Lattice-free MMI training

Peter Bell

Automatic Speech Recognition— ASR Lecture 12
28 February 2019

This lecture

- Motivating sequence training with the Maximum Mutual Information (MMI) criterion
- Fundamentals of MMI training
- Lattice-based MMI
- Purely sequence trained models
- Practical implementation of LF-MMI training

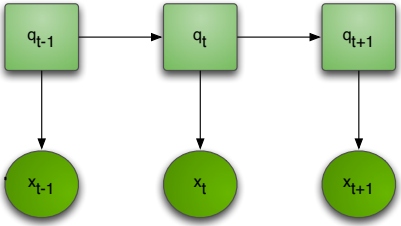
Some notation

λ	Parameters of the acoustic model
X	Sequence of acoustic observations
x_t	Observation at time t
Y	Sequence of words
q_t	HMM hidden state at time t
j	Index over HMM states
u	Index over utterances
$\gamma_j(t)$	Probability of being in state j at time t, given X

This may not exactly match notation in other lectures!

Traditional approach

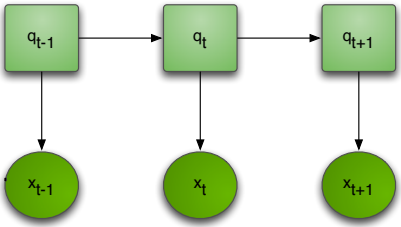
Use HMM-GMM as a generative sequence model



This is more for convenience than anything else

Traditional approach

Use HMM-GMM as a generative sequence model



This is more for convenience than anything else

Great algorithms for:

- finding $P(q_t|X)$, the probability of being in state q at time t (forward-backward)
- finding the most likely state sequence (Viterbi)

Problem - HMM assumptions don't hold in practice

- Observations are absolutely not conditionally independent, given the hidden state
- When states are phone-based, observations are not independent of past/future phone states, given the current state

Problem - HMM assumptions don't hold in practice

- Observations are absolutely not conditionally independent, given the hidden state
- When states are phone-based, observations are not independent of past/future phone states, given the current state

Two useful hacks:

- Expand the state space to incorporate phonetic context (makes the decoder more complicated, but we can cope...)
- Augment the feature vector to incorporate adjacent acoustic features using deltas, or feature splicing + linear transforms, attempting to keep individual elements uncorrelated

Problem - HMM assumptions don't hold in practice

- Observations are absolutely not conditionally independent, given the hidden state
- When states are phone-based, observations are not independent of past/future phone states, given the current state

Two useful hacks:

- Expand the state space to incorporate phonetic context (makes the decoder more complicated, but we can cope...)
- Augment the feature vector to incorporate adjacent acoustic features using deltas, or feature splicing + linear transforms, attempting to keep individual elements uncorrelated

Oops, this massively over-states the framewise probabilities, so throw in an acoustic scaling fudge factor, κ, of about $1/12$

A quote

"[O]ur knowledge about speech is at such a primitive stage that if we are not to be completely devastated by the problem of having too many free parameters then any model of an informative observation sequence will have to be based on some invalid assumptions. This led us to an investigation of an alternative to MLE, MMIE, which does not derive its raison d'etre from an implicit assumption of model correctness."

Peter Brown, 1987

Switch training criterion

- Maximum likelihood is theoretically optimal (even for classification), but only when the model is correct

Switch training criterion

- Maximum likelihood is theoretically optimal (even for classification), but only when the model is correct
- If not, an explicitly discriminative training criterion might be better

Switch training criterion

- Maximum likelihood is theoretically optimal (even for classification), but only when the model is correct
- If not, an explicitly discriminative training criterion might be better
- Minimum Classification Error (MCE) is a natural choice for classification, but not for sequences

Switch training criterion

- Maximum likelihood is theoretically optimal (even for classification), but only when the model is correct
- If not, an explicitly discriminative training criterion might be better
- Minimum Classification Error (MCE) is a natural choice for classification, but not for sequences
- Try to maximise mutual information instead

Discriminative training criteria

Maximum likelihood objective:

$$F_{ML}(\lambda) = \sum_u \log p_\lambda(X_u|W_u)$$

Discriminative training criteria

Maximum likelihood objective:

$$F_{ML}(\lambda) = \sum_u \log p_\lambda(X_u|W_u)$$

Maximum mutual information objective:

$$F_{\text{MMIE}}(\lambda) = \sum_u \log \frac{p(X_u, W_u)}{p(X_u)P(W_u)} = \sum_u \Big[\log \frac{p(X_u, W_u)}{p(X_u)} - \log P(W_u)\Big]$$
$$= \sum_u \Big[\log \frac{p_\lambda(X_u|W_u)^\kappa P(W_u)}{\sum_W p_\lambda(X_u|W)^\kappa P(W)} - \log P(W_u)\Big]$$

