How to build a LVCSR system
( using the Entropic* HTK )

Dr. R. Togneri
CIIPS (SLSR Group)
E&E Eng, UWA

* acquired by Microsoft 29/10/99
A few steps?

Training

- Building the task dictionary
- Creating the training phone and word transcriptions
- Converting the training data to feature vectors
- Building the monophone models
- Building the triphone models
- Making tied-state triphones
- Increasing the number of mixtures
In fact lots of small steps!

**Testing**
- Creating the test phone and word transcriptions
- Converting the test data to feature vectors
- Building the language model / task grammar
- Producing the N-best transcriptions

**Adaptation**
- Perform regression class tree analysis
- "Enrol" speaker
- Adapt models to new speaker either by MLLR or MLLR+MAP re-estimation
The ATIS task

Airline Travel Information System (ATIS)
- Highly constrained task
- Spontaneous speech
  - hesitation, silence, “vocalised” pauses (/er/, /ah/, /um/, . . . ), background noise
- 15,000 training utterances, 450 speakers
  - collected at ATT, BBN, CMU, MIT CSL and SRI

Typical Utterances
- “What is the cheapest airfare to Boston?”
- “What is the earliest flight to NY departing after 1pm?”
**Create ATIS pronunciation dictionary**

- **Add special models**
  - silence at beginning and end of utterances
  - sp (short-pause) at the end of each word

- **HTK Schematic**
  - [global ded edit commands to add sp and silence] +
    {atislist list of ATIS words} + {cmudict English pronunciation dictionary} + {addict ATIS special pronunciation dictionary} → HDMan → {atisdict ATIS pronunciation dictionary} + {monophones list of monophones}

- **HTK Invocation**
  - HDMan -m -w atislist -n monophones1 atisdict cmudict addict
Phone / Word transcriptions

Training

- ATIS transcriptions (*.sro) parsed for word-level transcription (words.mlf) and utterances to be pruned (words.prune)
  - Corrupt data detected and pruned
  - Vocalised pauses mapped to filled-noise model
  - Background noise mapped to noise model

Testing

- Word-level transcriptions ignore spontaneous speech effects (these do not have to be recognised)
  - filled-noise, noise, silence and sp models are stripped from the decoding transcription N-best list
Phone / Word transcriptions

- **words.mlf** is created manually
- **Word-level transcriptions converted to phone-level transcription** (*phones.mlf*)
  
  **HTK Schematic**
  
  - `[mkphones.led edit commands] + {atisdict ATIS pronunciation dictionary} + {words.mlf word-level transcription}`
  - `→ HLEd → {phones.mlf phone-level transcription}`

  **HTK Invocation**
  
  - `HLEd -l ‘ * ’ -d atisdict -i phones.mlf mkphones.led words.mlf`
  - *phones0.mlf* | monophones0 sp model deleted
  - *phones1.mlf* | monophones1 sp model included
Feature extraction

Feature parameters

- **MFCC features** (TARGETKIND = MFCC_0_Z)
  - 12 MFCC + energy term = 13 dim “static” features
  - 25 ms analysis frame
  - 10 ms frame rate
  - Cepstral Mean Normalisation (CMN)

- **config** sampled data to features configuration

Convert (*wav) data to MFCC features (*mfc)

- **codetr.scp** one data file per line
  - <path>/<*.wav> <path>/<*.mfc>
Feature extraction

-HTK Invocation
   - HCopy -T 1 -C config -S codetr.scp

- List of `<path>/*.mfcc` feature files
   - `train.scp` one utterance per line (12150 for training)
   - `test.scp` one utterance per line (1976 for testing)

- Augmenting features
  - features augmented directly when applied to HMM models for training/testing/adaptation
    - `MFCC_O_Z` \(\Rightarrow\) `MFCC_O_Z_D_A`
    - 13 static \(\Rightarrow\) 13 static + 13 delta + 13 delta-delta = 39
HMM topology

- HMM phone/silence/noise model topology
  - 3-state left-right no-skip
    - 5-state HTK to include ENTER and EXIT states
  - Diagonal covariance, Gaussian pdf
    - Initially 1-mixture then increased to 5-mixture
    - Issue 1-mixture needed since HTK cannot do decision tree-based clustering with multiple mixtures. Why?

