Progress in English Conversational Telephone Speech Transcription

Khe Chai Sim, Mark Gales, Xunying Liu, Phil Woodland & Kai Yu

March 2005

Cambridge University Engineering Department

Outline

- Increased number of model parameters
  - 9K states to 15K states
  - use of multiple STCs

- Combination results

- Initial experiments with fMPE
  - view fMPE as temporally varying *shift* of mean vectors
  - extension: pMPE as temporally varying *scale* of precision matrices

Acoustic Training Set-Up

- Acoustic Model Training Data (`fsh2004h5train03b` - 2180hours):
  - `h5train03b`: 360hours used in 2003 evaluation
  - `fsh2004`: 1820hours BBN/Wordwave+LDC quick transcriptions
- Acoustic Model Test Data:
  - `eval03` 6 hours (3 hours Switchboard2 Phase 5, 3 hours Fisher)
  - `dev04` 3 hours Fisher data
- Front-end
  - 12 `PLP` cepstral parameters + C0 and 1st/2nd/3rd derivatives + HLDA
  - Side-based cepstral mean and variance normalisation plus VTLN
- Baseline Acoustic Models
  - Gender independent, decision tree state clustered triphones
  - MPE training with dynamic MMI prior
- Language Models (see RT04f workshop paper for details)
  - 2003 trigram (tgint03) unadapted decodes
  - 2004 evaluation LM for 10xRT experiments

Varying Model Complexity

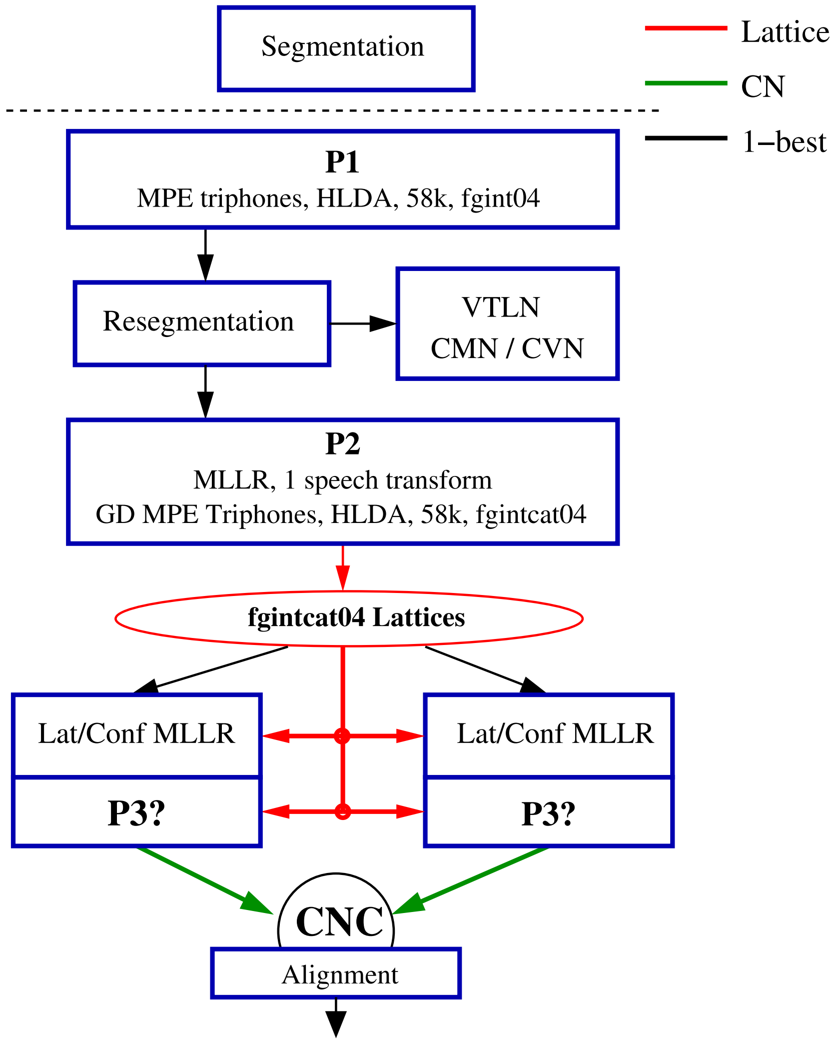
System	# States (# Comp)	MPE Iter	eval03 s25	eval03 fsh	eval03 Avg	dev04
S1	6K (28)	0 (ML)	34.1	26.0	30.2	26.4
S4	9K (36)		33.0	24.8	29.0	25.3
S6	15K (36)		32.1	24.3	28.3	24.3
S1	6K (28)	8	27.9	20.2	24.2	20.5
S4	9K (36)		26.8	19.5	23.3	20.0
S6	15K (36)		26.5	19.5	23.1	19.7