Discriminative training criteria

Maximum likelihood objective:

$$F_{ML}(\lambda) = \sum_u \log p_\lambda(X_u|W_u)$$

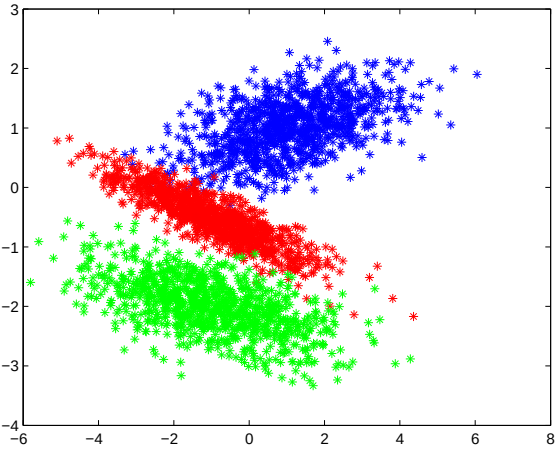
Maximum mutual information objective:

$$F_{\text{MMIE}}(\lambda) = \sum_u \log \frac{p(X_u, W_u)}{p(X_u)P(W_u)} = \sum_u \Big[\log \frac{p(X_u, W_u)}{p(X_u)} - \log P(W_u)\Big]$$
$$= \sum_u \Big[\log \frac{p_\lambda(X_u|W_u)^\kappa P(W_u)}{\sum_W p_\lambda(X_u|W)^\kappa P(W)} - \log P(W_u)\Big]$$

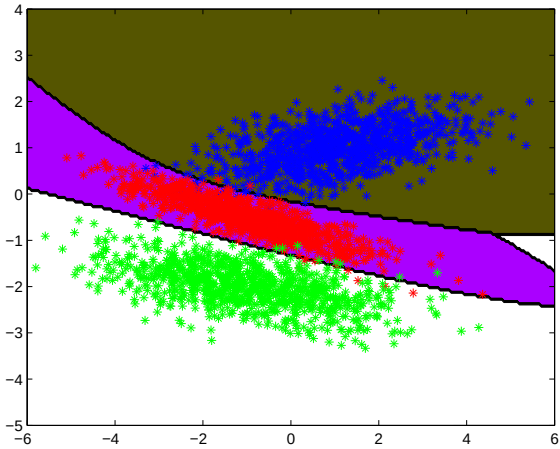
When $P(W)$ is constant, equivalent to conditional ML:

$$F_{\text{CML}}(\lambda) = \sum_u \log P(W_u|X_u) = \sum_u \log \frac{p_\lambda(X_u|W_u)^\kappa P(W_u)}{\sum_W p_\lambda(X_u|W)^\kappa P(W)}$$

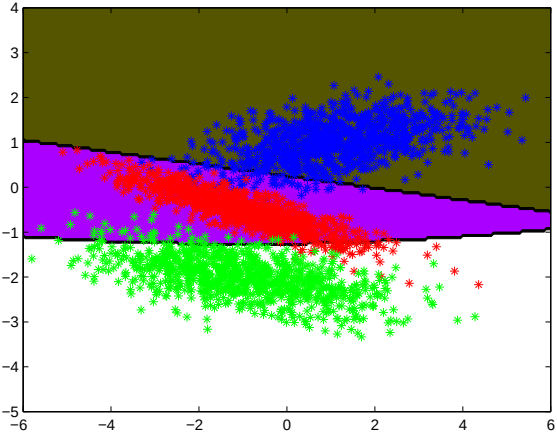
Example



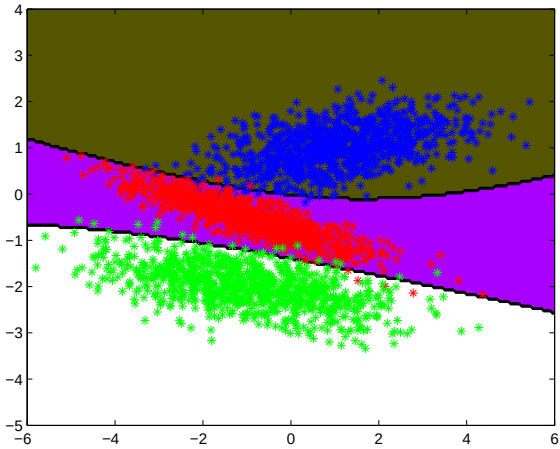
Full covariance Gaussian with ML training



