Carnegie Mellon.
SCHOOL OF COMPUTER SCIENCE

Implementing and Improving MMIE Training in SphinxTrain

Long Qin and Alex Rudnicky

Carnegie Mellon University

Language Technologies Institute

Motivation

- Realizes discriminative training in SphinxTrain
- Improves baseline MMIE performance
  - Gets better lattices by removing duplicate hypotheses
  - Reduces computation cost of statistics accumulation on lattices

Outline

- Introduction
- Baseline MMIE training
- MMIE training on pruned lattices
- Experiment
- Conclusion

Introduction

- SphinxTrain
  - Acoustic model trainer in Sphinx
  - Maximum Likelihood Estimation (MLE) is used
- Discriminative training
  - Forces the model to correctly recognize the training data
  - Builds an objective function to capture how well the model would recognize the training data
  - Starts from a MLE trained model and then optimize the chosen objective function

Outline

- Introduction
- ***Baseline MMIE training***
- MMIE training on pruned lattices
- Experiment
- Conclusion

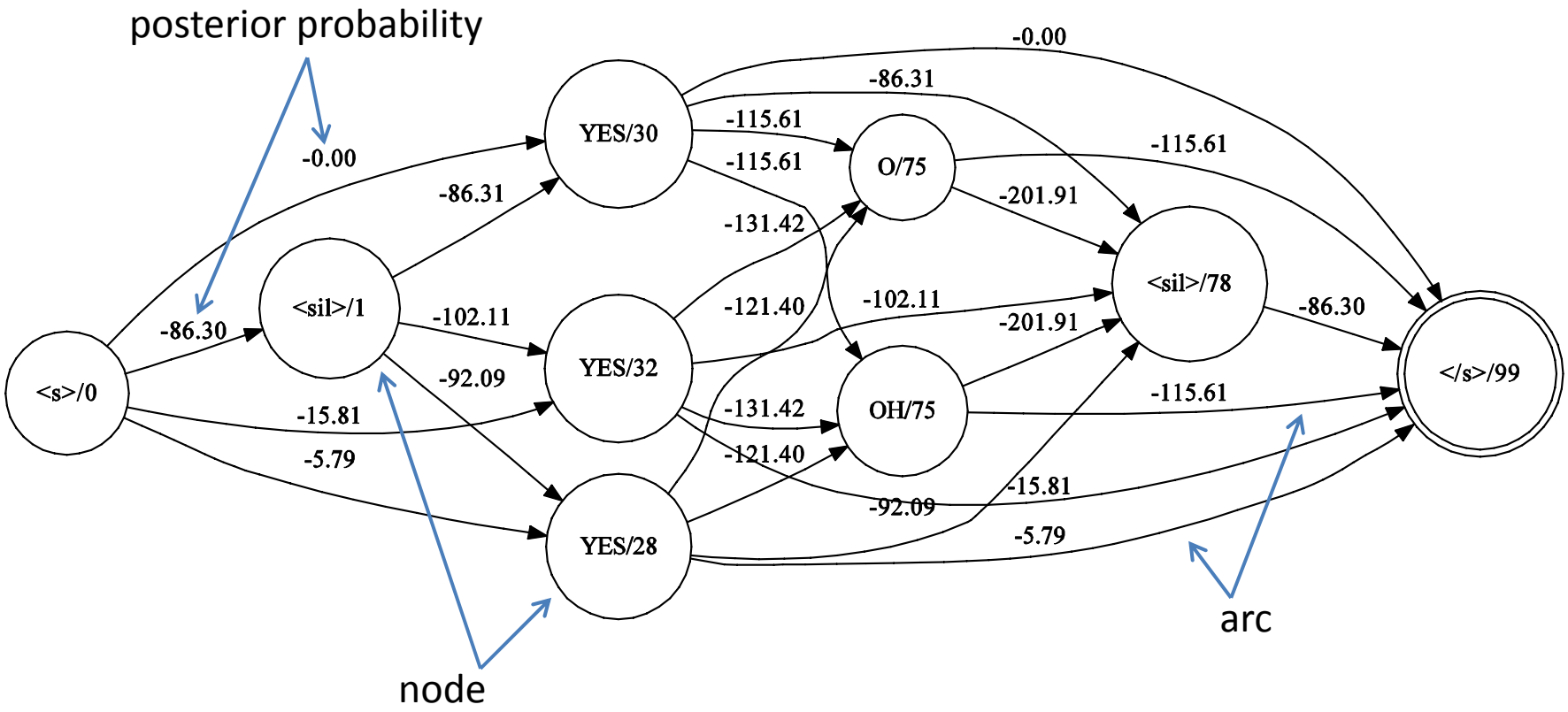
MMIE Training

- Maximum mutual information estimation (MMIE)
  - Objective function
    - Maximizes the posterior probability of the correct hypothesis

$$F(\lambda) = \sum_{r=1}^{R} \log \frac{P_\lambda(O_r \mid M_{s_r}) P(S_r)}{\sum_s P_\lambda(O_r \mid M_s) P(S)}$$

