisup multitask	-	300	2400	34.0

from AMI-IHM to Tedlium using two different LMs for decoding unsupervised data – an out-of-domain *AMI LM* and in-domain *Ted LM*. The LMs are described in Section 5.6.2.1. The results show a much larger WER recovery (64% at best) with *Ted LM* compared to 46% with *AMI LM*. Similarly, in Table 5.7, for adaptation from Tedlium to *How2 challenge* corpus, we see that using the in-domain *How2 4-gram LM* is 0.6% absolute better (17.0 vs 16.4) than using the out-of-domain *Ted 4-gram LM*.

Figure 5.5 shows WER on *Tedlium dev* and *Tedlium test* sets for different amounts of unsupervised data for adaptation from AMI-IHM to Tedlium corpus. When the mismatched *AMI LM* is used, the improvement is smaller or there is even degradation on increasing the amount of unsupervised data.

We observe that that increasing the amount of unsupervised data in the target-domain is helpful, but only if we also use a matched LM to decode the unsupervised data. There is not much benefit in increasing the unsupervised data if we do not use a matched LM to decode that data. From these results, we conclude that it is very important for the language model used to decode the unsupervised data to match the target domain so that we get better supervision lattices for semi-supervised training.

5.6.3.2 Effect of domain-specific phone LM and separate output layer

In this section, we discuss three approaches to handling language domain mismatch between source and target data:

shared den-fst using a shared phone LM trained using phone counts from source and target domains as described in Section 5.2.3

domain den-fst a target domain-specific phone LM and denominator FST as described in Section 5.2.3.1

separate outputs separate output layers during multitask training.

As can be seen from the results in Table 5.6, having separate outputs (rows 4, 7 and 9) for the two domains degrades the performance. This is the case even the target-domain data is supervised as can be seen from the WER results of the Tedlium oracle systems (row 9).

As can also be seen from the results in Table 5.6, using target domain-specific phone LM gives almost the same performance as or is slightly better than using a shared phone LM.

Based on these results, we conclude that it is better to share all the layers of the network when doing transfer learning with multitask training. We also see that using target domain-specific phone LM is about the same or slightly better than using shared phone LM. However, we recommend this approach because it allows to use for the supervised part of the data (source-domain data) the same denominator FST and training examples dumped for training the seed system.

5.6.3.3 Effect of rescoring with strong LM

In this section, we discuss the effect on semi-supervised transfer learning of rescoring lattices obtained by decoding unsupervised data using an RNNLM.

Table 5.7 shows WER on *how2 dev* set using both the *How2 4-gram LM* and *How2 RNNLM*. Comparing row 3 and row 4 in the table, we see that using RNNLM rescoring gives a slighly better performance (0.2% absolute) than using just the n-gram LM for generating lattices. However, the difference is very small compared to the improvement over the baseline system. We point out that this RNNLM rescoring gives a significant WER improvement on the *how2 dev* set (18.7 to 16.8) for the seed system (Tedlium baseline system). While the improved seed system implies better lattices and numerator supervision which can help semi-supervised training, here we see from the results that a 2% improvement in WER of the seed system through LM rescoring only amounts to a 0.2% improvement in semi-supervised training results.

Given the computational cost of RNNLM rescoring and the tiny gains that we see from it compared to the improvement over the baseline, we do not

recommend for most applications RNNLM rescoring of unsupervised data lattices for semi-supervised transfer training.

5.6.3.4 Effect of multitask training and amount of unsupervised data

In this section, we compare *multitask training* with two other strategies – *train from scratch* and *init from baseline* – described in Section 5.6.3 for different amounts of unsupervised data.

Figure 5.5 shows WER on *Tedlium dev* and *Tedlium test* when using different amounts of unsupervised data for adaptation from AMI-IHM to Tedlium corpus. Multitask learning is generally seen to outperform *train from scratch* and *init from baseline*. Here, we see that WER generally improves upon increasing the amount of unsupervised data from Tedlium corpus when using in-domain *Ted LM* for decoding as described in Section 5.6.3.1. However, when the out-domain *AMI LM* is used for decoding, there is not always improvement on increasing the amount of unsupervised data.

