# SEMI-SUPERVISED TRAINING OF ACOUSTIC MODELS USING LATTICE-FREE MMI

*Vimal Manohar*[1,2], *Hossein Hadian*[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1] Center for Language and Speech Processing
[2] Human Language Technology Center of Excellence
Johns Hopkins University, Baltimore, MD 21218, USA

## ABSTRACT

The lattice-free MMI objective (LF-MMI) has been used in supervised training of state-of-the-art neural network acoustic models for automatic speech recognition (ASR). With large amounts of unsupervised data available, extending this approach to the semi-supervised scenario is of significance. Finite-state transducer (FST) based supervision used with LF-MMI provides a natural way to incorporate uncertainties when dealing with unsupervised data. In this paper, we describe various extensions to standard LF-MMI training to allow the use as supervision of lattices obtained via decoding of unsupervised data. The lattices are rescored with a strong LM. We investigate different methods for splitting the lattices and incorporating frame tolerances into the supervision FST. We report results on different subsets of Fisher English, where we achieve WER recovery of 59-64% using lattice supervision, which is significantly better than using just the best path transcription.

***Index Terms*—** Semi-supervised training, Lattice-free MMI, Sequence training, Automatic speech recognition

## 1. INTRODUCTION

Deep neural network (DNN) based acoustic models currently form the standard for automatic speech recognition (ASR) systems. Of late, sequence-level objectives like Connectionist Temporal Classification (CTC) [1] and Lattice-free Maximum Mutual Information (LF-MMI) [2] are preferred over traditional frame-level objectives like cross-entropy, especially for sequence tasks. Training neural networks from random initialization using these is effective even when using an additional pass of sequence discriminative training using state Minimum Bayes Risk (sMBR) [3] as shown in [2]. However, these methods are known to be data hungry and are more effective with large datasets with $>$100 hours of data. The performance is shown to degrade with smaller datasets [4].

While it is difficult to obtain large amounts of supervised data, the amount of unsupervised audio available is larger by several orders of magnitude. Unsupervised audio has been used successfully in various semi-supervised training approaches that predate even the modern DNN acoustic models [5]. The most common approach to semi-supervised learning is self-training. The general recipe here is to use a system trained on supervised data to generate transcripts for unsupervised data and select the transcripts based on various confidence score based filtering schemes. Confidence scores can be at the frame level [6], word level [7] or utterance level [6, 8, 9].

Discriminative training is sensitive to the accuracy of the transcripts [10, 11, 12]. Frame-level filtering has been used with conventional sequence discriminative training in [10, 13]. In [14], the lattice-free MMI objective is used with utterance-level confidence filtering and post-processing of 1-best hypothesis. However, unlike that work, here we make use of a whole lattice as supervision and show that it is significantly better. Lattice supervision has been used in [15] with conventional sequence objectives like MMI [16] as well as in [17, 18] for semi-supervised training with lattice entropy minimization. Unlike any of the previous works, here we use lattice-based supervision in lattice-free MMI framework along with the standard tricks such as a 3-fold frame subsampling inside the neural network, and incorporate language model (LM) costs into LF-MMI supervision.

This paper is organized as follows. Section 2 gives an overview of the lattice-free MMI training. Section 3 describes our different methods of creating supervision for training on unsupervised data. Section 4 describes our experiments and their results. Section 5 presents our conclusions and sets up future work.

## 2. LF-MMI TRAINING

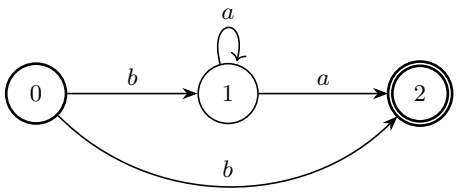
For an overall description of the LF-MMI training, the reader is directed to [2]. In this section, we give an overview of the training that is relevant to the modifications we do for semi-supervised training.

The standard method of numerator graph creation as described in [2] is applicable when there is a single transcript (word sequence) for an utterance. In [2], a GMM system is used to dump lattices of alternative pronunciations of the manually transcribed word sequences. In this work's unsupervised scenario, we can use a seed LF-MMI system to dump lattices of alternate pronunciations of the best path transcripts obtained from decoding unsupervised utterances. Using the same method as in [2], we process these lattices into phone graphs that are compiled into utterance-specific FSTs by composing the graphs with a context-dependency transducer (C) [19, 20] and transducer representing the HMM topology (H). Figure 1 shows the phone topology we normally use for LF-MMI experiments, which allows one symbol "b" followed by zero or more copies of "a".