- HMM short-pause (sp) model topology
  - 1-state bypass-skip
  - State tied to centre state of silence
HMM parameter estimation

- **HMM parameters to estimate (per model)**
  - 6x transition probabilities
    - self-loop and next state probabilities for 3 states
  - 1-mixture Gaussian pdf per state
    - 39-dim mean
    - 39-dim diagonal covariance
  - **Total parameters (1-mixture)**
    - 6 probabilities + 3 x (39+39) pdf = 240 parameters
  - **Total parameters (5-mixtures)**
    - approx. 1200 parameters
Model initialisation

- **HMM model files**
  - `proto` specifies HMM topology and features
  - `hmmdefs` HMM MMF models file
  - `macros` HMM MMF global macros file

- **Flat-start**
  - Initialise models by global mean and covariance
  - **HTK Schematic**
    - `{train.scp` list of training utterances} +
    - `{proto` initial HMM topology specification}
    - → HCompV → `{hmm0/proto` initialised prototype}`
Model initialisation

- **HTK Invocation**
  - `HCompV -C config -f 0.01 -m -S train.scp -M hmm0 proto`

- **Required models then cloned from** *proto*

**Bootstrapping**

- **First, initialise models by flat-start**
- **Second, use phonetically transcribed TIMIT data**
- **to “bootstrap” monophone models**

- **Run HInit followed by HRest**
  - `HInit`  K-means segmentation
  - `HRest` Baum-Welch re-estimation
Model initialisation

- **HTK Schematic**
  - \{\textit{timitlabs.mlf}\} phone-level transcription \textit{with segment “boundaries”}\} + \{\textit{train-timit.scp}\} location of utterance feature files\} + \{\textit{hmm0/hmmdefs}\} HMM models file\} + \{\textit{phone}\} name of monophone model to train\}
  
  \rightarrow \text{HInit} | \text{HRest} \rightarrow \{\textit{hmm0/0/hmmdefs}\} re-estimated model\}

- **HTK Invocation**
  - \text{HInit|HRest} -C config -T 1 -I timitlabels.mlf -S train-timit.scp
    
    -H hmm0/macros -H hmm0/hmmdefs -M hmm0/0
    
    -l \${phone} \${phone}

- **Use \textit{phones0.mlf} and \textit{monophones0}\**

- **Get good silence estimate first, then create sp**
Model re-training

3 HERest runs after each model modification

- Follows every modification described
  - model initialisation
  - creating sp model
  - Viterbi realignment
  - creating triphone models
  - clustering and tying states
  - increasing the number of mixtures

- HERest
  - Performs embedded Baum-Welch re-estimated
  - Does not require phone “boundary” information
  - Trains all data and models concurrently
Model re-training

- **Monophone model re-training**
  - **HTK Schematic**
    - `{phones.mlf` phone-level transcription}+
    - `{train.scp` list of utterances}+
    - `{hmm0/hmmdefs` HMM models file}+
    - `{monophones` list of HMM models}
      → HERest → `{hmm1/hmmdefs` re-estimated models}
  - **HTK Invocation**
    - HERest -C config -T 1 -I phones.mlf -S train.scp
      -H hmm0/macros -H hmm0/hmmdefs -M hmm1
      monophones
    - **2nd run:** -H hmm1/macros -H hmm1/hmmdefs -M hmm2
    - **3rd run:** -H hmm2/macros -H hmm2/hmmdefs -M hmm3
Creating the **sp** model

- **Make sp = middle state of silence**
  - Need to add transitions and tie to silence
  - **HTK Schematic**
    - `[sil.hed directives to add transitions and tie to silence] +
      `{hmm4/hmmdefs HMM models file} +
      `{monophones list of monophones}
    → HHEd → `{hmm5/hmmdefs includes new sp model}
  - **HTK Invocation**
    - HHEd -H hmm4/macros -H hmm4/hmmdefs
      -M hmm5 sil.hed monophones1
Multiple pronunciations per word

- Use Viterbi decoding to select best-match pronunciation hypothesis
  - No change if there is only one pronunciation

HTK Invocation

- HVite -l '*' -o SWT -T 1 -b SILENTPAUSE -C config
  -a -H hmm7/macros -H hmm7/hmmdefs -i aligned.mlf -m -y lab
  -I words.mlf -S train.scp atisdict monophones1

HTK Description

- Uses hmm7/hmmdefs HMM models and training data in train.scp with initial words.mlf word transcription and atisdict possible pronunciations to create new aligned.mlf phone (best pronunciation) transcription
Creating triphone models

Generate triphone transcription file

- **Word-internal (no crossword) CD triphones**
  - monophones: sill th lh s sp m ae n sp
  - triphones: sill th+ih th-ih+s ih-s sp m+ae m-ae+n ae-n sp

**HTK Schematic**
- `[mktri.led edit commands] + {aligned.mlf phone transcriptions}
  → HLEd →{wintri.mlf triphone transcriptions} +
  {triphones list of triphone models}

**HTK Invocation**
- HLEd -n triphones -l '*' -I wintri.mlf mktri.led aligned.mlf
Creating triphone models

Generate triphone HMM models

- **Tie transition matrix for same triphone base**
  - *-aw+-* belong to base /aw/

