Introduction to HTK Toolkit

Berlin Chen
Department of Computer Science & Information Engineering
National Taiwan Normal University

Reference:
Outline

• An Overview of HTK
• HTK Processing Stages
• Data Preparation Tools
• Training Tools
• Testing Tools
• Analysis Tools
• Homework: Exercises on HTK
An Overview of HTK

• HTK: A toolkit for building Hidden Markov Models

• HMMs can be used to model any time series and the core of HTK is similarly general-purpose

• HTK is primarily designed for building HMM-based speech processing tools, in particular speech recognizers
An Overview of HTK (cont.)

- Two major processing stages involved in HTK
  - **Training Phase:** The training tools are used to estimate the parameters of a set of HMMs using training utterances and their associated transcriptions.
  - **Recognition Phase:** Unknown utterances are transcribed using the HTK recognition tools.

![Diagram of HTK processing stages]
An Overview of HTK (cont.)

• HTK Software Architecture
  – Much of the functionality of HTK is built into the library modules
    • Ensure that every tool interfaces to the outside world in exactly the same way

• Generic Properties of an HTK Tools
  – HTK tools are designed to run with a traditional command line style interface

```
HFoo -T -C Config1 -f 34.3 -a -s myfile file1 file2
```

• The main use of configuration files is to control the detailed behavior of the library modules on which all HTK tools depend
HTK Processing Stages

• Data Preparation
• Training
• Testing/Recognition
• Analysis
Data Preparation Phase

• In order to build a set of HMMs for acoustic modeling, a set of speech data files and their associated transcriptions are required
  – Convert the speech data files into an appropriate parametric format (or the appropriate acoustic feature format)
  – Convert the associated transcriptions of the speech data files into an appropriate format which consists of the required phone or word labels

• **HSLAB**
  – Used both to record the speech and to manually annotate it with any required transcriptions if the speech needs to be recorded or its transcriptions need to be built or modified
Data Preparation Phase (cont.)

![Waveform: noname1.wav, label: noname1.lab, Num samples 40000, HTK sampling rate: 16000 KHz]

**Fig. 14.1** HSLab display window
Data Preparation Phase (cont.)

- **HCOPY**
  - Used to parameterize the speech waveforms to a variety of acoustic feature formats by setting the appropriate configuration variables

  ![Diagram of speech processing]

  **MFCC-based Front-End Processor.**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPC</td>
<td>linear prediction filter coefficients</td>
</tr>
<tr>
<td>LPCREFC</td>
<td>linear prediction reflection coefficients</td>
</tr>
<tr>
<td>LPCEPSTRA</td>
<td>LPC cepstral coefficients</td>
</tr>
<tr>
<td>LPDELCEP</td>
<td>LPC cepstra plus delta coefficients</td>
</tr>
<tr>
<td>MFCC</td>
<td>mel-frequency cepstral coefficients</td>
</tr>
<tr>
<td>MELSPEC</td>
<td>linear mel-filter bank channel outputs</td>
</tr>
<tr>
<td>DISCRETE</td>
<td>vector quantized data</td>
</tr>
</tbody>
</table>
Data Preparation Phase (cont.)

• **HLIST**
  – Used to check the contents of any speech file as well as the results of any conversions before processing large quantities of speech data

• **HLED**
  – A script-driven text editor used to make the required transformations to label files, for example, the generation of context-dependent label files

• **HLSTATS**
  – Used to gather and display statistical information for the label files

• **HQUANT**
  – Used to build a VQ codebook in preparation for building discrete probability HMM systems
Training Phase