  - Involves a sum over all possible hypotheses
  - Approximated by the most likely hypotheses
    - N-best list
    - Word lattice

Word Lattice



Extended Baum-Welch (EBW)

- Forward-Backward on lattices to accumulate statistics
  - Uses Viterbi to estimate the acoustic score
  - Uses a unigram LM to compute the LM score

- Parameter update

$$\hat{\mu}_{j,m} = \frac{\{\theta^{num}_{j,m}(O) - \theta^{den}_{j,m}(O)\} + D_{j,m}\mu_{j,m}}{\{\gamma^{num}_{j,m} - \gamma^{den}_{j,m}\} + D_{j,m}}$$

$$\hat{\sigma}^2_{j,m} = \frac{\{\theta^{num}_{j,m}(O^2) - \theta^{den}_{j,m}(O^2)\} + D_{j,m}(\sigma^2_{j,m} + \mu^2_{j,m})}{\{\gamma^{num}_{j,m} - \gamma^{den}_{j,m}\} + D_{j,m}} - \hat{\mu}^2_{j,m}$$

Forward-Backward on Lattices

1. Initializes $\alpha(1) = \beta(Q) = 1.0$
2. for arc *q = 1…Q*

   for arc *r* preceding *q*

   $$\alpha(q) += \alpha(r) \cdot ac(q) \cdot lm(q)$$

3. for arc *q = 1…Q*

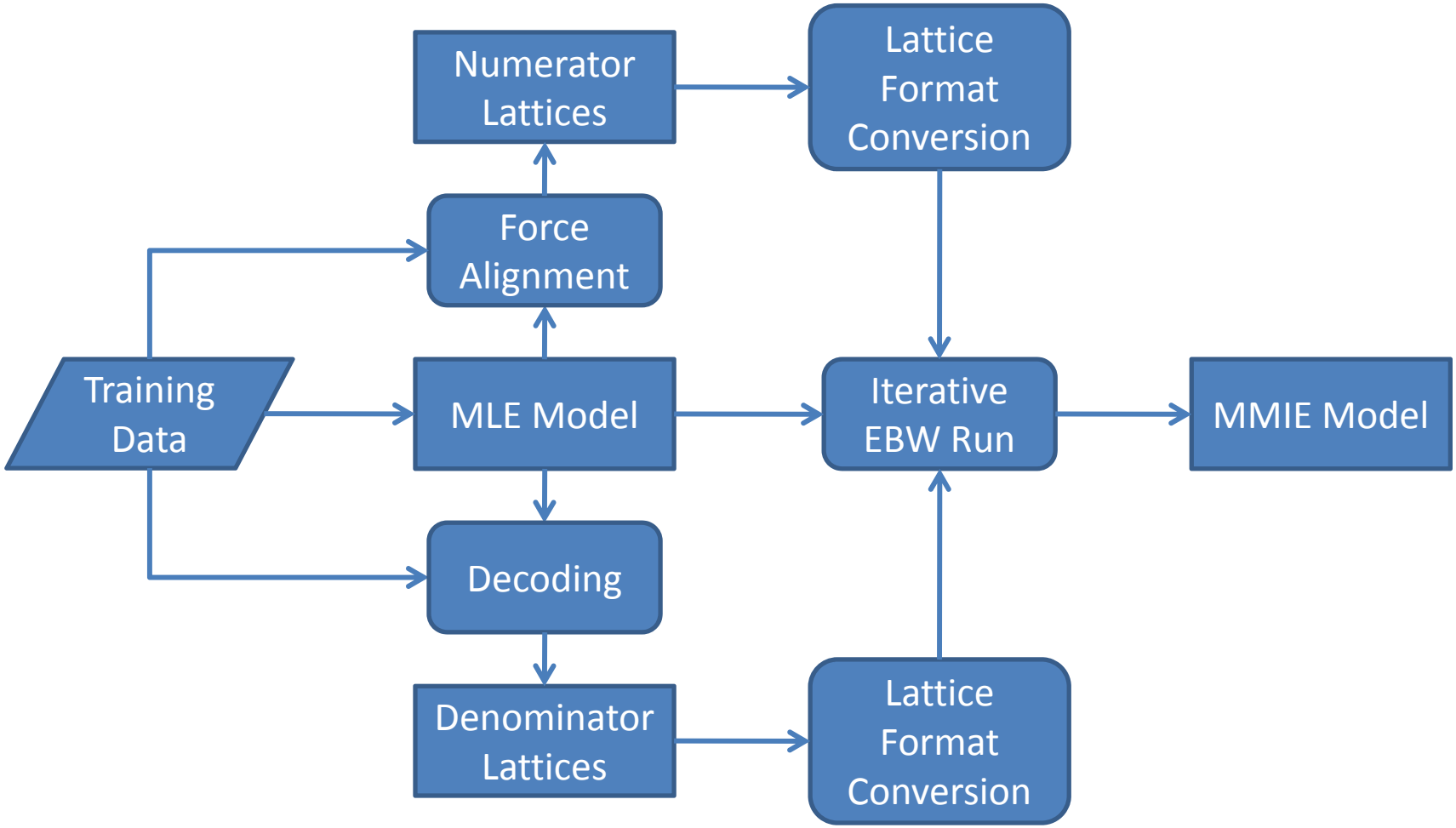
   for arc *r* following *q*

   $$\beta(q) += \beta(r) \cdot ac(q) \cdot lm(q)$$

4. for arc *q = 1…Q*

   $$\gamma(q) = \frac{\alpha(q)\beta(q)}{ac(q) \cdot lm(q) \cdot \beta(1)}$$

How does it work in Sphinx



Experiment Setup

- Training Data
  - WSJ-SI84
    - 15 hours speech, 2931 tied states, 8 Gaussian per mixture
  - WSJ-SI284
    - 81 hours speech, 5132 tied states, 16 Gaussian per mixture
- Testing Data
  - Nov. ‘92 5k-word task evaluation set
  - 5k-word trigram LM
- Lattices are generated only once
- No speaker adaptation

Baseline MMIE Results

- MLE vs. MMIE

		WSJ-SI84	WSJ-SI284
WER (%)	MLE	6.63	4.52
	MMIE	6.22	4.30
Imp		6.2%	4.9%

Outline

- Introduction
- MMIE training details
- Baseline MMIE results
- ***MMIE training on pruned lattices***
- Experiment results
- Conclusion

Sphinx Lattices

- A large number of word hypotheses
- Many duplicate hypotheses
  - Same word but different entry or exit time
  - Most of the time a model just competes with itself but not other confusable models
- Solution
  - Lattice pruning