**HTK Schematic**

- [mktri.hed edit commands and directives to generate triphones and tie states] + {hmm9/hmmdefs HMM phone models list} + {monophones1 list of phones}
  → HHEd →{hmm10/hmmdefs HMM triphone models list}

**HTK Invocation**

- HHEd -B -H hmm9/macros -H hmm9/hmmdefs -M hmm10 mktri.hed monophones1
Model re-training

- **Triphone model re-training**
  - **HTK Schematic**
    - `{wintri.mlf` triphone transcription of all utterances}+
    - `{train.scp` list of utterances}+
    - `{hmm10/hmmdefs` HMM models file}+
    - `{triphones` list of HMM triphone models}
    - \textbf{\rightarrow HERest} \rightarrow `{hmm11/hmmdefs` re-estimated models}
  - **HTK Invocation**
    - \textbf{HERest -B -C config -T 1 -I wintri.mlf -S train.scp}
      - `-H hmm10/macros -H hmm10/hmmdefs -M hmm11 triphones`
    - \textbf{Use option ‘-B’ to read and write binary HMM MMF \texttt{hmmdefs} files (to save space with so many models!)}
Building tied-state triphones

Why tie states?

- Too many parameters for too few data
  - unreliable parameter estimates
  - increased computational and memory requirements
- Gaussian distributions in tied states are shared
  - $N$ triphone models are shared
    - $3 \times (39+39) \times N$ parameters before tying
    - $3 \times (39+39)$ parameters after tying

Which states to tie?

- Cluster states from same triphone base
Building tied-state triphones

Data-driven clustering

- **Init** Place all states in individual clusters
- **Merge** Find pair of clusters which when merged form the smallest resultant cluster
- **Repeat Merge until**
  - size of largest cluster < TC
  - number of clusters < NC
- **Effective, but not knowledge driven**
Building tied-state triphones

- Binary classification tree clustering
  - Pool all states into a single Gaussian cluster
  - Split pool according to YES/NO question
    - Phonetic question of the left or right context
    - e.g. “Is the left context a nasal?”
  - Calculate log likelihood of split pool and find question producing max. log likelihood
    - Keep asking questions, splitting pool and calculating log likelihood until the question which maximally splits the pool is found
  - Repeat for each node of tree until threshold
    - increase in log likelihood by splitting < TB
Building tied-state triphones

- **Advantage of synthesising unseen triphones**
  - For unseen triphones travel down the tree (by answering the question at each node) to get to leaf node and then synthesise triphone from leaf node pool

- **HTK Schematic**
  - [tree.hed directives specifying the questions to ask and tied-state tying action by decision tree clustering including synthesising unseen triphones] + {hmm12/hmmdefs HMM triphone models file} + {triphones list of triphone models} → HHEd → {hmm13/hmmdefs HMM tied-state and unseen synthesised triphone models file} + {tiedlist list of tied triphone models}

- **HTK Invocation**
  - HHEd -B -H hmm12/macros -H hmm12/hmmdefs -M hmm13 tree.hed triphones
Model re-training

- **Tied-state triphone model re-training**
  
  - **HTK Schematic**
    
    - \{wintri.mlf\} triphone transcription of all utterances}+
    - \{train.scp\} list of utterances}+
    - \{hmm13/hmmdefs\} HMM models file}+
    - \{tiedlist\} list of HMM triphone models}
    
    → HERest → \{hmm14/hmmdefs\} re-estimated models
  
  - **HTK Invocation**
    
    - HERest -B -C config -T 1 -I wintri.mlf -S train.scp
    -H hmm13/macros -H hmm13/hmmdefs -M hmm14 tiedlist
Increase number of mixtures

- Increment number of mixtures to 5
  - Split mixture by copying and perturbing
    - Increase to 2, 3, 5 and re-train models at each step
  - HTK Schematic
    - [adjmixtures.hed directive to increase the number of mixtures] + {hmm15/hmmdefs HMM triphone models file} + {triphones list of triphone models} → HHEd → {hmm16/hmmdefs HMM triphone increased mixtures models file}
  - HTK Invocation
    - HHEd -B -H hmm15/macros -H hmm15/hmmdefs -M hmm16 adjmixtures.hed tiedlist
Adapting the HMM models

**Adaptation data**
- Select a small sample of training utterances
- Generate the adaptation data
  - Select speaker and environment for adaptation
  - Acquire adaptation sample data, transform to feature data and create word-level transcriptions

**Binary regression class tree construction**
- HMM model parameters clustered into classes
  - Apply same adaptation transformation to class
- Use centroid-splitting algorithm to grow tree
- HTK Invocation
  - HHEd command with regtree.hed
Adapting the HMM models