For adaptation from Fisher English to Fearless steps corpus using 180 hours of unsupervised data, the results in Table 5.8 show a 2.2% improvement (38.8 $\rightarrow$ 36.6) using the *train from scratch* method. However, if we use *init from baseline* i.e. initialize the network from the seed Aspire baseline network and update the model, we get a further 2.5% absolute improvement. Similarly, we get an overall 4.6% improvement (38.8 $\rightarrow$ 34.2) when using multitask training including 300 hours of augmented Fisher English data. However, increasing the amount of unsupervised data to 2400 hours gives no additional improvement with *multitask training* or *init from baseline*, but we do get some

improvement when *training from scratch* on only unsupervised data. With such a large amount of unsupervised data for training, it is likely that initialization from the Aspire baseline network is not necessary [13].

Based on the results in this section, we conclude that, as in the case of T-S learning (Section 5.3), semi-supervised multitask learning with some amount of out-of-domain supervised data is very useful, and is more robust than training only on unsupervised data.

5.6.3.5 Overall impact of semi-supervised transfer learning

In this section, we discuss the overall impact of our proposed semi-supervised transfer learning methods for domain adaptation when there is no parallel data in source and target domains. We rely on the our experimental results in Tables 5.6, 5.7 and 5.8.

As can be seen in Table 5.6, semi-supervised transfer learning with multitask training on supervised source-domain data and unsupervised target-domain data recovers 64% of the WER improvement over the baseline (trained on source-domain) that we get using an oracle system. The recovery rate is smaller when using a mismatched *AMI LM* to decode the unsupervised target-domain data from Tedlium corpus.

We see that this approach also scales well to large unsupervised data based on the results in Table 5.7. Comparing the semi-supervised multitask training results with the Supervised How2 system results, we see that using 2200

[13] It should be noted that with 2400 hours of data, we trained a much larger neural network model with 5 LSTM layers instead of just 4 as in the Aspire baseline, which might have contributing to the better results using multitask training.

hours of unsupervised target-domain How2 data (along with 452 hours of supervised source-domain Tedlium data), we can get a WER close to that obtained using 300 hours of supervised target-domain How2 data. This shows the utility of unsupervised target-domain data compared to the supervised target-domain data, which is much harder to obtain. Furthermore, this is a significant 2.5% absolute improvement (1.6% after RNNLM rescoring) over the Tedlium baseline trained on only the out-of-target-domain Tedlium.

As can be seen in Table 5.8, for adaptation to Fearless steps corpus, we get a 10% relative improvement over an out-of-domain Aspire baseline (38.8 to 33.9). But the improvement is likely limited because of the lack of a good in-domain LM matched to the Fearless steps corpus. Further, experimentation is need here with better or improved LM to see if we can benefit from the large amount of unsupervised data.

Chapter 6

Conclusion and future work

6.1 Conclusions

In this thesis, we proposed various approaches for semi-supervised sequence training of acoustic models for automatic speech recognition. In the first part, we focused on the conventional lattice-based sequence discriminative training framework. We proposed to minimize the lattice entropy, which is analogous to the MMI objective in the semi-supervised training scenario. The lattice allows to incorporate uncertainties in the hypothesis, without requiring a confidence-based filtering of utterances. While the standard objectives like sMBR degraded the performance when applied to unsupervised data using just a single best transcript, the proposed approach still gave an improvement. The proposed approach gave a WER recovery rate of 25% combined over Fisher English test sets relative to the WER improvement obtained using sMBR objective with the oracle transcription.

In the next part, we focused on extending the lattice-free MMI objective to the semi-supervised training scenario. We explored various methods of

creating the supervision for computing the numerator of LF-MMI objective. We introduced a "smart" splitting approach which correctly splits the lattice into chunks while accounting for the appropriate initial and final scores at the beginning and ends of chunks. We found that it is important for the supervision to have alternative phone sequences for each word sequence to cover the different pronunciation variations of the words as well as the position and durations of optional silences. This gave a consistent improvement in performance on all our setups. We investigated the effect of lattice beam, scale on word LM scores and applying tolerance on position and duration of phones in the supervision. A beam of 4.0 with an LM scale of 0.5 on both word LM and phone LM scores and a tolerance of $\pm 30ms$ on phone timings in the supervision was found to be optimal for setups with small amount of supervised data (15 or 50 hours). But for larger setups with 100 hours of supervised data, a beam of 2.0 with an LM scale of 0.0 on word LM was the most beneficial.