%WER GI unadapted decode 2003 trigram

- S6 system (540K) is $1.7\times$ larger than S4 system (320K) on dev04
  - S6 MLE 1.0%/MPE 0.3% better than S4 MLE/MPE system
- Unfortunately MPE gains consistently less than MLE gains
  - complexity affects MPE gains - on dev04

S1 5.9% S4 5.3% S6 4.6%

10xRT Framework



- Evaluation 10xRT framework:
- Multi-pass framework
- Confusion network generation
- Confusion network combination
- Evaluation system used:
  - P3b: S4: Triphone GD MPron
  - P3q: Q1: Quinphone SAT SPron
- Use alternative P3 branches
  - ignore time constraints ...

10xRT Framework Results

System		eval03			dev04
		s25	fsh	Avg	
P3b-cn	S4 GD	21.7	14.7	18.3	15.1
P3q-cn	Q1 SPRON-SAT	21.5	14.8	18.2	15.0
P3d-cn	S6 GD	21.3	14.5	18.0	14.9
P3s-cn	S4 SAT-SPAM	21.0	14.6	17.9	14.7
P3b+P3q		20.9	14.1	17.6	14.3
P3b+P3d		21.3	14.3	17.9	14.7
P3d+P3q	CNC	20.6	13.9	17.4	14.1
P3s+P3q		20.4	14.0	17.3	14.1
P3s+P3d		20.8	14.3	17.7	14.3

% WER 2004 10xRT rescoring/combination, 2004 RT04f LMs

- Best single branch S4 SAT-SPAM system (too slow for real 10xRT!)
- S6 GD about 0.2% better than S4 GD
- Gain maintained after combination with Q1, 0.2% better than eval system.

Semi-Tied Covariance Matrices (reminder)

- IBM investigated full covariance matrices
  - simpler updates than SPAM/EMLLT systems
  - but dramatic increase in number of parameters/decode cost
  - necessary to limit number of components (IBM: 144K vs 800K)
- Examine simpler precision matrix models - semi-tied covariance matrices
  - simple/efficient update formulae
  - efficient likelihood calculation

$$\mathcal{L}(\mathbf{o};\boldsymbol{\mu}^{(m)},\boldsymbol{\Sigma}^{(m)}_{\text{diag}},\mathbf{A}^{(r)}) = |\text{det}(\mathbf{A}^{(r)})|\mathcal{N}(\mathbf{A}^{(r)}\mathbf{o};\boldsymbol{\mu}^{(m)},\boldsymbol{\Sigma}^{(m)}_{\text{diag}})$$

- HLDA subsumes a global STC transform
- Normally only a small number of semi-tied transforms considered
  - with more data, dramatically increase the number of transforms
  - no need to limit number of components

Unadapted STC Results

System	#STC XForms	MPE Iter	eval03 s25	eval03 fsh	eval03 Avg	dev04
S4	—	0 (ML)	31.6	24.3	28.1	24.6
	1K		31.3	24.0	27.8	—
	9K		30.7	23.1	27.0	23.5
S4	—	8	26.7	19.6	23.3	20.0
	9K		26.3	18.9	22.7	19.4
S6	—	8	26.5	19.4	23.0	19.5