The utterance-specific FSTs are then composed with a frame-by-frame mask of what phones are allowed on specific frames, derived from the times of phones in the lattices extended with a tolerance. In [2], a tolerance of 40ms corresponding to $\pm 2$ frames (10ms per frame) is used to allow the phones to appear within a 50ms window around where it appeared in the lattice. For the unsupervised audio

This work was partially supported by NSF Grant No CRI-1513128 and IARPA Contract No 2012-12050800010. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, IARPA, DoD/ARL or the U.S. Government.

**Fig. 1**. Phone topology used in LF-MMI




here, since a *3x* subsampled LF-MMI system is used to generate the lattices the frame shift is 30ms. Here, we use a tolerance of $\pm 1$ subsampled frame, which corresponds to a tolerance of 60ms.

The numerator FSTs created above are unweighted. For efficient minibatch training on GPUs, these are split into chunks of around 150 frames and weights are added from the denominator FST. For details of this process of adding costs to the numerator FST, the reader is directed to [2].

## 3. NUMERATOR GRAPH CREATION

In this section, we describe various methods for creating the numerator graphs for unsupervised utterances. We assume we have a seed neural network model trained with LF-MMI on only the supervised data. This is used to decode the unsupervised audio to get lattices.

### 3.1. Best path word sequence

When we use the best path word sequence as the transcript for an utterance from the unsupervised dataset, the approach is as described previously in Section 2.

### 3.2. Lattices: Naïve splitting

In this approach, we convert a lattice of word sequences obtained from the decoding of an utterance into phone graphs that are compiled into utterance-specific FSTs. We call this "naïve" approach because when we split the supervision, we do not add appropriate costs to the beginning and end of the splits. The lattice is optionally rescored with a strong 4-gram LM or RNN-LM, and is further pruned with a beam *beam*. A *beam* of 0 is like using the best path word sequence, but without the alternate pronunciations for the words. We refer to this case as "best path phone sequence".

When creating the phone graphs, the graph costs from the lattice are added with an LM-scale *lm-scale* (e.g. 0.5). Note that this is different from Section 3.1, where we did not add graph costs from the lattice (equivalent to *lm-scale* $= 0$) and only added costs from the denominator FST after splitting the numerator FST into chunks. These phone graphs are compiled into utterance-specific FSTs the same way as in Section 2 with the difference being that the phone graphs are from multiple word sequences and include LM costs. Also, in order to avoid double counting of LM costs (from the lattice and denominator FST), we add the costs from the denominator FST with a scale of $1 -$ *lm-scale*.

When we split the numerator FSTs, we do not add any costs to the beginning and end of the splits. So all the paths in the split have the same initial and final costs (of 0). This is not correct when using LM costs from the lattice (i.e. *lm-scale* $> 0$), especially when using a large *beam*, and this motivates the approaches described below.

### 3.3. Lattices: No splitting

For efficiency of the neural network training, we need to create minibatches of fixed-size chunks. In the previous sections, we split all utterances into chunks of 150 frames. In this section, we use an alternate approach where we use the full utterance without splitting. However, for training efficiency we ensure that all the utterances in the unsupervised dataset are modified to be of around 20 distinct lengths. We slightly alter the speed of the audio so that each utterance's length becomes the nearest of the distinct lengths. Alternatively, we can pad each utterance with silence to get one of the distinct lengths. This approach of modifying lengths is used in the work [21] for end-to-end LF-MMI training. When using full unsplit utterances, we use the same approach as before for creating the numerator FSTs, but now we do not need to worry about the costs at the beginning and end of the splits.

### 3.4. Lattices: Smart splitting

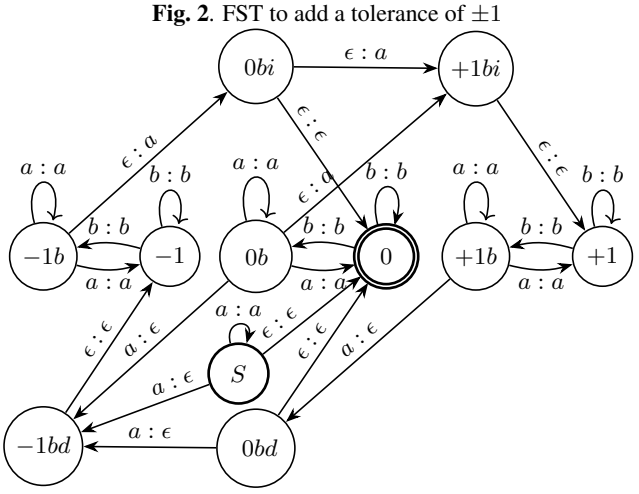
In this section we describe a smarter approach to create numerator FSTs, where we split the lattices directly to get fixed-length chunks (150 frames of 10ms), while also adding appropriate initial and final costs. This is unlike in Sections 3.1 and 3.2, where we convert the lattice into utterance-specific FSTs, add tolerances and then split the FSTs. For this, we first dump the lattices (with *transition-ids* as input labels and words as output labels [22]) retaining the correct acoustic costs on each arc. Then we run a forward-backward [23] on the lattice to compute the alpha and beta scores for each state. Then we split the lattice into fixed-length chunks. For each initial state of this chunk, we add as the initial cost the alpha cost for that state. For each final state of this chunk, we add as the final cost the beta cost for that state. This ensures that the forward-backward cost for this chunk and the frame posteriors are exactly the same as for the original lattice before splitting. We project this to the input labels to get a finite state acceptor (FSA) with *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 following section.