• Prototype HMMs
  – Define the topology required for each HMM by writing a prototype Definition
  – HTK allows HMMs to be built with any desired topology
  – HMM definitions stored as simple text files
  – All of the HMM parameters (the means and variances of Gaussian distributions) given in the prototype definition are ignored only with exception of the transition probability

```xml
~o <VecSize> 39 <MFCC_0_D_A>
~h "proto"
<BeginHMM>
  <NumStates> 5
  <State> 2
    <Mean> 39
      0.0 0.0 0.0 ... 
    <Variance> 39
      1.0 1.0 1.0 ...
  <State> 3
    <Mean> 39
      0.0 0.0 0.0 ... 
    <Variance> 39
      1.0 1.0 1.0 ...
  <State> 4
    <Mean> 39
      0.0 0.0 0.0 ... 
    <Variance> 39
      1.0 1.0 1.0 ...
  <TransP> 5
    0.0 1.0 0.0 0.0 0.0 
    0.0 0.0 0.6 0.0 0.0 
    0.0 0.0 0.6 0.4 0.0 
    0.0 0.0 0.0 0.7 0.3
    0.0 0.0 0.0 0.0 0.0 
<EndHMM>
```
Training Phase (cont.)

- There are two different versions for acoustic model training which depend on whether the sub-word-level (e.g. the phone-level) boundary information exists in the transcription files or not.

  - If the training speech files are equipped the sub-word boundaries, i.e., the location of the sub-word boundaries have been marked, the tools \textit{HINIT} and \textit{HREST} can be used to train/generate each sub-word HMM model individually with all the speech training data.
Training Phase (cont.)

• **HINIT**
  – Iteratively computes an initial set of parameter value using the *segmental k-means training* procedure
    • It reads in all of the bootstrap training data and cuts out all of the examples of a specific phone
    • On the first iteration cycle, the training data are uniformly segmented with respective to its model state sequence, and each model state matching with the corresponding data segments and then means and variances are estimated. If mixture Gaussian models are being trained, then a modified form of k-means clustering is used
    • On the second and successive iteration cycles, the uniform segmentation is replaced by Viterbi alignment

• **HREST**
  – Used to further re-estimate the HMM parameters initially computed by **HINIT**
  – *Baum-Welch* re-estimation procedure is used, instead of the segmental k-means training procedure for **HINIT**
Training Phase (cont.)

K-means

Global mean

Cluster 1 mean

Cluster 2 mean

\{\mu_{12}, \Sigma_{12}, \omega_{12}\} \rightarrow \{\mu_{11}, \Sigma_{11}, \omega_{11}\}

\{\mu_{13}, \Sigma_{13}, \omega_{13}\} \rightarrow \{\mu_{14}, \Sigma_{14}, \omega_{14}\}
Training Phase (cont.)

Prototype HMM

Uniform Segmentation

Initialise Parameters

Viterbi Segmentation

Update HMM Parameters

Converged?

Yes

Initialised HMM

HInit Operation

Input HMM Def

HInit

Output HMM Def

File Processing in HInit

data/tr1.mfc
data/tr2.mfc
data/tr3.mfc
data/tr4.mfc
data/tr5.mfc
data/tr6.mfc

labs/tr1.lab
labs/tr2.lab
labs/tr3.lab
labs/tr4.lab
labs/tr5.lab
labs/tr6.lab
Training Phase (cont.)

1. Initial HMM
2. Forward/Backward Algorithm
3. Update HMM Parameters
4. Converged?
   - No
   - Yes → Estimated HMM
Training Phase (cont.)

• On the other hand, if the training speech files are not equipped the sub-word-level boundary information, a so-called flat-start training scheme can be used
  – In this case all of the phone models are initialized to be identical and have state means and variances equal to the global speech mean and variance. The tool HCOMPV can be used for this

• HCOMPV
  – Used to calculate the global mean and variance of a set of training data
Training Phase (cont.)

• Once the initial parameter set of HMMs has been created by either one of the two versions mentioned above, the tool HEREST is further used to perform embedded training on the whole set of the HMMs simultaneously using the entire training set.
Training Phase (cont.)

- **HEREST**
  - Performs a single *Baum-Welch* re-estimation of the whole set of the HMMs simultaneously
  - For each training utterance, the corresponding phone models are concatenated and the forward-backward algorithm is used to accumulate the statistics of state occupation, means, variances, etc. for each HMM in the sequence
  - When all of the training utterances has been processed, the accumulated statistics are used to re-estimate the HMM parameters
  - **HEREST** is the core HTK training tool
Training Phase (cont.)