Using our proposed semi-supervised training approach with lattice-based supervision, we were able to recover up to 65% of the WER improvement possible using oracle transcriptions when using only 15 hours of supervised data from Fisher English. The WER Recovery Rate (WRR) remained consistent in the 40-60% range even on large-scale setups with 100 hours of supervised data and up to 1600 hours of unsupervised data. Lattice-based supervision was 5-10% absolute better in WRR than 1-best transcription as supervision. However, since the WERs start saturating upon increasing the amount of data, increasing to a large amount of unsupervised data showed tiny improvements

in WER. We also showed that a strong LM is important for generating lattices used as supervision. Using a strong LM trained on text corpus besides the supervised data transcripts gave a 5-10% absolutely better WRR than using an LM trained only on the supervised data transcripts. However, we did not find a significant improvement with lattice rescoring with higher order n-gram or RNNLM, and the higher computational cost does not merit the use of rescoring. Thus, we recommend using a large text corpus to train the LM for decoding unsupervised data. We finally showed that our proposed method also works equally well in multiple Babel languages, where we got a WRR of around 40-50%.

Finally, we looked at the scenario where the unsupervised data is from a different domain. Here, we first investigated the scenario where there is parallel data available in source and target domains. In such a case, we showed that a teacher-student (T-S) learning approach is very effective. Here, we generated numerator supervision using the teacher network trained on the source domain and used it to train a student network on the target domain. We showed experiments on 3 different scenarios – adaptation from clean to noisy speech, adaptation from 8kHz Fisher to 16kHz AMI-IHM where there is mismatch in bandwidth, microphone and language domain, and adaptation from headset microphone to distance microphone condition on both the ICSI and Mixer 6 corpora.

We showed that T-S learning with multitask training on both supervised and unsupervised data is significantly better than training only on unsupervised data. This was the case even when there was a big mismatch between

the data as in the case of Fisher vs AMI. We showed that sequence-KL divergence objective is more robust than LF-MMI and prevents overtraining that is seen with LF-MMI when training the student network only on unsupervised data. However, when multitask training was used, the difference was less significant. Since multitask training was very effective in handling domain mismatch, we recommend using LF-MMI or an interpolation of LF-MMI and sequence-KL with a higher weight on LF-MMI. Overall, we showed a good WER recovery rate of 50% or more for adaptation from clean to noisy and close-talk to distant microphone. But our proposed T-S learning approach was not effective by itself in dealing with the adaptation from 8kHz Fisher to 16kHz AMI, because parallel data was not available to handle the different kinds of mismatch other than bandwidth mismatch.

In general cases, where parallel data is not available in source and target domains, we showed that the semi-supervised LF-MMI with multitask training on supervised and unsupervised data can effectively handle the domain mismatch. Multitask training is more robust than training only on unsupervised data even when initializing from a seed network. By using a language model trained on the target domain text, we showed that we can effectively deal with language domain mismatch between source and target data. Using a matched language model improved WRR from 46% to 64% for adaptation from AMI-IHM to Tedlium corpus. A matched language model also showed a higher benefit from adding more unsupervised data to training, while a mismatched language model did not always show WER improvement on

adding more unsupervised data. Thus, we recommend using a large external LM corpus matched to the target-domain to train an LM for decoding unsupervised data.

When the language domain of the unsupervised data is different, we also found that it was better to use different denominator FSTs for computing the supervised and unsupervised data denominator likelihoods. For this, we estimated a phone LM using the best path phone sequence hypotheses of the unsupervised data utterances. However, using separate output layers degraded the performance, and we conclude that it is better to share all the layers of network when doing semi-supervised transfer learning with multitask training.

We also showed results for the applicability of these methods on large scale unsupervised corpora with natural and realistic speech – the How2 challenge corpus and the Fearless steps challenge corpus. For adaptation from Tedlium to How2 corpus, we were able to achieve as much a performance gain as with 300 hours of in-domain supervised data with just 2200 hours of in-domain completely unsupervised data. This shows that the proposed approach can greatly reduce the cost and time expensive manual transcription process. For adaptation to Fearless steps corpus, we were able to get 10% relative improvement over an out-of-domain Aspire baseline, but the improvement is likely limited because of the lack of a good in-domain LM matched to the Fearless steps corpus. Further, experimentation is needed here with better or improved LM.