%WER GI unadapted decode, `HDecode`, PronProbs, 2003 trigram

- S4 9K STC system is $1.6\times$ larger than standard S4 system
  - S4 9K STC system MLE 1.1%/MPE 0.6% absolute better than S4 system
  - slightly better (0.1%-0.3%) than S6 system
- Unfortunately adaptation more complex (same as full cov)

fMPE – from the Model Parameter Point of View

- IBM's fMPE — a form of feature interpolation based on posterior information
- Equivalent to temporally varying *shift* of mean vectors

$$\boldsymbol{\mu}_{mt} = \boldsymbol{\mu}_m + \sum_{i=1}^{n} p(c_i|\boldsymbol{o}_t)\boldsymbol{b}_i = \boldsymbol{\mu}_m + \boldsymbol{b}_t$$

  - c_i: cluster centroid, $n \gg d$

- *Static* ($\boldsymbol{\mu}_m$) & *dynamic* ($\boldsymbol{b}_t$) parameters
- Interleaving update of static and dynamic parameters:
  - **static**: update $\boldsymbol{\mu}_m$ (ML); fix $\boldsymbol{b}_i$
  - **dynamic**: update $\boldsymbol{b}_i$ (fMPE); fix $\boldsymbol{\mu}_m$

Update of Temporal Mean Vectors

- Update of $\boldsymbol{\mu}_m$ (ML): $\boldsymbol{\mu}_m = \frac{\sum_{t=1}^{T} \gamma_m(t)(\boldsymbol{o}_t - \boldsymbol{b}_t)}{\sum_{t=1}^{T} \gamma_m(t)}$

- Update of b_{ij} (fMPE): $\hat{b}_{ij} = b_{ij} + \eta_{ij} \frac{\partial \mathcal{F}}{\partial b_{ij}}$

$$
\begin{aligned}
\frac{\partial \mathcal{F}}{\partial b_{ij}} &= \sum_{t=1}^{T} p(c_i|\boldsymbol{o}_t) \left\{ \sum_{m=1}^{M} \left(\frac{\partial \mathcal{F}}{\partial \mathcal{L}^m} \frac{\partial \mathcal{L}^m}{\partial \mu_{mjt}} + \frac{\partial \mathcal{F}}{\partial \mu_{mj}} \frac{\partial \mu_{mj}}{\partial \mu_{mjt}} + \frac{\partial \mathcal{F}}{\partial \sigma_{mj}^2} \frac{\partial \sigma_{mj}^2}{\partial \mu_{mjt}} \right) \right\} \\
\mathcal{L}^m &= K - \frac{1}{2} \sum_{j=1}^{d} \left(\log(\sigma_{mj}^2) - \frac{(o_{jt} - \mu_{mjt})^2}{\sigma_{mj}^2} \right)
\end{aligned}
$$

- Exactly the same as fMPE ...

- But, motivates the extension for *temporal precision matrices*

Temporal Precision Matrices (pMPE)

- Temporal *scaling* of *diagonal* precision elements ($s_{mj} = 1/\sigma^2_{mj}$)

$$s_{mjt} = \left(1 + \sum_{i=1}^{n} p(c_i|\boldsymbol{o}_t)a_{ij}\right)^2 s_{mj} = a^2_{jt}s_{mj}$$

- Positive temporal scaling, a^2_{jt}, to ensure positive variances
- Similar interleaving update as fMPE
- Likelihood calculation more expensive:

$$\mathcal{L}^m = K + \frac{1}{2}\sum_{j=1}^{d}\left(\log(s_{mj}) + \log(a^2_{jt}) - a^2_{jt}s_{mj}(o_{jt} - \mu_{mjt})^2\right)$$

  - cache a^2_{jt} and $\sum_{j=1}^{d}\log(a^2_{jt})$
  - extra d multiplications and 1 addition

Update of Temporal Precision Matrices

- Update of σ^2_{mj} (ML):

$$\sigma^2_{mj} = \frac{\sum_{t=1}^{T} \gamma_m(t) a^2_{jt} (o_{jt} - \mu_{mjt})^2}{\sum_{t=1}^{T} \gamma_m(t)}$$