#### *3.4.1. Adding tolerances*

While we solve the issue of initial and final costs, 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. 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 $\pm 1$ frame and a transition model from Figure 1 with only one phone, this FST is as shown in Figure 2. Here $b$ is the forward *transition-id* and $a$ is the self-loop *transition-id* . 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 2 shows three *offset states* for $o \in \{-1, 0, 1\}$. The offset represents the number of extra self-loop *transition-id*s 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:

**Fig. 2**. FST to add a tolerance of $\pm 1$




First, we allow a deletion or insertion of self-loop *transition-id*s only at the end of the longest $ba^*$ sequence. e.g. the FST can a 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-id*s 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 *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 $\epsilon$ 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 $\epsilon$ 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*.

## 4. EXPERIMENTS

### 4.1. Experimental setup

We report results on the Fisher English corpus [24]. We set aside a randomly chosen subset of speakers (250 hours) from the corpus as unsupervised data. We use the transcripts from the remaining 1250 hours to train the LMs for decoding unsupervised data. We use different subsets of supervised data – 15 hours, 50 hours – to compare the effect of various proportions of supervised and unsupervised data. The results are reported on separately held-out dev and test sets (about 5 hours each), which are part of the standard Kaldi [20] recipe for Fisher English. We use WER Recovery Rate (WRR) [25] as a metric to evaluate the WER improvements from semi-supervised training:

$$\mathrm{WRR} = \frac{\mathrm{BaselineWER} - \mathrm{SemisupWER}}{\mathrm{BaselineWER} - \mathrm{OracleWER}}. \quad (1)$$

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) [26] system. The training details are in [2]. We use i-vector [27] 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. Also, for comparison purposes, we use statistics from only the supervised data to train the context-dependency decision tree.

The phone LM used for creating the denominator FST is estimated using phone sequences from both supervised and unsupervised data as in [28, 29]. We give a higher weight to the phone sequences from supervised data (2.5 for the 15 hours supervised dataset and 1.5 for the 50 hours one). We did not tune these factors.

#### *4.1.1. 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 [6]. The per-frame derivatives of our objective are scaled by these weights. The results are as shown in Table 1 for 15 hours supervised and 250 hours unsupervised data (with *lm-scale* 0.5, *beam* 4.0 and tolerance $\pm 1$ frame). Using best path posteriors as frame-weights is significantly better with both naïve and smart splitting of lattices. Therefore, all subsequent results are reported using the best path posteriors as frame-weights.

**Table 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 post | **22.37** | **22.14** | **22.02** | **21.89** |

#### *4.1.2. Effect of supervision type*

Table 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 the best path word sequence as supervision gives around 7% WER improvement over the baseline. Using only the best path phone sequence from the lattice i.e. a *beam* of 0.0 gives 6% improvement over the supervised baseline. This suggests that it is especially important for unsupervised data supervision to have all the alternate pronunciations [1]. Further experiments conducted after the submission of this paper show that alternate pronunciations help even when using lattice supervision. We will report these results in a future paper. However, for this paper, we report only lattice supervision results using determinized lattices i.e. for each word sequence in the lattice we retain a unique HMM state sequence and hence a unique phone sequence.

For lattice supervision results shown in Table 2, we use a *beam* of 4.0 and *lm-scale* of 0.5. Lattice supervision is clearly better than best path supervision, 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.

[1] This also includes alternate paths that vary only in the presence or absence of optional silence.