Diagonal covariance Gaussian with ML training



Diagonal covariance Gaussian with MMI training



Challenges of MMI

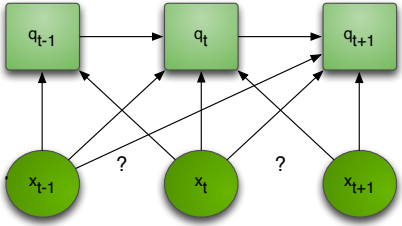
- Frame-level models are good, but sequence-level models are poor $\Rightarrow$ need to operate at the sequence level.
- It's hard to estimate the denominator probabilities over a complete sequence

$$\sum_{W} p_\lambda(X|W)P(W)$$

for anything beyond small tasks

Training with DNNs

- DNNs are better at incorporating wider context because they can more easily model correlated input features.
- We're still breaking the conditional independence assumption – a scaling factor still needed
- DNNs are normally trained discriminatively at the frame level using the cross-entropy criterion
- Fortunately we can apply sequence training with the MMI criterion in a very similar way to traditional HMM-GMM systems



The fundamentals of MMI training

Regardless of the method used, we need to compute two sets of occupancy probabilities:

numerator: $\gamma_j^{\text{num}}(t) = P(q_t = j | X_u, \mathcal{M}_u^{\text{num}})$

denominator: $\gamma_j^{\text{den}}(t) = P(q_t = j | X_u, \mathcal{M}_u^{\text{den}})$

$\mathcal{M}_u^{\text{num}}$ represents the HMM state sequence corresponding to the transcription of utterance u

$\mathcal{M}_u^{\text{den}}$ represents all possible HMM state sequences for u

The fundamentals of MMI training

Regardless of the method used, we need to compute two sets of occupancy probabilities:

numerator: $\gamma_j^{\text{num}}(t) = P(q_t = j | X_u, \mathcal{M}_u^{\text{num}})$

denominator: $\gamma_j^{\text{den}}(t) = P(q_t = j | X_u, \mathcal{M}_u^{\text{den}})$

$\mathcal{M}_u^{\text{num}}$ represents the HMM state sequence corresponding to the transcription of utterance u

$\mathcal{M}_u^{\text{den}}$ represents all possible HMM state sequences for u

Computing $\gamma_j^{\text{den}}(t)$ is hard

The fundamentals of MMI training

Regardless of the method used, we need to compute two sets of occupancy probabilities:

numerator: $\gamma_j^{\text{num}}(t) = P(q_t = j | X_u, \mathcal{M}_u^{\text{num}})$

denominator: $\gamma_j^{\text{den}}(t) = P(q_t = j | X_u, \mathcal{M}_u^{\text{den}})$

$\mathcal{M}_u^{\text{num}}$ represents the HMM state sequence corresponding to the transcription of utterance u

$\mathcal{M}_u^{\text{den}}$ represents all possible HMM state sequences for u

Computing $\gamma_j^{\text{den}}(t)$ is hard

Computing $\gamma_j^{\text{num}}(t)$ is the standard forward-backward algorithm. (But we need to make sure that the statistics are consistent with $\gamma_j^{\text{den}}(t)$)

Using the occupancy probabilities

GMM: accumulate 0th, 1st and 2nd order statistics for numerator and denominator

$$\Lambda_j^{(0)}(X) = \sum_t \gamma_j(t), \quad \Lambda_j^{(1)}(X) = \sum_t \gamma_j(t) x_t, \quad \Lambda_j^{(2)}(X) = \sum_t \gamma_j(t) x_t x_t^T$$

See Povey (2003) for full GMM update equations.