- Update of a_{ij} (pMPE):

$$\hat{a}_{ij} = a_{ij} + \eta_{ij} \frac{\partial \mathcal{F}}{\partial a_{ij}}$$

$$\frac{\partial \mathcal{F}}{\partial a_{ij}} = 2 \sum_{t=1}^{T} a_{jt} p(c_i|\boldsymbol{o}_t) \left\{ \sum_{m=1}^{M} \left(\frac{\partial \mathcal{F}}{\partial \mathcal{L}^m} \frac{\partial \mathcal{L}^m}{\partial s_{mjt}} + \frac{\partial \mathcal{F}}{\partial \sigma^2_{mj}} \frac{\partial \sigma^2_{mj}}{\partial s_{mjt}} + \frac{\partial \mathcal{F}}{\partial \mu_{mjt}} \frac{\partial \mu_{mjt}}{\partial s_{mjt}} \right) \right\}$$

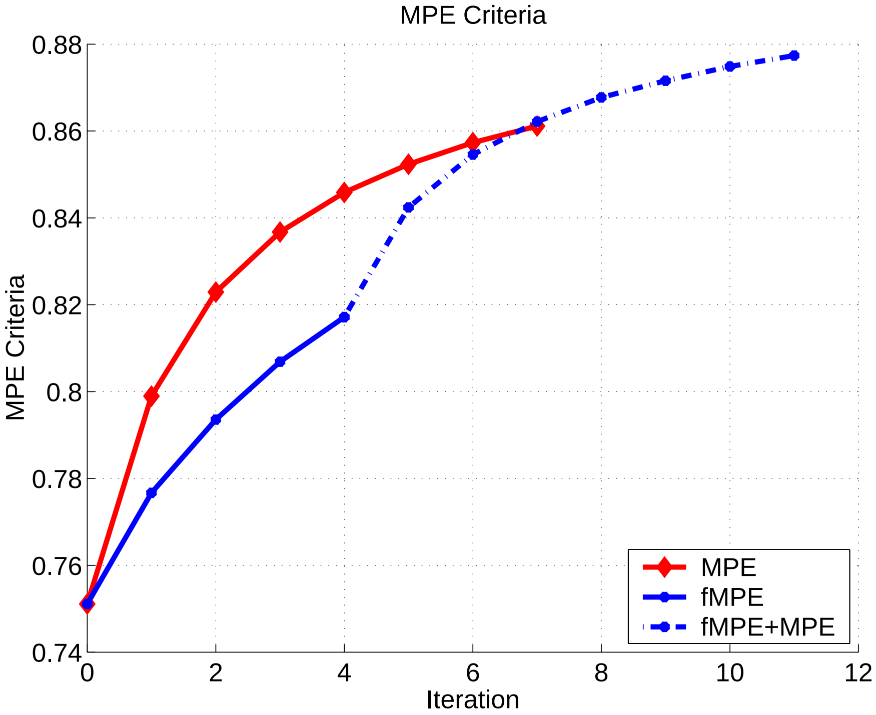
- Learning rate:

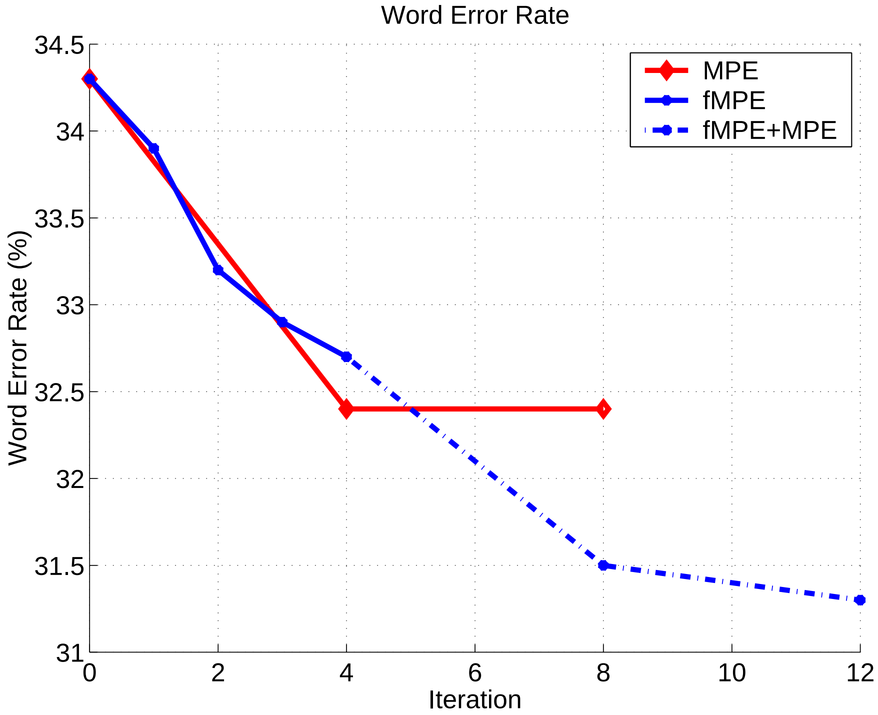
$$\eta_{ij} = \frac{\alpha}{(p_{ij} + n_{ij})}$$