**Table 2**. WER(%) results using various supervision (15 hours supervised + 250 hours unsupervised data)

| Supervision | *lm-scale* | *beam* | tol | dev | test | WRR |
|---|---|---|---|---|---|---|
| Baseline | - | - | - | 29.41 | 29.22 | - |
| Best path words | 0.0 | 8.0 | 1 | 22.55 | 22.75 | 58% |
| Best path phones | 0.0 | 0.0 | 1 | 23.04 | 23.23 | 54% |
| Naïve split | 0.5 | 4.0 | 1 | 22.37 | 22.14 | 62% |
| No split + sil | 0.5 | 4.0 | 1 | 22.04 | 22.25 | 63% |
| No split + speed | 0.5 | 4.0 | 1 | 22.13 | 22.31 | 62% |
| Smart split | 0.5 | 4.0 | 1 | **22.02** | **21.89** | 64% |
| Oracle | - | - | - | 17.92 | 17.95 | - |

#### *4.1.3. Effect of tolerance*

Table 3 shows results with various tolerances with *lm-scale* 0.5 and *beam* 4.0. The results with naïve splitting using tolerance $\pm 1$ frame (60ms) and tolerance $\pm 2$ (120ms) frames are both significantly better ($\sim$1% absolute) than using a tolerance of 0. There is no significant difference in WER between using tolerance $\pm 1$ and $\pm 2$ and we henceforth use $\pm 1$ frame of tolerance.

**Table 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** |

#### *4.1.4. Effect of LM scale and beam*

Experiments with various LM scales and beams are shown in table 4. As a result of these experiments, we recommend an *lm-scale* of 0.5 and a *beam* of 4.0.

Our intuition is that *lm-scale* has the largest effect when using a wide beam because then there are more paths to weight. We see this reflected in the *beam* 8.0 results with naïve splitting. However, more experiments are needed in the future with other beams and *lm-scale* values to confirm this.

We also note that when using smart splitting of lattices, with an *lm-scale* of 0.0, we are not really adding any initial and final costs (they are scaled by 0.0). So using a higher *lm-scale* is more appropriate in this case.

**Table 4**. WER(%) results (15 hours supervised + 250 hours unsupervised) for various beam sizes and LM scales

| | | | naïve split | | Smart split | |
|---|---|---|---|---|---|---|
| *lm-scale* | *beam* | tol | dev | test | dev | test |
| 0.0 | 0.0 | 1 | 23.04 | 23.23 | 23.04 | 23.25 |
| 0.0 | 2.0 | 1 | 22.43 | 22.59 | 22.46 | 22.47 |
| 0.0 | 4.0 | 1 | 22.27 | 22.30 | 22.38 | 22.60 |
| 0.0 | 8.0 | 1 | 23.48 | 23.73 | | |
| 0.5 | 2.0 | 1 | 22.44 | 22.69 | 22.50 | 22.37 |
| 0.5 | 4.0 | 1 | **22.37** | **22.14** | **22.02** | **21.89** |
| 0.5 | 8.0 | 1 | 22.44 | 22.46 | 22.11 | 22.15 |

#### *4.1.5. Summary of alternatives*

Table 5 shows a more complete comparison of the main alternatives we are exploring (best path; lattices with naïve split; lattices with smart split), with both the 15-hours-supervised and the 50-hours-supervised setups, and with the recommended settings as mentioned in the previous sections. The lattice-based methods are convincingly better than using only the best path, which confirms our intuition that it makes sense to preserve the uncertainty in the transcript into the training phase. However it is less clear whether there is a robust difference between the naïve and smart splitting methods. More experiments will be performed in future to clarify this.

**Table 5**. WER(%) results with 250 hours unsupervised data – 15 hours supervised vs 50 hours supervised data. WRR is the WER recovery rate after averaging WERs from dev and test

| | 15 hours sup | | | 50 hours sup | | |
|---|---|---|---|---|---|---|
| System | dev | test | WRR | dev | test | WRR |
| Baseline | 29.41 | 29.22 | - | 22.63 | 21.97 | - |
| Best path phones | 23.04 | 23.23 | 54% | 20.00 | 19.82 | 52% |
| Naïve split | 22.37 | 22.14 | 62% | 19.54 | 19.52 | 60% |
| Smart split | 22.02 | 21.89 | 64% | 19.60 | 19.61 | 59% |
| Oracle | 17.92 | 17.95 | - | 17.55 | 17.92 | - |

## 5. CONCLUSIONS AND FUTURE WORK

In this work, we investigated using lattice-based supervision with lattice-free MMI objective for semi-supervised training. We explored various ways of creating supervision from lattices obtained from decoding of unsupervised data. We improved the supervision by incorporating LM costs and allowing phone tolerances. We proposed a new FST-based approach of adding phone tolerances directly into the lattices. This smart approach ensures correct costs in the supervision and is suitable for efficient chunk-based minibatch training on GPU. Overall, we get a WER recovery rate of 59-64% using our proposed method for creating LF-MMI supervision. Our proposed method is substantially better than using only the best path phones, which has a WER recovery rate of less than $<$55%.