Lattice Pruning Approaches

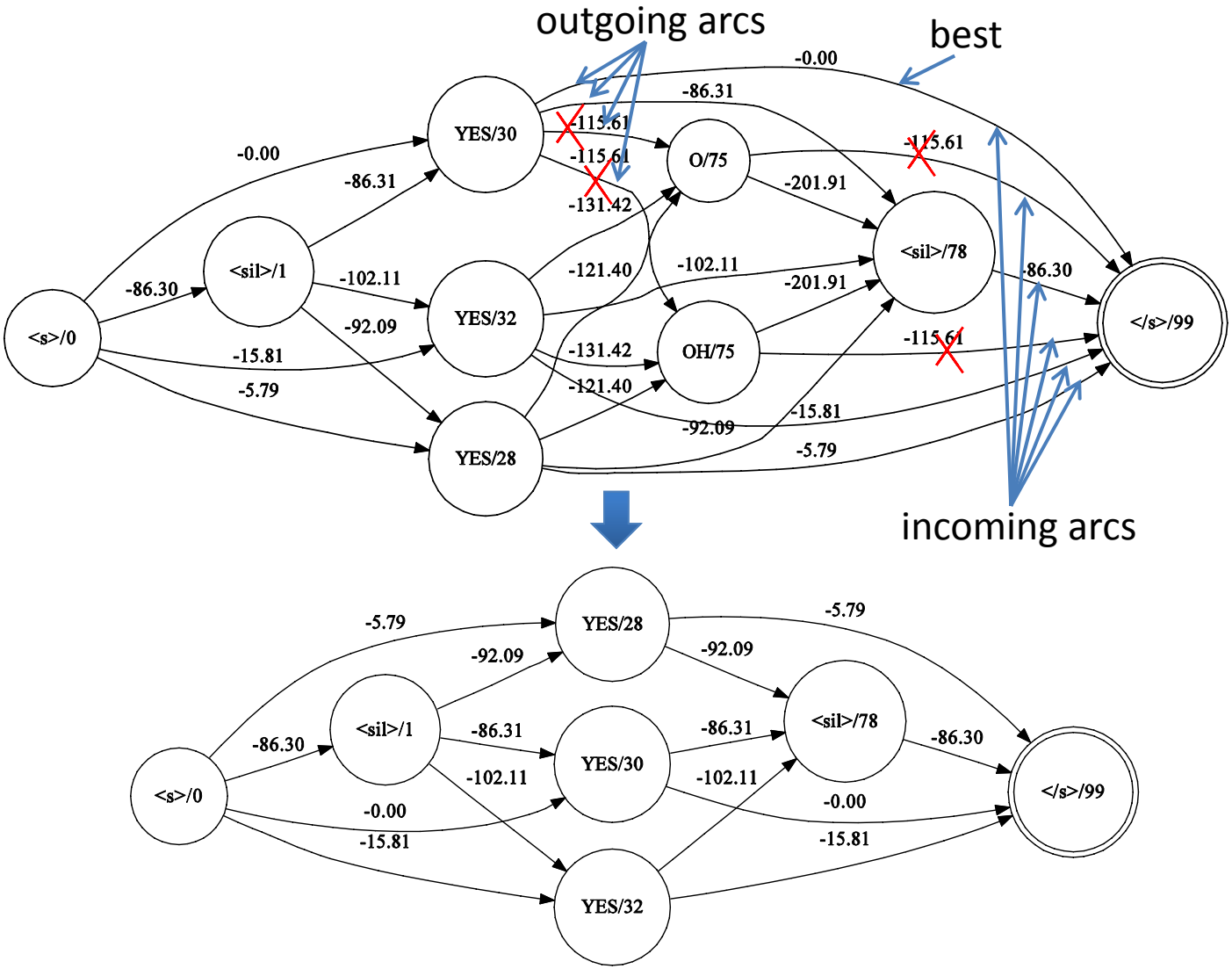
- Pruning during decoding
  - Beam pruning
- Pruning after lattice generation
  - Finite states automata determination and minimization
  - Confusion network

- Neither appropriate for MMIE training
  - Unigram LM probabilities are used in training
  - Time alignment needs to be kept for EBW computation
  - Gaussian occupation count is weighted by the posterior probability of a word hypothesis