6.2 Future work

6.2.1 Sequence-training objectives beyond MMI

It is generally known that MMI is the simplest sequence-training objective, this can be improved upon by explicitly incorporating a measure of sequence-level accuracy into the objective such as in sMBR [34] and boosted MMI [35], [100]. So, we can expect better improvement by replacing MMI-type objectives with a more sMBR-like or boosted-MMI objective even for semi-supervised training. This is done by extending the standard sMBR or boosted-MMI objectives to compute the expected accuracy w.r.t. a lattice-based supervision instead of a 1-best path supervision. The sMBR-version is as in (6.1), where π is a HMM state sequence and the summation is over all state sequences. In the conventional sequence discriminative training, this summation is over a lattice referred to as the denominator lattice. In the lattice-free framework, this summation is over a full HMM that is represented as a denominator FST, $\mathcal{G}_{\text{Den}}$, as described in Section 2.6.2 and the lattice $\mathcal{L}^{(r)}$ is converted into a numerator graph $\mathcal{G}_{\text{Num}}(\mathcal{L}^{(r)})$ as described in Section 2.6.3. In this case, we can call the objective as "lattice-free sMBR" and this is as shown in (6.2).

$$\mathcal{F}_{\text{EsMBR}}(\lambda) \triangleq \sum_r \sum_\pi \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda) A(\mathcal{L}^{(r)}, \pi), \tag{6.1}$$

$$= \sum_r \sum_{\pi \in \mathcal{G}_{\text{Den}}} \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda) A(\mathcal{G}_{\text{Num}}(\mathcal{L}^{(r)}), \pi), \tag{6.2}$$

We can compute the objective and gradients for (6.1) by defining $A(\mathcal{L}^{(r)}, \pi)$

appropriately:

$$A(\mathcal{L}^{(r)}, \pi) = \sum_t \gamma_t(\rho(a_t)), \tag{6.3}$$

where $\gamma_t(l(a_t))$ is the lattice posterior probability (or numerator posterior probability in lattice-free sMBR) at time t of senone $\rho(a_t)$ in lattice $\mathcal{L}^{(r)}$, and a_t is the HMM state at time i in the HMM state sequence π. If the lattice $\mathcal{L}^{(r)}$ has only one path, then $\gamma_t(\rho(a_t))$ falls back to an indicator function of the senone $\rho(a_t)$ at time t and is same as the conventional sMBR.

We conducted preliminary experiments using this objective on subsets of the Fisher English corpus. While using this objective by itself degraded the performance, using a combination of lattice-free sMBR objective with lattice-free MMI for the unsupervised part of the semi-supervised training was slightly better than using lattice-free MMI objective. But the difference was not very significant. We also did not see a similar improvement when we tried this or lattice-free boosted MMI (LF-bMMI) – for the supervised training. However, in [101], the authors were able to get about 5% relative improvement using LF-bMMI. As future work, we could investigate further into using these objectives for semi-supervised training.

6.2.2 Lightly supervised training

Chapters 3 and 4 dealt with methods for semi-supervised training when there is a large amount of unsupervised audio available. In some cases, in addition to the audio, there is also some form of light supervision available. For example,

1. In the broadcast news scenario, there might be reading scripts available,

which might not completely match the audio because of improvisations by the reader.

2. In TV shows, there might closed captions available, which might not align with audio segments correctly.

3. In YouTube videos, there might be caption words, which do not form a full transcript of the audio, but are only a few keywords.

4. In standard ASR corpora, there might be human transcription errors.

The imprecise transcriptions in all these cases are called "light supervision" and the scenario is called "lightly supervised training". In this scenario, we might be able to do better than standard semi-supervised training by making use of the available light supervision.

In most standard lightly-supervised training approaches, a biased LM is used to decode the initial audio segments to get best-path hypothesis and these are aligned with the original transcripts to retrieve the aligned portions for training. The lattice-free MMI framework in Chapter 4 allows us to use lattice-based supervision instead of supervision from just a single path. As future work, we could investigate using this framework to create better supervision using the biased LM in the lightly-supervised training scenario.

Bibliography

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, ACM, 2006, pp. 369–376.

[2] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI," in *Proc. INTERSPEECH*, 2016, pp. 2751–2755.

[3] G. Pundak and T. N. Sainath, "Lower frame rate neural network acoustic models.," in *Proc. INTERSPEECH*, 2016, pp. 22–26.