• Model Refinement
  – The philosophy of system construction in HTK is that HMMs should be refined incrementally
  – **CI to CD**: A typical progression is to start with a simple set of single Gaussian context-independent phone models and then iteratively refine them by expanding them to include context-dependency and use multiple mixture component Gaussian distributions
  
  \[
  \begin{align*}
  \psi (j) & \rightarrow (j_a) \rightarrow \epsilon (au) \\
  (j_e) & \rightarrow (e)
  \end{align*}
  \]
  
  right-context-dependent modeling

  – **Tying**: The tool **HHED** is a HMM definition editor which will clone models into context-dependent sets, apply a variety of parameter tyings and increase the number of mixture components in specified distributions
  
  – **Adaptation**: To improve performance for specific speakers the tools **HEADAPT** and **HVITE** can be used to adapt HMMs to better model the characteristics of particular speakers using a small amount of training or adaptation data
Recognition Phase

• **HVITE**
  - Performs Viterbi-based speech recognition
  - Takes a network describing the allowable word sequences, a dictionary defining how each word is pronounced and a set of HMMs as inputs
  - Supports cross-word triphones, also can run with multiple tokens to generate lattices containing multiple hypotheses
  - Also can be configured to rescore lattices and perform forced alignments
  - The word networks needed to drive HVITE are usually either simple word loops in which any word can follow any other word or they are directed graphs representing a finite-state task grammar
    • **HBUILD** and **HPARSE** are supplied to create the word networks
Recognition Phase (cont.)
Recognition Phase (cont.)

• Generating Forced Alignment
  – HVite computes a new network for each input utterance using the word level transcriptions and a dictionary
  – By default the output transcription will just contain the words and their boundaries. One of the main uses of forced alignment, however, is to determine the actual pronunciations used in the utterances used to train the HMM system
Analysis Phase

• The final stage of the HTK Toolkit is the analysis stage
  – When the HMM-based recognizer has been built, it is necessary to evaluate its performance by comparing the recognition results with the correct reference transcriptions. An analysis tool called HRESULTS is used for this purpose

• HRESULTS
  – Performs the comparison of recognition results and correct reference transcriptions by using dynamic programming to align them
  – The assessment criteria of HRESULTS are compatible with those used by the US National Institute of Standards and Technology (NIST)

\[
\begin{array}{ccc}
  t_{s1} & t_{e1} & a \\
  t_{s2} & t_{e2} & b \\
  t_{s3} & t_{e3} & b \\
  \cdot & \cdot & \\
  t_{s1} & t_{e1} & a \\
  t_{s2} & t_{e2} & c \\
  t_{s3} & t_{e3} & b \\
  \cdot & \cdot & \\
\end{array}
\]

reference \hspace{2cm} test
A Tutorial Example

• A Voice-operated interface for phone dialing

_Dial three three two six five four_
_Dial nine zero four one oh nine_
_Phone Woodland_
_Call Steve Young_

(regular expression)

– $digit = \text{ONE | TWO | THREE | FOUR | FIVE | SIX | SEVEN | EIGHT | NINE | OH | ZERO;}

$name = [ \text{JOOP } \text{JANSEN | [ JULIAN } \text{ODELL | [ DAVE } \text{OLLASON | [ PHIL } \text{WOODLAND | [ STEVE } \text{YOUNG;}}$

( SENT-START ( DIAL <$digit> | (PHONE|CALL) $name) SENT-END )
Grammar for Voice Dialing

- Grammar for Phone Dialing
Network

- The above high level representation of a task grammar is provided for user convenience.
- The HTK recognizer actually requires a word network to be defined using a low level notation called *HTK Standard Lattice Format* (SLF) in which each word instance and each word-to-word transition is listed explicitly.

```
HParse gram wdnet
```
Dictionary

