Introduction to HTK Toolkit

Berlin Chen 2005

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
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.)

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 MFCC-based Front-End Processor](image)

<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.

```
~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 0.6 0.4 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 **HINIT** and **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}, \phi_{14}\}
Training Phase (cont.)

Prototype HMM

Uniform Segmentation

Initialise Parameters

Viterbi Segmentation

Update HMM Parameters

Converged?

Yes

Initialised HMM

No

Input HMM Def [lh.1]

HInit

Output HMM Def [lh.2]

File Processing in HInit

HInit Operation

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

000003 .ch
000001 .ch
000055 .ch
000032 .ch
000023 .ch
021091 .ih
092134 .t
135182 .ih
Training Phase (cont.)

Initial HMM

Forward/Backward Algorithm

Update HMM Parameters

Converged?

No

Yes

Estimated HMM

HRest Operation
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.

  ![Context-Dependent Modeling Diagram]

  – **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)
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;}

  ( 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-START</td>
<td>[] sil</td>
</tr>
<tr>
<td>SENT-END</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)

```mlf
#!MLF#!
"*/S0001.lab"
ONE
VALIDATED
ACTS
OF
SCHOOL
DISTRICTS
.
"*/S0002.lab"
TWO
OTHER
CASES
ALSO
WERE
UNDER
ADVISEMENT
.
"*/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 Liftering Setting
- Number of output cepstral coefficients

in 100 nanosecond unit

10ms

25ms
Coding The Data (cont.)

HCopy -T 1 -C config -S codetr.scp
Training
Tee Model
Recognition

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

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

\[
\text{Percent Accuracy} = \frac{N - D - S - I}{N} \times 100\%
\]
Homework 3: 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**

  **Specify the detailed configuration for feature extraction**

  **Specify the pcm and coefficient files and their respective directories**

  ```
  # Coding parameters
  SOURCEFORMAT=ALEN
  HEADERSIZE=0
  SOURCERATE=825
  TARGETRATE=16000
  TARGETKIND=MFCC_Z_E_D_A
  TARGETRATE=100000.0
  #frameshift 1um
e 2 bytes per sample
  E
  
  # savecompressed=F
  SAVewithCRC=F
  WINDOWSIZE=32000.0
  # framesize = 32ms
  USEHAMMING=T

  # Preemphasis
  PREEMCOEF=0.97
  NUNCHANS=26
  CEPLIFTER=22
  NUMCEPS=12
  ENORMALIZE=T

  NATURALREADORDER=TRUE
  NATURALWRITEORDER=TRUE
  ```
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 resultant prototype HMM (with the global mean and variance setting)
- mean will be updated
- The prototype 1-state HMM with zero mean and variance of value 1
- a list of coefficient files

**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.
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_e
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
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```

```bash
..\Global_pro hmm def39\pro_39_m1_s4
```
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\HHEd.fig -d ..\Init_pro_hmm -M ..\Init_pro_hmm_mixture ..\Script\HHEdCmd.scp ..\Script\rcdmodel_sil`

<table>
<thead>
<tr>
<th>HHEd configuration</th>
<th>HMM model list</th>
<th>List of the models to be trained</th>
</tr>
</thead>
<tbody>
<tr>
<td>NU 8 ( sic_a.state[2-3].mix )</td>
<td>NU 4 ( sic_o.state[2-3].mix )</td>
<td>a</td>
</tr>
<tr>
<td>NU 4 ( sic_u.state[2-3].mix )</td>
<td>NU 16 ( sic_a.state[2-3].mix )</td>
<td>a1</td>
</tr>
<tr>
<td>NU 16 ( sic_o.state[2-3].mix )</td>
<td>NU 32 ( sic_i.state[2-3].mix )</td>
<td>a_n</td>
</tr>
<tr>
<td>NU 16 ( sic_u.state[2-3].mix )</td>
<td>NU 16 ( sic_iu.state[2-3].mix )</td>
<td>ang</td>
</tr>
<tr>
<td>NU 16 ( b_u.state[2-3].mix )</td>
<td>NU 16 ( b_a.state[2-3].mix )</td>
<td>au</td>
</tr>
<tr>
<td>NU 16 ( b_o.state[2-3].mix )</td>
<td>NU 16 ( b_a.state[2-4].mix )</td>
<td>b_o</td>
</tr>
<tr>
<td>NU 3 ( b_e.state[2-4].mix )</td>
<td>NU 3 ( b_e.state[2-4].mix )</td>
<td>b_e</td>
</tr>
<tr>
<td>NU 3 ( b_ee.state[2-4].mix )</td>
<td>NU 3 ( b_ee.state[2-4].mix )</td>
<td>b_ee</td>
</tr>
<tr>
<td>NU 3 ( b_i.state[2-4].mix )</td>
<td>NU 3 ( b_i.state[2-4].mix )</td>
<td>b_i</td>
</tr>
<tr>
<td>NU 3 ( b_o.state[2-4].mix )</td>
<td>NU 3 ( b_o.state[2-4].mix )</td>
<td>b_o</td>
</tr>
<tr>
<td>NU 16 ( chi_i.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>b_u</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>ch_e</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>ch_o</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>ch_u</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>ch_i</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>chi_i</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>d_a</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>d_b</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>d_e</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>d_i</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>d_o</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>d_u</td>
</tr>
<tr>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>NU 16 ( chi_iu.state[2-4].mix )</td>
<td>e_</td>
</tr>
</tbody>
</table>
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

  ......  

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

• 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 11000000 sil
11000000 28000000 b_o
28000000 36000000 o
36000000 48000000 l_u
48000000 65000000 uan
65000000 73000000 f_a
73000000 88000000 en
88000000 10200000 j_e
10200000 11400000 eng
11400000 15900000 sil
```

List of the models to be trained

```
a
ai
an
ang
au
b_a
b_e
b_ee
b_i
b_o
b_u
ch_a
ch_e
chempt
ch_o
ch_u
chi_l
chi_iu
d_a
d_e
d_ee
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**
  
  ```bash
  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

The search/recognition network generated by HParse command
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
  ```

  A list of the label files generated by the recognition process

  Dir lookup the reference label files

  ```
  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
  ```

  The extension file name for the label files
Step09_BatchMFCC_Def39.bat

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

\[
\text{Hinit} \rightarrow (\text{HHEd}) \rightarrow \text{HRest} \rightarrow \text{HERest}
\]

• 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