[4] G. Zavaliagkos, M. Siu, T. Colthurst, and J. Billa, "Using untranscribed training data to improve performance," in *Proc. ICSLP*, ISCA, 1998.

[5] K. Vesely, M. Hannemann, and L. Burget, "Semi-supervised training of Deep Neural Networks," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, IEEE, 2013, pp. 267–272.

[6] F. Grezl and M. Karafiat, "Semi-supervised bootstrapping approach for neural network feature extractor training," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, 2013, pp. 470–475. DOI: 10.1109/ASRU.2013.6707775.

[7] P. Zhang, Y. Liu, and T. Hain, "Semi-supervised dnn training in meeting recognition," in *Proceedings of*, Sheffield, 2014.

[8] L. Mathias, G. Yegnanarayanan, and J. Fritsch, "Discriminative training of acoustic models applied to domains with unreliable transcripts.," in *ICASSP (1)*, 2005, pp. 109–112.

[9] K. Yu, M. Gales, L. Wang, and P. C. Woodland, "Unsupervised training and directed manual transcription for LVCSR," *Speech Communication*, vol. 52, no. 7, pp. 652–663, 2010.

[10] X. Cui, J. Huang, and J.-T. Chien, "Multi-view and multi-objective semi-supervised learning for large vocabulary continuous speech recognition," in *Proc. ICASSP*, 2011, pp. 4668–4671. DOI: 10.1109/ICASSP.2011.5947396.

[11] L. Lamel, J.-L. Gauvain, and G. Adda, "Lightly supervised and unsupervised acoustic model training," *Computer Speech & Language*, vol. 16, no. 1, pp. 115–129, 2002.

[12] H. Y. Chan and P. C. Woodland, "Improving broadcast news transcription by lightly supervised discriminative training," in *IN PROC. IEEE INT. CONF. ACOUST., SPEECH, SIGNAL PROCESS*, 2004.

[13] S.-H. Liu, F.-H. Chu, S.-H. Lin, and B. Chen, "Investigating data selection for minimum phone error training of acoustic models," in *Multimedia and Expo, 2007 IEEE International Conference on*, 2007, pp. 348–351. DOI: 10.1109/ICME.2007.4284658.

[14] D. Povey, "Discriminative Training for Large Voculabulary Speech Recognition," PhD thesis, Cambridge University, 2004.

[15] L. Bahl, P. Brown, P. de Souza, and R. Mercer, "Maximum Mutual Information Estimation of Hidden Markov Model parameters for Speech Recognition," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, vol. 11, 1986, pp. 49–52. DOI: 10.1109/ICASSP.1986.1169179.

[16] J.-T. Huang and M. Hasegawa-Johnson, "Semi-supervised training of gaussian mixture models by conditional entropy minimization," *Optimization*, vol. 4, p. 5, 2010.

[17] V. Manohar, D. Povey, and S. Khudanpur, "Semi-supervised maximum mutual information training of deep neural network acoustic models.," in *Proc. INTERSPEECH*, 2015, pp. 2630–2634.

[18] V. Manohar, H. Hadian, D. Povey, and S. Khudanpur, "Semisupervised training of acoustic models using lattice-free mmi," in *ICASSP*, 2018.

[19] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[20] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: A survey," *Knowledge-Based Systems*, vol. 80, pp. 14–23, 2015.

[21] Y. Bengio *et al.*, "Deep learning of representations for unsupervised and transfer learning.," *ICML Unsupervised and Transfer Learning*, vol. 27, pp. 17–36, 2012.

[22] D. Wang and T. F. Zheng, "Transfer learning for speech and language processing," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific*, IEEE, 2015, pp. 1225–1237.

[23] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, IEEE, 2013, pp. 7893–7897.

[24] J. Wong and M. Gales, "Sequence student-teacher training of deep neural networks," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 8, 2016, pp. 2761–2765.

[25] N. Kanda, Y. Fujita, and K. Nagamatsu, "Investigation of lattice-free maximum mutual information-based acoustic models with sequence-level kullback-leibler divergence," in *Proc. ASRU*, 2017.

[26] P. Ghahrehmani, V. Manohar, D. Povey, and S. Khudanpur, "Investigation of Transfer Learning for LF-MMI Trained Neural Networks for ASR," in *Automatic Speech Recognition and Understanding (ASRU), 2017 IEEE Workshop on*, 2017.