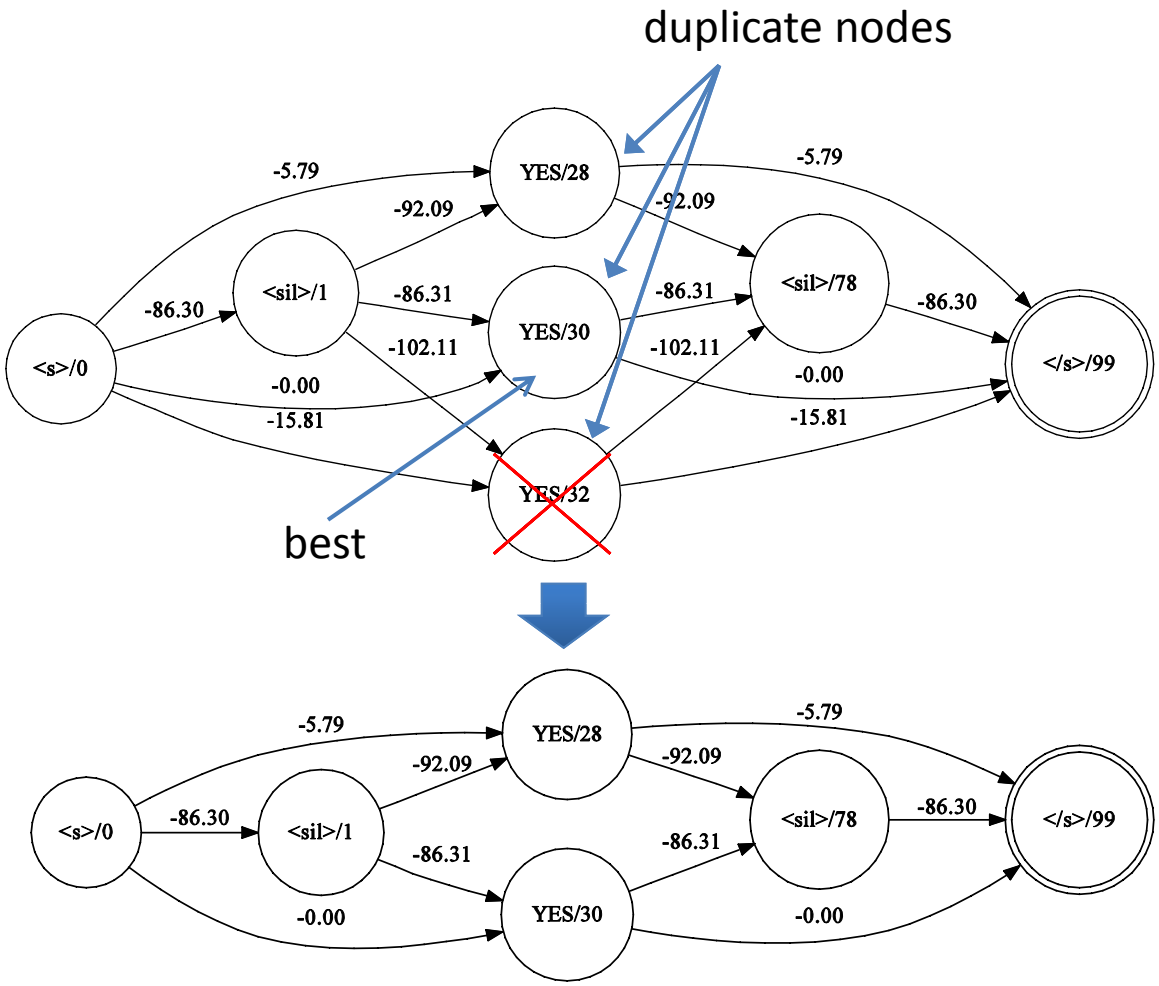
Posterior Probability Pruning

- Pruning hypotheses using posterior probability
  - Removing word hypotheses with low posterior probabilities doesn't affect accumulated statistics
  - Directly removes duplicate hypotheses
- Pruning performed on nodes and arcs
  - Node pruning