DNN:

$$\frac{\partial F_{\text{MMIE}}}{\partial \log p(x_t|j)} = \gamma_j^{\text{num}}(t) - \gamma_j^{\text{den}}(t)$$

$$\frac{\partial F_{\text{MMIE}}}{\partial \log a_t(s)} = \sum_j \frac{\partial F_{\text{MMIE}}}{\partial \log p(x_t|j)} \frac{\partial \log p(x_t|j)}{\partial a_t(s)}$$

$$= \sum_j (\gamma_j^{\text{num}}(t) - \gamma_j^{\text{den}}(t)) \frac{\partial \log p(x_t|j)}{\partial a_t(s)}$$

Recap: the forward backward algorithm

Forward probability $\alpha_j(t) = p(x_1, \ldots, x_t | q_t = j)$

$$\alpha_j(t) = \sum_i \alpha_i(t-1) a_{ij} p(x_t | q_t = j)$$

Backward probability $\beta_j(t) = p(x_{t+1}, \ldots, x_T | q_t = j)$

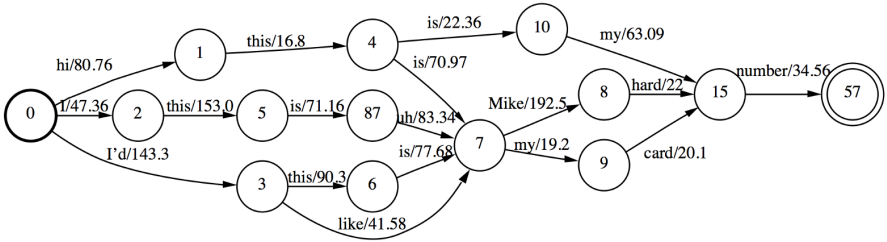
$$\beta_i(t) = \sum_j a_{ij} p(x_{t+1} | q_{t+1} = j) \beta_j(t+1)$$

Then compute state occupancy as

$$\gamma_j(t) = \frac{\alpha_j(t)\beta_j(t)}{\alpha_E(T)}$$

Lattice-based MMI

- Approximate $\sum_W$ with a sum over a lattice
- Generate lattice for each utterance using an initial model
- Use a weak language model
- But attempt to minimise the size of the lattice
- Derive phone arcs from the lattice



Lattice from www.cs.nyu.edu/~mohri/asr12/lecture_12.pdf

Forward-backward over lattices

Define forward and backward probabilities over phone arcs r with known start and end times

$$\alpha_r = \sum_{r' \to r} \alpha_{r'} a_{r'r} p(r)$$
$$\beta_r = \sum_{r \to r'} a_{rr'} p(r') \beta(r')$$
$$\gamma_r = \alpha_r \beta_r$$

Where $p(r)$ denotes the log likelihood over the arc
Use standard FB algorithm within arcs to compute state occupancies for time t

LF-MMI

"Purely sequence-trained models for ASR based on lattice-free MMI"

(Povey et al, 2016)

"Purely sequence-trained models for ASR based on lattice-free MMI"

(Povey et al, 2016)

- Solves a fundamental problem – a practical method for computing HMM "true" state posteriors using a DNN acoustic model
- Uses this to train a properly normalised sequence model, trained with MMI right from the start
- Removes the need for an acoustic scaling fudge factor

Getting it to work...

The core idea

- Both numerator and denominator state sequences are represented as *HCLG* FSTs
- Parallelise denominator forward-backward computation on a GPU
- Replace word-level LM with a 4-gram phone LM for efficiency
- Reduce the frame rate
  - might be a good idea for other reasons...
- Changes to HMM topology motivated by CTC (see Lecture 15)

Getting it to work...

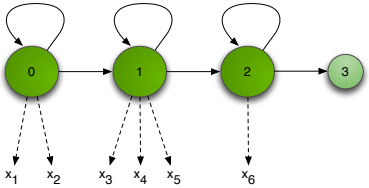
Extra tricks

- Train on small fixed-size chunks (1.5s)
  - probably enough to counter the flaws in the conditional independence assumption
- Careful optimisation of denominator FST to minimise the size
- Various types of regularisation

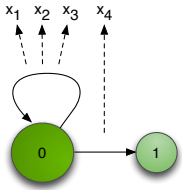
HMM topologies

Replace standard 3-state HMM with topology that can be traversed in a single frame

Standard topology



LF-MMI topology



Denominator FST

- LM is essentially a 3-gram phone LM
- No pruning and no backoff to minimise the size
  - Use of unpruned 3-grams means that there is always a 2-word history.
  - Minimises the size of the recognition graph when phonetic context is incorporated
- Addition of a fixed number of the most common 4-grams
- Conversion to *HCLG* FST in the normal way
- *HCLG* size reduced by a series of FST reversal, weight pushing and minimisation operations, followed by epsilon removal