[27] V. Manohar, P. Ghahremani, D. Povey, and S. Khudanpur, "A teacher-student learning approach for unsupervised domain adaptation of sequence-trained asr models," in *SLT*, 2018.

[28] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[29] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[30] H. A. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Norwell, MA, USA: Kluwer Academic Publishers, 1993, ISBN: 0792393961.

[31] B.-H. Juang, W. Hou, and C.-H. Lee, "Minimum Classification Error rate methods for Speech Recognition," *Speech and Audio Processing, IEEE Transactions on*, vol. 5, no. 3, pp. 257–265, 1997.

[32] D. Povey. and P. C. Woodland, "Minimum Phone Error and I-smoothing for Improved Discriminative Training," in *ICASSP*, 2002.

[33] J. Kaiser, B. Horvat, and Z. Kacic, "A novel loss function for the overall risk criterion based discriminative training of HMM models," in *Sixth International Conference on Spoken Language Processing*, 2000.

[34] M. Gibson and T. Hain, "Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition.," in *INTERSPEECH*, Citeseer, 2006.

[35] D. Povey, D. Kanevsky, *et al.*, "Boosted MMI for Feature and Model Space Discriminative Training," in *ICASSP*, 2008.

[36] V. Valtchev, J. Odell, P. Woodland, and S. Young, "Lattice-based discriminative training for large vocabulary speech recognition," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, vol. 2, 1996, 605–608 vol. 2. DOI: 10.1109/ICASSP.1996.543193.

[37] V Valtchev, J. Odell, P. C. Woodland, and S. J. Young, "MMIE training of large vocabulary recognition systems," *Speech Communication*, vol. 22, no. 4, pp. 303–314, 1997.

[38] D. Povey, "Discriminative Training for Large Voculabulary Speech Recognition," PhD thesis, Cambridge University, 2004.

[39] P. C. Woodland and D. Povey, "Large scale discriminative training of Hidden Markov Models for Speech Recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 25–47, 2002.

[40] X. He, L. Deng, and W. Chou, "Discriminative learning in sequential pattern recognition," *Signal Processing Magazine, IEEE*, vol. 25, no. 5, pp. 14–36, 2008.

[41] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks.," in *INTERSPEECH*, 2013, pp. 2345–2349.

[42] D. Yu and L. Deng, "Chapter 8: Deep Neural Network Sequence-Discriminative Training," in *Automatic Speech Recognition — A Deep Learning Approach*, Automatic Speech Recognition — A Deep Learning Approach. Springer, 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/chapter-8-deep-neural-network-sequence-discriminative-training/.

[43] A. Senior, H. Sak, F. de Chaumont Quitry, T. Sainath, and K. Rao, "Acoustic modelling with cd-ctc-smbr lstm rnns," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 604–609. DOI: 10.1109/ASRU.2015.7404851.

[44] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds., Cambridge, MA: MIT Press, 2005, pp. 529–536.

[45] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiát, S. Kombrink, P. Motlicek, Y. Qian, K. Riedhammer, K. Veselỳ, and T. N. Vu, "Generating Exact Lattices in The WFST Framework," in *Proceedings of 2012 IEEE International Conference on Acoustics, Speech and Signal Processing*, Kyoto, JP: IEEE Signal Processing Society, 2012, pp. 4213–4216, ISBN: 978-1-4673-0044-5.

[46] Z. Li and J. Eisner, "First-and second-order expectation semirings with applications to minimum-risk training on translation forests," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, Association for Computational Linguistics, 2009, pp. 40–51.

[47] J. T. Huang, "Semi-supervised learning for acoustic and prosodic modeling in speech applications," PhD thesis, 2012.

[48] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, 1988.

[49] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 8619–8623. DOI: 10.1109/ICASSP.2013.6639348.

[50] J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 7304–7308. DOI: 10.1109/ICASSP.2013.6639081.

[51] C. Cieri, D. Miller, and K. Walker, "The fisher corpus: A resource for the next generations of speech-to-text.," in *LREC*, vol. 4, 2004, pp. 69–71.

[52] M. Harper, *IARPA Babel Program*, 2014.

[53] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving Deep Neural Network Acoustic Models using Generalized Maxout Networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 2014, pp. 215–219. DOI: 10.1109/ICASSP.2014.6853589.