Transform the HMM models

- **MLLR** \{maximise \( L = P(x/\theta) \) \}
  - Apply affine transformation to Gaussian means \( (= \theta) \)
    - also covariances
  - Choose transformation parameters so as to maximise the log likelihood \( \rightarrow \) MLLR

- **MAP** \{maximise \( L = P(x/\theta)P(\theta) \) \}
  - Maximise the likelihood expression assuming an informative prior, \( P(\theta) \), for the mean \( (= \theta) \)
  - Initial model mean can act as prior
  - MLLR estimate of mean can act as prior
Adapting the HMM models

**HTK Procedure**

- **Use HVite to perform Viterbi re-alignment**
  - Convert word-level transcriptions of adaptation data to phone-level transcriptions

- **Use two-steps of HEAdapt (MLLR)**
  - HEAdapt -B -C config -g -S adapt.scp -I adaptPhones.mlf -H hmm/macros -H hmm/hmmdefs -K global.tmf tiedlist
  - HEAdapt -B -C config -g -S adapt.scp -I adaptPhones.mlf -H hmm/macros -H hmm/hmmdefs -K rc.tmf -J global.tmf tiedlist
  - file rc.tmf contains MLLR transformation information
    - use ‘-J rc.tmf’ with HVite when decoding to load transformation
  - use ‘-j’ option for MAP transformation to a new hmmdefs file
    - use newly transformed hmmdefs with HVite when decoding
Building the language model

- **Word network**
  - Defines the task grammar
    - identifies words and their contextual relationship
    - finite state grammar
      - construction in Standard Lattice Format (SLF) using Extended Backus-Naur Form (EBNF) grammar notation
    - Example
      - `sil < one | two | three > sil`
  - Only useful for highly constrained and well-defined task applications
    - ATIS is highly constrained but not well-defined
Building the language model

**Statistical Language Model (LM)**

- **Backed-off Bigram LM**
  - use training transcriptions to estimate the bigram word probabilities, \( P(j|i) \), of word \( j \) following word \( i \)
  - back-off to unigram, \( P(i) \), if insufficient data to estimate \( P(j|i) \)

- **HTK Schematic**
  - \{atislist list of ATIS words\} + \{words.mlf word level transcription of training data\} → HLStats
  - \{bgraml backed-off bigram language model estimates\}

- **HTK Invocation**
  - HLStats -o -b bgraml atislist words.mlf
Building the language model

Building the word network from the LM

- **HTK Schematic**
  - \{bgraml bigram language model\} + \{atislist list of ATIS words\}
  - \(\rightarrow\) HBuild \(\rightarrow\) \{wdnet SLF word network\}

- **HTK Invocation**
  - HBuild -b -n bgraml atislist wdnet
**Viterbi decoding ➞ N-best lists**

#### Viterbi search with fast decoding

- **Constrain search to most likely candidate words**
  - weight Viterbi score by \(wdnet\) bigram probability
- **Prune low probability paths (beam search)**
  - Delete paths with score < \(\text{max. score - threshold}\)
- **HTK Schematic**
  - \([\text{arguments}] + \{\text{test.scp list of test utterances}\} + \{\text{wdnet word network}\} + \{\text{atisdict ATIS pronunciation dictionary}\} + \{\text{tiedlist list of tied triphone models}\} \rightarrow HVite \rightarrow \{\text{recout.mlf N-best list}\}\)
Viterbi decoding $\Rightarrow$ N-best lists

- **HTK Invocation**
  - HVite -C config -H hmm/macros -H hmm/hmmdefs -S test scp -l '*' -i recout.mlf -n 20 100 -w wdnet -s 25 -t 250 atisdict tiedlist

- **HTK Arguments**
  - -n 20 100 100-best list using 20 tokens
  - -t 250 enable beam search with specified threshold
  - -s 25 grammar scale factor for bigram LM weighting
    - large scale for highly constrained ATIS task
**Why not sentence recognition rate?**
- Always produces a poor result since a sentence is wrong even if one word is in error
- Robust NLP should cope with insertion, deletion and substitution errors in sentence

**Word Error Rate (WER) results**
- **Best**: 4% WER ("highly tuned")
- **ATIS test**: 14% WER (no adaptation)
- **Oz speaker**: 18% WER (no adaptation)
  - 16% WER (adaptation)
- **Non-native**: 50% WER (no adaptation)
  - 22% WER (adaptation)
Conclusions

**HTK Implementation Issues**
- Decision Tree-based clustering for tying states cannot use multiple mixtures per state, why?
- No support for trigram LM or class-based LM
  - Implement class-based LM manually

**HTK Status**
- HTK 2.2 is last version but it is license “free”
  - thanks Microsoft!
- Building a complete ASR system
  - UNIX HTK to train and build models
  - Windows HAPI for integrated ASR application design