Experimental Setup

- Acoustic model data sets:
  - **Training data**: 76 hours `h5etrain03sub` & 296 hours `h5etrain03`
  - **Test data**: 3 hours `dev01sub` & 6 hours `eval03`
- Front-end: Standard CUED CTS set-up
- Posterior calculations:
  - ~ 70k & ~ 100k Gaussians for posterior calculation
  - Gaussians grouped into 1024 clusters
  - Evaluate the top 5 Gaussians with ~ 2 active posteriors/frame
  - Single frame posteriors *without* context
- Baseline Acoustic Models
  - 12 & 16 component `VarMix` gender independent
  - Decision tree state clustered triphones (~ 6000 states)
  - MPE training *without* dynamic MMI prior
- Trigram Language Models

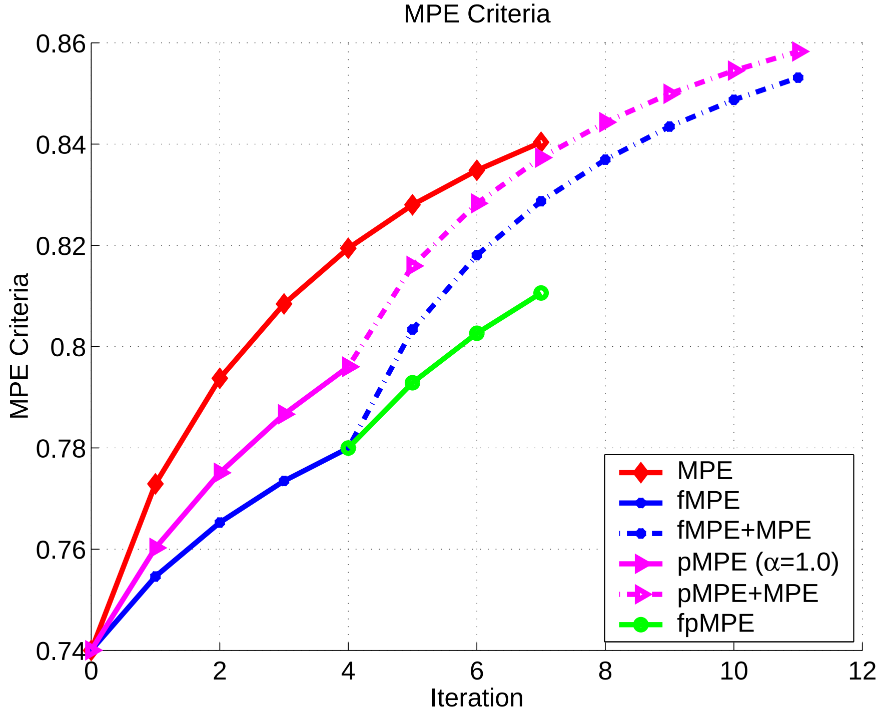
dev01sub results of fMPE trained on h5etrain03sub