- A dictionary with a few entries

<table>
<thead>
<tr>
<th>Word</th>
<th>Pronunciation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ah sp</td>
</tr>
<tr>
<td>A</td>
<td>ax sp</td>
</tr>
<tr>
<td>A</td>
<td>ey sp</td>
</tr>
<tr>
<td>CALL</td>
<td>k ao l sp</td>
</tr>
<tr>
<td>DIAL</td>
<td>d ay ax l sp</td>
</tr>
<tr>
<td>EIGHT</td>
<td>ey t sp</td>
</tr>
<tr>
<td>PHONE</td>
<td>f ow n sp</td>
</tr>
<tr>
<td>SENT-END</td>
<td>sil</td>
</tr>
<tr>
<td>SENT-START</td>
<td>sil</td>
</tr>
<tr>
<td>SEVEN</td>
<td>s eh v n sp</td>
</tr>
<tr>
<td>TO</td>
<td>t ax sp</td>
</tr>
<tr>
<td>TO</td>
<td>t uw sp</td>
</tr>
<tr>
<td>ZERO</td>
<td>z ia r ow sp</td>
</tr>
</tbody>
</table>

- Function words such as A and TO have multiple pronunciations
- The entries
- For SENTSTART and SENTEND have a silence model sil as their pronunciations and null output symbols
Transcription

- To train a set of HMMs, every file of training data must have an associated phone level transcription
- Master Label File (MLF)

```bash
#!/MLF!
"*/S0001.lab"
ONE
VALIDATED
ACTS
OF
SCHOOL
DISTRICTS

"*/S0002.lab"
TWO
OTHER
CASES
ALSO
WERE
UNDER
ADVICE

"*/S0003.lab"
BOTH
FIGURES
(etc.)
```
Coding The Data

- Configuration (Config)

```plaintext
# Coding parameters
TARGETKIND = MFCC_0
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = F
```

- Pre-emphasis filter coefficient
- Filter bank numbers
- Cepstral Lifting Setting
- Number of output cepstral coefficients
Coding The Data (cont.)

HCopy -T 1 -C config -S codetr.scp
Training

Prototype HMM Definition
(proto)

HCOMPV

Training Files listed in
(train.scp)

HMM list
(monophones0)

hmm0
macros
hmndefs

hmm1
macros
hmndefs

HERest

Phone Level Transcription
(phones0.mlf)
Tee Model
Recognition

- HVite -T 1 -S test.scp -H hmmset -i results -w wdnet dict hmmlist
- HResults -l refs wlist results

\[
\text{Percent Correct} = \frac{N - D - S}{N} \times 100\%
\]

\[
\text{Percent Accuracy} = \frac{N - D - S - I}{N} \times 100\%
\]
Exercises on HTK

• Practice the use of HTK
• Five Major Steps
  – Environment Setup
  – Data Preparation
    HCopy
  – Training
    HHed, HCompV, HERest
    Or Hinit, HHed, HRest, HERest
  – Testing/Recognition
    HVite
  – Analysis
    HResults
Experimental Environment Setup

- Download the HTK toolkit and install it
- Copy zipped file of this exercise to a directory name “HTK_Tutorial”, and unzipped the file
- Ensure the following subdirectories have been established (If not, make the subdirectories !)
Step01_HCopy_Train.bat

- **Function:**
  - Generate MFCC feature files for the training speech utterances

- **Command**

  ```
  HCOPY -T 00001 -C ..\config\HCOPY.fig -S ..\script\HCopy_Train.scp
  ```

  Level of trace information

  - user defined wave format
  - file header (set to 0 here)
  - in accordance with sampling rate
  - Z(zero mean), E(Energy), D(delta)
  - A(Delta Delta)

  2 bytes per sample

  Hamming window

  Pre-emphasis

  filter bank no

  liftering setting

  Cepstral coefficient no

  Intel PC byte Order

  #Coding parameters
  SOURCEFORMAT=ALEN
  HEADERSIZE=0
  SOURCERATE=625
  1e7/16000
  TARGETKIND=MFCC_Z_E_D_A
  TARGETRATE=1000000 1e-3 *1e7
  #framseshift 10ms
  SAVECOMPRESSED=F
  SAVETHRCRC=F
  WINOFFSET=320000.0
  WINDOWSIZE=320000.0
  # framesize = 32ms
  USEHAMMING=T
  PREEMPHF=0.97
  NUMCHANS=26
  CEPLIFTER=22
  NUMCEPS=12
  ENORMALIZE=T
  NATURALREADORDER=TRUE
  NATURALWRITEORDER=TRUE
  ```

  specify the detailed configuration for feature extraction

  specify the pcm and coefficient files and their respective directories

  - specify the detailed configuration for feature extraction

  - specify the pcm and coefficient files and their respective directories
Step02_HCompv_S1.bat

• Function:
  – Calculate the global mean and variance of the training data
  – Also set the prototype HMM

• Command:

```
HCompV -C ..\Config\Config.fig -m -S ..\script\HCompV.scp -M ..\Global_pro_hmm_def39
..\HTK_pro_hmm_def39\pro_39_m1_s1
```

The prototype 1-state HMM with zero mean and variance of value 1

mean will be updated

a list of coefficient files

the resultant prototype HMM (with the global mean and variance setting)

• Similar for the batch instructions

   Step02_HCompv_S2.bat
   Step02_HCompv_S3.bat
   Step02_HCompv_S4.bat

Generate prototype HMMs with different state numbers
Step02_HCompv_S1.bat (count.)

- Note! You should manually edit the resultant prototype HMMs in the directory “Global_pro_hmm_def39” to remove the row
  ~h “prot_39_m1_sX”

  - Remove the name tags, because these proto HMMs will be used as the prototypes for all the INITIALs, FINALs, and silence models

```
<STREAMINFO> 1 39
<STREAM> 39
<h "prot_39_m1_s1"
<BEGINMARK>
<NUMSTATES> 3
<STATE> 2
<MEAN> 39
  1.813439e-008 1.162030e-008 -2.191493e-009 -3.174569e-009 -2.966750e-009 2.4
<VARIANCE> 39
  5.612159e+001 5.308986e+001 6.076218e+001 4.772774e+001 5.011119e+001 4.7409
<GCONST> 9.571951e+001
<TRANS> 3
  0.000000e+000 1.000000e+000 0.000000e+000
  0.000000e+000 7.000000e-001 3.000000e-001
  0.000000e+000 0.000000e+000 0.000000e+000
<ENDMARK>
```

---

```
<STREAMINFO> 1 39
<Vocoder> 39<brm>
<BEGINMARK>
<NUMSTATES> 3
<STATE> 2
<MEAN> 39
  1.813439e-008 1.162030e-008 -2.191493e-009 -3.174569e-009 -2.966750e-009 2.4
<VARIANCE> 39
  5.612159e+001 5.308986e+001 6.076218e+001 4.772774e+001 5.011119e+001 4.7409
<GCONST> 9.571951e+001
<TRANS> 3
  0.000000e+000 1.000000e+000 0.000000e+000
  0.000000e+000 7.000000e-001 3.000000e-001
  0.000000e+000 0.000000e+000 0.000000e+000
<ENDMARK>
```
Step03_CopyProHMM.bat

• Function
  – Copy the prototype HMMs, which have global mean and variances setting, to the corresponding acoustic models as the prototype HMMs for the subsequent training process

• Content of the bath file

```bash
copy "\Global_pro_hmm_def39\pro_39_m1_s2" "\Init_pro_hmm\sic_c"
copy "\Global_pro_hmm_def39\pro_39_m1_s2" "\Init_pro_hmm\sic_a"
copy "\Global_pro_hmm_def39\pro_39_m1_s2" "\Init_pro_hmm\sic_o"
copy "\Global_pro_hmm_def39\pro_39_m1_s2" "\Init_pro_hmm\sic_i"
copy "\Global_pro_hmm_def39\pro_39_m1_s2" "\Init_pro_hmm\sic_u"
copy "\Global_pro_hmm_def39\pro_39_m1_s2" "\Init_pro_hmm\sic_3u"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\u"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\ue"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\ues1"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\uesn"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\ueng"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\uen"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\uengu"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\uengue"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\ai"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\a"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\an"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\anu"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\angue"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\anguee"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\ue"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\ueu"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\uei"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\e"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\eng"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\engu"
copy "\Global_pro_hmm_def39\pro_39_m1_s4" "\Init_pro_hmm\engue"
```
Step04_HHed_ModelMixSplit.bat

- **Function:**
  - Split the single Gaussian distribution of each HMM state into \( n \) mixture of Gaussian distributions, while the mixture number is set with respect to size of the training data for each model.

- **Command:**
  - `HHEd -C ..\Config\ConfigHHEd.fig -d ..\Init_pro_hmm -M ..\Init_pro_hmm_mixture ..\Script\HEdCmd.scp ..\Script\rcdmodel_sil`

  - `dir of the proto HMMs`
  - `dir of the resultant HMMs`

  - **HHEd configuration**
  - **HMM model list**

  - **List of the models to be trained**