    - Searches for duplicate nodes in a 20-frame window
  - Arc pruning
    - Forward-backward arc pruning
  - Details provided later

Posterior Probability Arc Pruning



Posterior Probability Node Pruning



Experiment Setup

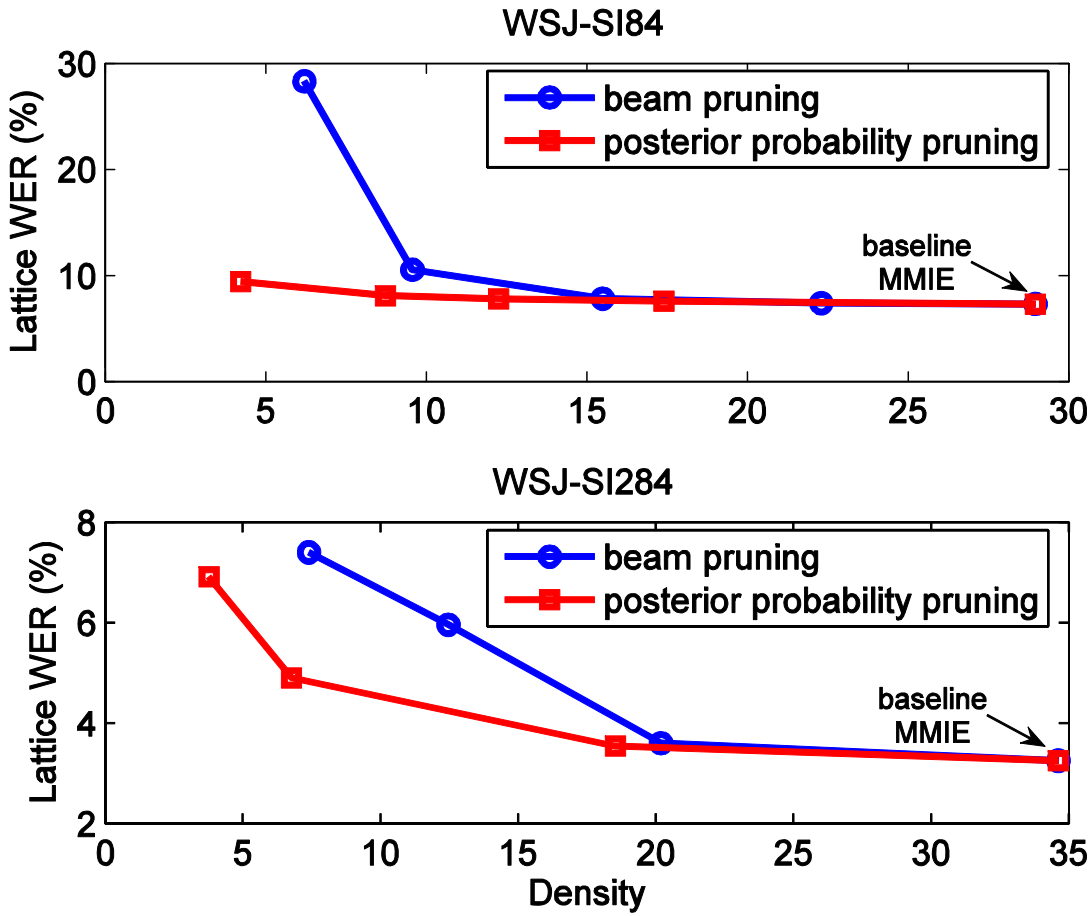
- Training Data
  - WSJ-SI84 and WSJ-SI284
- Testing Data
  - Nov. ‘92 5k-word task evaluation set