The normalisation FST

- The phone-LM assumes that we are starting at the beginning of an utterance $\rightarrow$ not suitable for use with 1.5s chunks
- Need to adjust the initial probabilities for each HMM state
- Iterate 100 times through the denominator FST to get better initial occupancy probabilities

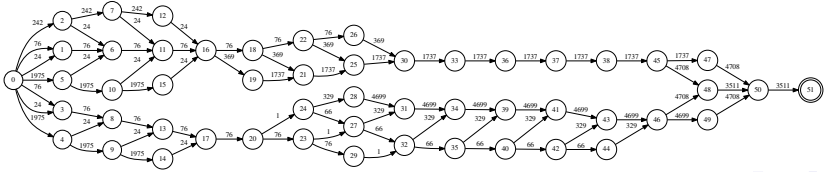
$$\alpha_j^{(n)}(0) = \sum_i a_{ij} \alpha_i^{(n-1)}(0)$$

- Add a new initial state to the denominator FST that connects to each state with the new probabilities $\rightarrow$ the "normalisation FST"

Numerator FST

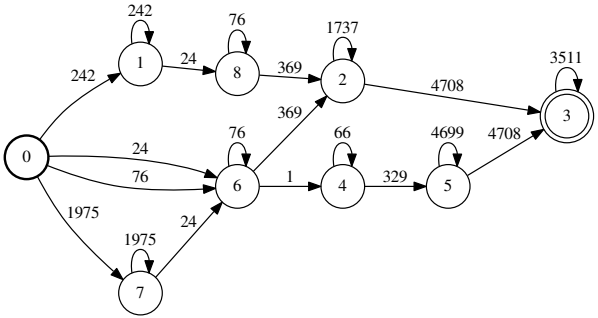
The original paper

- Used GMM system to generate lattices for training utterances, representing alternate pronunciations
- Lattice determines which phones are allowed to appear in which frames, with an additional tolerance factor
- Constraints encoded as an FST
- Compose with the normalisation FST to ensure that the logprob objective function is always < 0



Numerator FST

More recently, unconstrained numerator found to work better (Hadian, Povey et al, IEEE SLT, 2018)



Specialised forward-backward algorithm

- Work with probabilities rather than log-probabilities to avoid expensive log/exp operations
- Numeric overflow and underflow is a big problem
- Two specialisations:
  - re-normalise probabilities at every time step
  - the "leaky HMM" - gradual forgetting of context

Probability re-normalisation

Define

$$A(t) = \sum_i \alpha_i(t)$$

Forward/backward passes become

$$\alpha_j(t) = \sum_i \alpha_i(t-1) a_{ij} p(x_t | q_t = j) / A(t)$$

$$\beta_i(t) = \sum_j a_{ij} p(x_{t+1} | q_{t+1} = j) \beta_j(t+1) / A(t+1)$$

Add a correction factor to the total log probability:

$$\log p(X) = \log \alpha_E(T) + \sum_t \log A(t)$$

Leaky-HMM

The above is still susceptible to overflow in backward computation.
Introduce a leak-probability, η, of transition to any other state

Leaky-HMM

The above is still susceptible to overflow in backward computation.
Introduce a leak-probability, η, of transition to any other state

Probability of transition to state i given by

$$\eta\alpha_i(0)$$

(where these are the iterated occupancy probabilities)

Leaky-HMM

The above is still susceptible to overflow in backward computation.
Introduce a leak-probability, η, of transition to any other state

Probability of transition to state i given by

$$\eta\alpha_i(0)$$

(where these are the iterated occupancy probabilities)

Define $\hat{\alpha}_i(t) = \alpha_i(t) + \eta A(t)\alpha_i(0)$

Leaky-HMM

The above is still susceptible to overflow in backward computation.
Introduce a leak-probability, η, of transition to any other state

Probability of transition to state i given by

$$\eta\alpha_i(0)$$

(where these are the iterated occupancy probabilities)

Define $\hat{\alpha}_i(t) = \alpha_i(t) + \eta A(t)\alpha_i(0)$

Forwards pass:

$$\alpha_j(t) = \sum_i \hat{\alpha}_i(t-1)a_{ij}p(x_t|q_t = j)/A(t)$$

$$p(X) = \sum_i \hat{\alpha}_i(T)$$

Leaky-HMM

Define backwards variables:

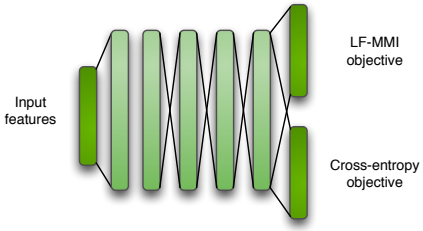
$$\hat{\beta}_i(T) = 1/p(X)$$
$$B(t) = \eta \sum_i \alpha_i(0)\hat{\beta}_i(t)$$
$$\beta_i(t) = \hat{\beta}_i(t) + B(t)$$

Backwards recursion:

$$\hat{\beta}_i(t) = \sum_j a_{ij} p(x_{t+1}|q_{t+1} = j)\beta_j(t+1)/A(t+1)$$
$$\gamma_j(t) = \sum_j a_{ij}\hat{\alpha}_i(t) p(x_t|q_t = j)\beta_j(t+1)/A(t)$$

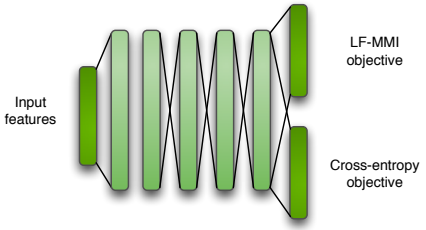
Regularisation

- Use standard Cross-Entropy objective as a secondary task
  - all but the final hidden layer shared between tasks
  - use numerator posteriors for convenience



Regularisation

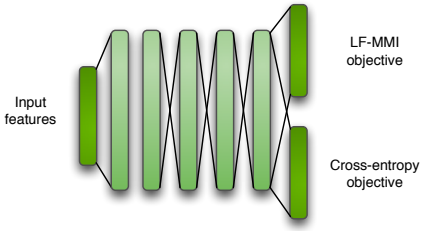
- Use standard Cross-Entropy objective as a secondary task
  - all but the final hidden layer shared between tasks
  - use numerator posteriors for convenience



- l_2 norm penalty on the main output

Regularisation

- Use standard Cross-Entropy objective as a secondary task
  - all but the final hidden layer shared between tasks
  - use numerator posteriors for convenience



- l_2 norm penalty on the main output
- Leaky HMM (mentioned earlier)

Benefits of LF-MMI

- Models are typically faster during training and decoding than standard models
- Word error rates are generally lower
- Ability to properly compute state posterior probabilities over arbitrary state sequences also opens possibilities for
  - Semi-supervised training
  - Cross-model student-teacher training

  where sequence information is critical

Benefits of LF-MMI

- Models are typically faster during training and decoding than standard models
- Word error rates are generally lower
- Ability to properly compute state posterior probabilities over arbitrary state sequences also opens possibilities for
  - Semi-supervised training
  - Cross-model student-teacher training

  where sequence information is critical

But – difficulties when training transcripts are unreliable

LF-MMI results on Switchboard

Results on SWB portion of the Hub 5 2000 test set, trained on 300h training set. Results use speed perturbation and i-vector based speaker adaptation.

Objective	Model (size)	WER (%)
CE	TDNN-A (16.6M)	12.5
CE $\rightarrow$ sMBR	TDNN-A (16.6M)	11.4
LF-MMI	TDNN-A (9.8M)	10.7
	TDNN-B (9.9M)	10.4
	TDNN-C (11.2M)	10.2
LF-MMI $\rightarrow$ sMBR	TDNN-C (11.2M)	10.0

See Povey et al (2016) for more results

Background reading

- Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahrmani, 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.
- Hossein Hadian, Hossein Sameti, Daniel Povey, and Sanjeev Khudanpur, "Flat-start single-stage discriminatively trained HMM-based models for ASR" in *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 2018
- Hossein Hadian, Daniel Povey, Hossein Sameti, Jan Trmal, and Sanjeev Khudanpur, "Improving LF-MMI using unconstrained supervisions for ASR" in *Proc. IEEE SLT*, 2018.
- Karel Veselý, Arnab Ghoshal, Lukáš Burget and Daniel Povey, "Sequence-discriminative training of deep neural networks", in *Proc. Interspeech*, 2013.
- Daniel Povey, "Discriminative Training for Large Vocabulary Speech Recognition". Ph.D. thesis, Cambridge University Engineering Department, 2003.
- Yves Normandin and Salvatore D. Morgera, "An improved MMIE training algorithm for speaker-independent, small vocabulary, continuous speech recognition" in *Proc. ICASSP*, 1991.