```
NU 8 ( sic_a.state[2-3].mix )
HU 4 ( sic_o.state[2-3].mix )
HU 16 ( sic_e.state[2-3].mix )
NU 32 ( sic_i.state[2-3].mix )
NU 16 ( sic_u.state[2-3].mix )
NU 16 ( sic_iu.state[2-3].mix )
NU 16 ( b_u.state[2-4].mix )
NU 16 ( b_a.state[2-4].mix )
NU 8 ( b_e.state[2-4].mix )
NU 8 ( b_e.state[2-4].mix )
NU 8 ( b_e.state[2-4].mix )
NU 8 ( b_o.state[2-4].mix )
NU 16 ( ch_e.state[2-4].mix )
NU 8 ( b_iu.state[2-4].mix )
NU 8 ( ch_o.state[2-4].mix )
NU 8 ( ch_e.state[2-4].mix )
NU 4 ( ch_i.state[2-4].mix )
NU 4 ( ch_o.state[2-4].mix )
NU 8 ( ch_u.state[2-4].mix )
```
Step05_HERest_Train.bat

• Function:
  – Perform HMM model training
  – Baum-Whelch (EM) training performed over each training utterance using the composite model

• Commands:

```
HERest -T 00001  -t 100 -v 0.000000001 -C ..\Config\Config.fig -L ..\label -X rec -d ..\Init_pro_hmm_mixture -s statics -M ..\Rest_E -S ..\script\HERest.scp ..\Script\rcdmodel_sil
```

Dir to look the corresponding label files
Dir of initial models
List of the coefficient files of the training data
List of the models to be trained
Pruning threshold of the forward-backward procedures
cut-off value of the variance

• You can repeat the above command multiple times, e.g., 30 time, to achieve a better set of HMM models
Step05_HERest_Train.bat (cont.)

A label file of a training utterance

```
0   1100000 sil
1100000 2800000 b_o
2800000 3600000 o
3600000 4800000 l_u
4800000 6500000 uan
6500000 7300000 f_a
7300000 8800000 en
8800000 1020000 j_e
10200000 11400000 eng
11400000 15900000 sil
```

Boundary information of the segments of HMM models (will not be used for HERest)

List of the models to be trained

```
a
ai
an
ang
au
b_a
b_e
b_ce
b_i
b_o
b_u
ch_a
ch_e
ch_empt
ch_o
ch_u
chi_i
chi_iu
d_a
d_e
d_e
d_e
d_i
d_o
d_u
e
```
Step06_HCopyTest.bat

- **Function:**
  - Generate MFCC feature files for the testing speech utterances

- **Command**
  
  HCOPY -T 00001 -C ..\Config\Config.fig -S ..\script\HCopy_Test.scp

The detailed explanation can be referred to:

**Step01_HCopy_Train.bat**
Step07_HVite_Recognition.bat

• Function:
  – Perform free-syllable decoding on the testing utterances

• Command

HVite -C ..\Config\Config.fig -T 1 -X ..\script\netparsed -o SW
  -w ..\script\SYL_WORD_NET.netparsed -d ..\Rest_E -l ..\Syllable_Test_HTK
  -S ..\script\HVite_Test.scp ..\script\SYLLABLE_DIC ..\script\rcdmodel_sil

The extension file name for the search/recognition network

Set the output label files format: no score information, and no word information

A list of the testing utterances

The search/recognition network generated by HParse command

A list to lookup the constituent INITIAL/FINAL models for the composite syllable models

Dir to load the HMM models

Dir to save the output label files
The search/recognition network before performing HParse command

A list to lookup the constituent INITIAL/FINAL models for the composite syllable models