Our proposed method of splitting lattices accounting for the correct costs allows us to effectively use strong LMs like RNNLMs, which will be investigated in the future. We will also look to do more experiments on large scale unsupervised datasets including on other languages.

## 6. REFERENCES


[1] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*. ACM, 2006, pp. 369–376.

[2] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur, "Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI," in *Proc. INTERSPEECH*, 2016, pp. 2751–2755.

[3] Janez Kaiser, Bogomir Horvat, and Zdravko 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.

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

[5] George Zavaliagkos, Manhung Siu, Thomas Colthurst, and Jayadev Billa, "Using untranscribed training data to improve performance," in *Proc. ICSLP*. 1998, ISCA.

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

[7] Samuel Thomas, Michael L Seltzer, Kenneth Church, and Hynek Hermansky, "Deep neural network features and semi-supervised training for low resource speech recognition," in *Proc. ICASSP*. IEEE, 2013, pp. 6704–6708.

[8] 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*, Dec 2013, pp. 470–475.

[9] Pengyuan Zhang, Yulan Liu, and Thomas Hain, "Semi-supervised dnn training in meeting recognition," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 141–146.

[10] Lambert Mathias, Girija Yegnanarayanan, and Juergen Fritsch, "Discriminative training of acoustic models applied to domains with unreliable transcripts.," in *Proc. ICASSP*, 2005, pp. 109–112.

[11] Kai Yu, Mark Gales, Lan Wang, and Philip C Woodland, "Unsupervised training and directed manual transcription for LVCSR," *Speech Communication*, vol. 52, no. 7, pp. 652–663, 2010.

[12] Xiaodong Cui, Jing Huang, and Jen-Tzung Chien, "Multi-view and multi-objective semi-supervised learning for large vocabulary continuous speech recognition," in *Proc. ICASSP*, May 2011, pp. 4668–4671.

[13] Shih-Hung Liu, Fang-Hui Chu, Shih-Hsiang Lin, and B. Chen, "Investigating data selection for minimum phone error training of acoustic models," in *Multimedia and Expo, 2007 IEEE International Conference on*, July 2007, pp. 348–351.

[14] Shane Walker, Morten Pedersen, Iroro Orife, and Jason Flaks, "Semi-supervised model training for unbounded conversational speech recognition," *arXiv preprint arXiv:1705.09724*, 2017.

[15] D. Povey, *Discriminative Training for Large Voculabulary Speech Recognition*, Ph.D. thesis, Cambridge University, 2004.

[16] L. Bahl, P. Brown, P.V. de Souza, and R. Mercer, "Maximum Mutual Information Estimation of Hidden Markov Model parameters for Speech Recognition," in *Proc. ICASSP*, Apr 1986, vol. 11, pp. 49–52.

[17] Jui-Ting Huang and Mark Hasegawa-Johnson, "Semi-supervised training of gaussian mixture models by conditional entropy minimization," *Optimization*, vol. 4, pp. 5, 2010.

[18] Vimal Manohar, Daniel Povey, and Sanjeev Khudanpur, "Semi-supervised maximum mutual information training of deep neural network acoustic models," in *Proc. INTERSPEECH*, 2015.

[19] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Speech recognition with weighted finite-state transducers," in *Springer Handbook of Speech Processing*, pp. 559–584. Springer, 2008.

[20] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, et al., "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011, number EPFL-CONF-192584.

[21] Hossein Hadian, Hossein Sameti, Daniel Povey, and Sanjeev Khudanpur, "Towards discriminatively-trained hmm-based end-to-end models for automatic speech recognition," in *Submitted to ICASSP*, 2018.

[22] Daniel Povey et al., "Generating exact lattices in the WFST framework," in *Proc. ICASSP*. 2012, pp. 4213–4216, IEEE Signal Processing Society.

[23] Lawrence 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.

[24] Christopher Cieri, David Miller, and Kevin Walker, "The fisher corpus: a resource for the next generations of speech-to-text.," in *LREC*, 2004, vol. 4, pp. 69–71.

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

[26] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin 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.

[27] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[28] Vimal Manohar, Daniel Povey, and Sanjeev Khudanpur, "JHU Kaldi System for Arabic MGB-3 ASR Challenge using Diarization, Audio-Transcript alignment and Transfer learning," in *Proc. ASRU 2017*, 2017.

[29] Pegah Ghahrehmani, Vimal Manohar, Daniel Povey, and Sanjeev 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.