[54] P. S. Rath, D. Povey, K. Veselỳ, and J. Cernockỳ, "Improved Feature Processing for Deep Neural Networks," in *Proceedings of Interspeech 2013*, Lyon, FR: International Speech Communication Association, 2013, pp. 109–113, ISBN: 978-1-62993-443-3.

[55] P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, "A Pitch Extraction Algorithm tuned for Automatic Speech Recognition," in *ICASSP*, 2014.

[56] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of deep neural networks with natural gradient and parameter averaging," *arXiv preprint arXiv:1410.7455*, 2014.

[57] H. Hadian, H. Sameti, D. Povey, and S. Khudanpur, "Towards discriminatively-trained hmm-based end-to-end models for automatic speech recognition," in *Submitted to ICASSP*, 2018.

[58] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[59] J. Ma and R. Schwartz, "Unsupervised versus supervised training of acoustic models," in *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[60] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[61] N. Dehak, P. J. 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, 2011.

[62] V. Manohar, D. Povey, and S. Khudanpur, "JHU Kaldi System for Arabic MGB-3 ASR Challenge using Diarization, Audio-Transcript alignment and Transfer learning," in *Proc. ASRU 2017*, 2017.

[63] H. Xu, D. Povey, L. Mangu, and J. Zhu, "Minimum bayes risk decoding and system combination based on a recursion for edit distance," *Computer Speech & Language*, vol. 25, no. 4, pp. 802–828, 2011.

[64] D. Povey, *Pocolm*, https://github.com/danpovey/pocolm, 2016.

[65] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *2015 International Conference on Acoustics, Speech, and Signal Processing*, 2015, pp. 5206–5210.

[66] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2006, pp. 535–541.

[67] J. Ba and R. Caruana, "Do deep nets really need to be deep?" In *Advances in neural information processing systems*, 2014, pp. 2654–2662.

[68] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[69] J. Li, M. L. Seltzer, X. Wang, R. Zhao, and Y. Gong, "Large-scale domain adaptation via teacher-student learning," *arXiv preprint arXiv:1708.05466*, 2017.

[70] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, IEEE, 2017, pp. 5220–5224.

[71] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition.," in *INTERSPEECH*, 2015, pp. 3586–3589.

[72] V. Peddinti, V. Manohar, Y. Wang, D. Povey, and S. Khudanpur, "Far-field asr without parallel data.," in *INTERSPEECH*, 2016, pp. 1996–2000.

[73] Y. Long, Y. Li, H. Ye, and H. Mao, "Domain adaptation of lattice-free mmi based tdnn models for speech recognition," *International Journal of Speech Technology*, vol. 20, no. 1, pp. 171–178, 2017.

[74] N. Kanda, Y. Fujita, and K. Nagamatsu, "Sequence distillation for purely sequence trained acoustic models," in *Proc. ICASSP 2018*, 2018.

[75] M. Harper, "The automatic speech recogition in reverberant environments (aspire) challenge," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, IEEE, 2015, pp. 547–554.

[76] I. McCowan, J. Carletta, W Kraaij, S Ashby, S Bourban, M Flynn, M Guillemot, T Hain, J Kadlec, V Karaiskos, *et al.*, "The ami meeting corpus," in *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, vol. 88, 2005, p. 100.

[77] L Brandschain, D Graff, C Cieri, K Walker, C Caruso, and A Neely, "The mixer 6 corpus: Resources for crosschannel and text independent speaker recognition," in *Proc. of LREC*, 2010.

[78] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, *et al.*, "The icsi meeting corpus," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, IEEE, vol. 1, 2003, pp. I–I.

[79] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. INTERSPEECH*, 2015, pp. 3214–3218.

[80] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[81] G. Cheng, V. Peddinti, D. Povey, V. Manohar, S. Khudanpur, and Y. Yan, "An exploration of dropout with LSTMs," in *Proc. INTERSPEECH*, 2017.

[82] M. Karafiat, L. Burget, P. Matejka, O. Glembek, and J. Cernocky, "iVector-based discriminative adaptation for automatic speech recognition," in *Proc. Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, IEEE, 2011, pp. 152–157, ISBN: 978-1-4673-0367-5. DOI: 10.1109/ASRU.2011.6163922.

[83] V. Peddinti, G. Chen, D. Povey, and S. Khudanpur, "Reverberation robust acoustic modeling using i-vectors with time delay neural networks," in *Proc. INTERSPEECH*, 2015.