```
{<
  j_empt-empt1 | 
  ch_empt-empt1 | 
  sh_empt-empt1 | 
  r_empt-empt1 |
  .
  .
  .
  ji_iu-ue |
  k_u-uo[] |
  sil ( )
}
```

Regular expression

```
HParse SYL_WORD_NET SYL_WORD_NET.netparsed
```

The search/recognition network generated by HParse command

```
VERSION=1.0
N=407  L=1213
I=0  W=!NULL
I=1  W=!NULL
I=2  W=j_empt-empt1
I=3  W=!NULL
I=4  W=ch_empt-empt1
I=5  W=sh_empt-empt1
I=6  W=r_empt-empt1
I=7  W=tz_empt-empt2
I=8  W=tz_empt-empt2
I=9  W=s_empt-empt2
I=10  W=sic_a-a
I=11  W=j_a-a
```

The search/recognition network generated by HParse command

```
```

A list to lookup the constituent INITIAL/FINAL models for the composite syllable models

```
j_empt-empt1  j_empt  empt1
    ch_empt-empt1  ch_empt  empt1
    .
    .
    .
    sic_a-a  sic_a a
    j_a-a  j_a a
    ch_a-a  ch_a a
    sh_a-a  sh_a a
    tz_a-a  tz_a a
    .
    .
    sil  sil
```
Step08_HResults_Test.bat

• Function:
  – Analyze the recognition performance

• Command

HResults -C ..\Config\Config.fig -T 00020 -X rec -e ??? sil -L ..\Syllable
-S ..\script\Hresults_rec600.scp ..\script\SYLLABLE_DIC

The extension file name for the label files
ignore the silence label “sil”

A list of the label files generated by the recognition process

Dir lookup the reference label files

------------------------------------------- HTK Results Analysis -------------------------------------------
Date: Sat Nov 22 10:02:13 2003
Ref : ../Syllable
Rec : ../Syllable\Test\HTK\bruce-b07-001.rec
     ../Syllable\Test\HTK\bruce-b07-002.rec
     ../Syllable\Test\HTK\bruce-b07-003.rec
     ../Syllable\Test\HTK\bruce-b07-004.rec
     ../Syllable\Test\HTK\bruce-b07-005.rec
     ../Syllable\Test\HTK\bruce-b07-006.rec
     ../Syllable\Test\HTK\bruce-b07-007.rec
     ../Syllable\Test\HTK\bruce-b07-008.rec
     ../Syllable\Test\HTK\bruce-b07-009.rec
     ../Syllable\Test\HTK\bruce-b07-010.rec
     ../Syllable\Test\HTK\bruce-b07-011.rec
     ../Syllable\Test\HTK\bruce-b07-012.rec
     ../Syllable\Test\HTK\bruce-b07-013.rec
     ../Syllable\Test\HTK\bruce-b07-014.rec
     ../Syllable\Test\HTK\bruce-b07-015.rec
     ../Syllable\Test\HTK\bruce-b07-016.rec
     ../Syllable\Test\HTK\bruce-b07-017.rec
     ../Syllable\Test\HTK\bruce-b07-018.rec
     ../Syllable\Test\HTK\bruce-b07-019.rec
     ../Syllable\Test\HTK\bruce-b07-020.rec

------------------------------------------- Overall Results -------------------------------------------
SEN: %Correct=11.33 [H=58, S=532, N=6001]
WORD: %Correct=69.49. Acc=64.16 [H=2685, D=47, S=1132, I=206, N=3864]

-------------------------------------------
Step09_BatchMFCC_Def39.bat

• Also, you can train the HMM models in another way

  Hinit \rightarrow (HHEd) \rightarrow HRest \rightarrow HERest

• For detailed information, please refer to the previous slides or the HTK manual

• You can compare the recognition performance by running

  Step02~Step05

or

  Step09 alone