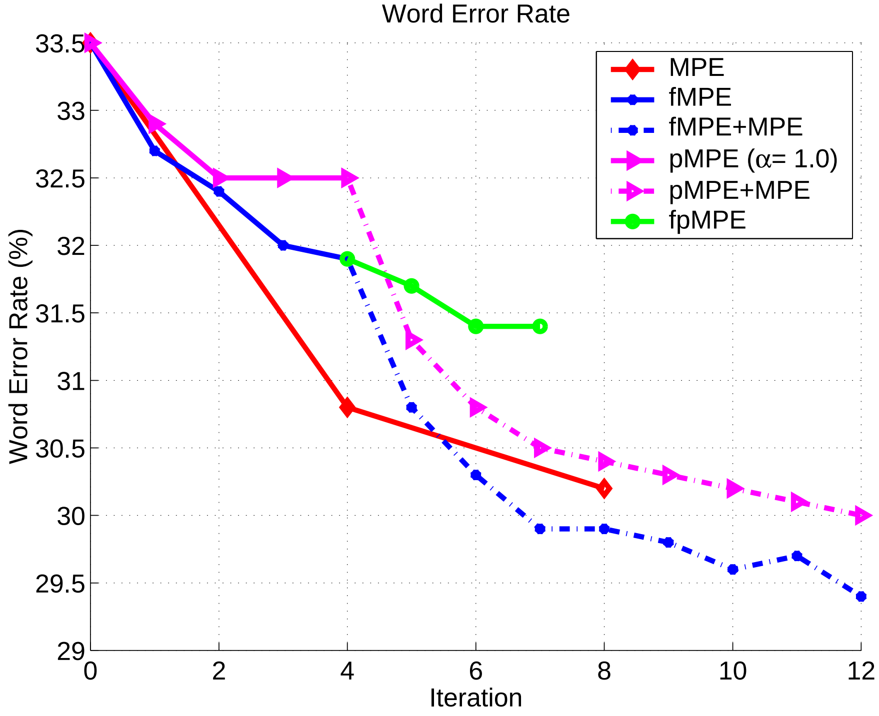




- fMPE (4 iter): 32.7% (+1.6% over ML)
- fMPE+MPE (8 iter): 31.1% (+1.4% over MPE & +3.2% over ML)
- pMPE & pMPE+MPE: less robust to overtraining

`dev01sub` results of fMPE & pMPE trained on `h5etrain03`





- fMPE & fMPE+MPE: similar gain as before
- pMPE converged quicker ($\sim$ 2 iterations) with $\sim$ 1.0% gain over ML
- pMPE+MPE gave 0.2-0.3% gain over MPE alone
- fpMPE further 0.5% gain over fMPE

eval03 results of fMPE & pMPE trained on h5etrain03

System	Iter 0			Iter 4			Iter 8		
	s25	fsh	Avg	s25	fsh	Avg	s25	fsh	Avg
MPE	36.4	27.1	31.9	33.6	24.2	29.1	33.2	23.6	28.6
fMPE+MPE	34.5	25.4	30.1	32.7	23.3	28.1	32.3	22.9	27.8
pMPE+MPE	35.1	26.0	30.7	33.2	24.1	28.8	32.9	23.6	28.4

%WER of 16-component systems on eval03

- Performance of fMPE and pMPE on eval03 similar to dev01sub
- Gains over ML: fMPE (1.8%), pMPE (1.2%), fpMPE (2.0%)
- Improvement over MPE alone: fMPE+MPE (0.8-1.0%), pMPE+MPE (0.2-0.3%) and fpMPE+MPE (0.4-0.5%)

Summary

- `S6` (15k states) gave $\sim$ 0.2-0.3% absolute gain
- `STC 9K` system:
  - alternative approach to building full covariance matrix system
  - gave $\sim$ 0.6% absolute gain (unadapted MPE)
- Initial fMPE results – similar gains to IBM
- Gains from pMPE smaller compared to fMPE
- Future work:
  - apply fMPE to larger training set (`fsh2004h5etrain03b`)
  - investigate interaction between fMPE and pMPE
  - lattice regeneration for fMPE+MPE training