[84] Rousseau, Anthony and Deléglise, Paul and Esteve, Yannick, "TED-LIUM: an Automatic Speech Recognition dedicated corpus.," in *LREC*, 2012, pp. 125–129.

[85] Hernandez, François and Nguyen, Vincent and Ghannay, Sahar and Tomashenko, Natalia and Estève, Yannick, "TED-LIUM 3: twice as much data and corpus repartition for experiments on speaker adaptation," in *International Conference on Speech and Computer*, Springer, 2018, pp. 198–208.

[86] R. Sanabria, O. Caglayan, S. Palaskar, D. Elliott, L. Barrault, L. Specia, and F. Metze, "How2: a large-scale dataset for multimodal language understanding," in *Proceedings of the Workshop on Visually Grounded Interaction and Language (ViGIL)*, NeurIPS, 2018. [Online]. Available: http://arxiv.org/abs/1811.00347.

[87] J. H. Hansen, A. Sangwan, L. Kaushik, and C. Yu, "Fearless steps: Advancing speech and language processing for naturalistic audio streams from earth to the moon with apollo," *The Journal of the Acoustical Society of America*, vol. 143, no. 3, pp. 1868–1868, 2018.

[88] J. H. Hansen, A. Sangwan, A. Joglekar, A. E. Bulut, L. Kaushik, and C. Yu, "Fearless steps: Apollo-11 corpus advancements for speech technologies from earth to the moon.," in *Interspeech*, 2018, pp. 2758–2762.

[89] L. Kaushik, A. Sangwan, and J. H. Hansen, "Multi-channel apollo mission speech transcripts calibration.," in *INTERSPEECH*, 2017, pp. 2799–2803.

[90] NASA, *Apollo Mission Reports*, https://www.hq.nasa.gov/alsj/alsjmrs.html.

[91] NASA, *Apollo Flight Journal*, https://history.nasa.gov/afj/.

[92] NASA, *Apollo Lunar Surface Journal*, https://history.nasa.gov/alsj/.

[93] A. Rousseau, P. Deléglise, and Y. Esteve, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *LREC*, 2014, pp. 3935–3939.

[94] H. Xu, K. Li, Y. Wang, J. Wang, S. Kang, X. Chen, D. Povey, and S. Khudanpur, "Neural network language modeling with letter-based features and importance sampling," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2018, pp. 6109–6113.

[95] H. Xu, T. Chen, D. Gao, Y. Wang, K. Li, N. Goel, Y. Carmiel, D. Povey, and S. Khudanpur, "A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 5929–5933.

[96] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *1995 International Conference on Acoustics, Speech, and Signal Processing*, IEEE, vol. 1, 1995, pp. 181–184.

[97] CMU, *Carnegie Mellon Pronouncing Dictionary*, http://www.speech.cs.cmu.edu/cgi-bin/cmudict, 1998.

[98] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech communication*, vol. 50, no. 5, pp. 434–451, 2008.

[99] J. R. Novak, N. Minematsu, and K. Hirose, "WFST-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding," in *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, 2012, pp. 45–49.

[100] G. Heigold, T. Deselaers, R. Schlüter, and H. Ney, "Modified mmi/mpe: A direct evaluation of the margin in speech recognition," in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 384–391.

[101] C. Weng and D. Yu, "A comparison of lattice-free discriminative training criteria for purely sequence-trained neural network acoustic models," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6430–6434. DOI: 10.1109/ICASSP.2019.8683664.

Vita

Vimal Manohar received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology Madras, Chennai, India in 2013 following thesis work with Dr. Umesh S. He joined the Ph.D. program in the Department of Electrical Engineering at the Johns Hopkins University, Baltimore in August 2013. He worked with Dr. Sanjeev Khudanpur and Dr. Daniel Povey at the Center for Language and Speech Processing in the general area of speech processing, focusing on semi-supervised training and unsupervised adaptation of acoustic models for automatic speech recognition. He received the ECE Graduate Fellowship in 2013, Hamburger Fellowship in 2013 and Alexa Graduate Fellowship in 2018. His work was nominated for the best student paper award at Interspeech 2015, and he won the best student paper award at ICASSP 2016. He is an active developer of the open-source Kaldi speech recognition toolkit.

In September 2019, Vimal started as a Research Scientist at Facebook in New York, New York.