- Fixed beam width for node pruning and tried different beam widths for arc pruning
  - Node beam: 1E-10
  - Arc beam: 1E-70, 1E-60, 1E-50, etc
- Lattice pruning is only applied to denominator lattices before lattice format conversion

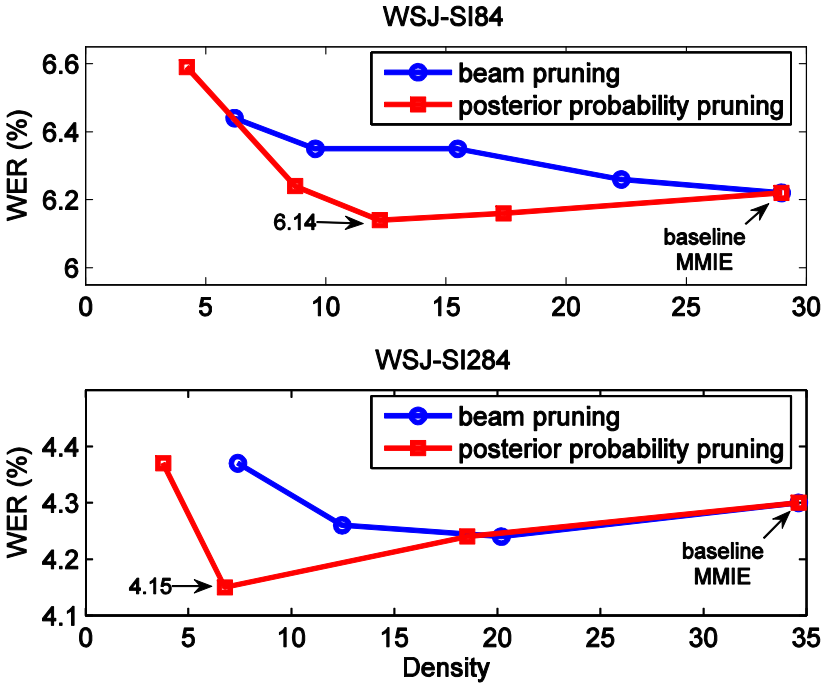
Evaluation Metrics

- Lattice Density
  - The number of nodes in the lattice divided by the actual number of words in the transcription
  - This is an estimate of lattice complexity
- Lattice WER
  - The best WER of any path in the lattice
  - A lower bound of WER by re-scoring the lattice
  - This is a rough measurement of lattice quality
- WER
  - The WER of recognition on testing data

Lattice Density vs. Lattice WER



Lattice Density vs. WER



		WSJ-SI84	WSJ-SI284
Imp (%)	Baseline	6.2	4.9
	Proposed	7.4	8.2

Computational Complexity Analysis

- Computation mainly comes from three steps
  - Lattice generation, lattice format conversion and iteratively EBW run
  - Computation is dominated by the first two steps
  - Computation from the latter two steps is proportional to the number of word hypotheses in the lattice
  - Computation cost for posterior probability pruning is small
- Saved about 40% to 60% of the baseline MMIE running time

Conclusion

- Posterior probability pruning is more appropriate than beam pruning for MMIE training
- Saves about 40% to 60% computation
- Gains greater improvement

- A new version of SphinxTrain with MMIE training to be released in April 2010.

Q & A

Thanks!