Disclaimer

- The material provided in this document is not my original work and is a summary of some one else’s work(s).
- A simple Google search of the title of the document will direct you to the original source of the material.
- I do not guarantee the accuracy, completeness, timeliness, validity, non-omission, merchantability or fitness of the contents of this document for any particular purpose.
- Downloaded from najeebkhan.github.io

Chapter # 2: Overview

HMM TOOL KIT HTK

Outline

- HTK Software Architecture
- Generic Properties of HTK Tool
- The Toolkit

HTK Software Architecture

- Much of the functionality of HTK is built into the library modules
- Hshell: User input/output and interaction with the operating system
- Hmem: Memory Management
- Hmath: Math support is provided
- HSigP: signal processing operations needed for speech analysis
- Hlabel: provides the interface for label files
- HLM: language model files
- Hnet: networks and lattices

HTK Software Architecture

- Hdict: dictionaries
- HVQ: VQ codebooks
- Hmodel: HMM definitions
- Hwave: speech input and output at the waveform level
- HParm: speech input and output at the parameterized level
- HTrain: support for the HTK training tools
- HAdapt: provides support for the various HTK adaptation tools
- HRec: the main recognition processing functions

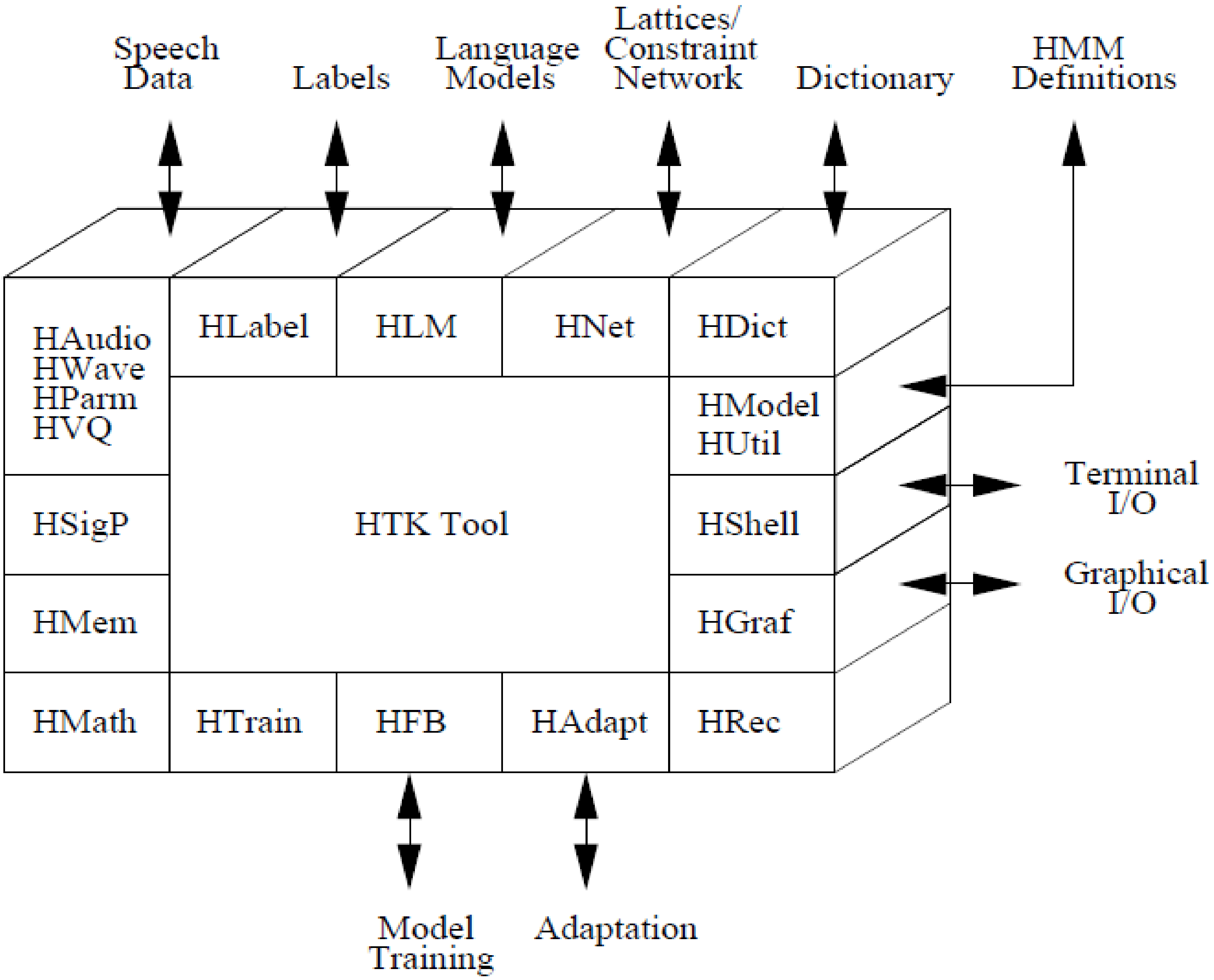


Fig. 2.1 Software Architecture

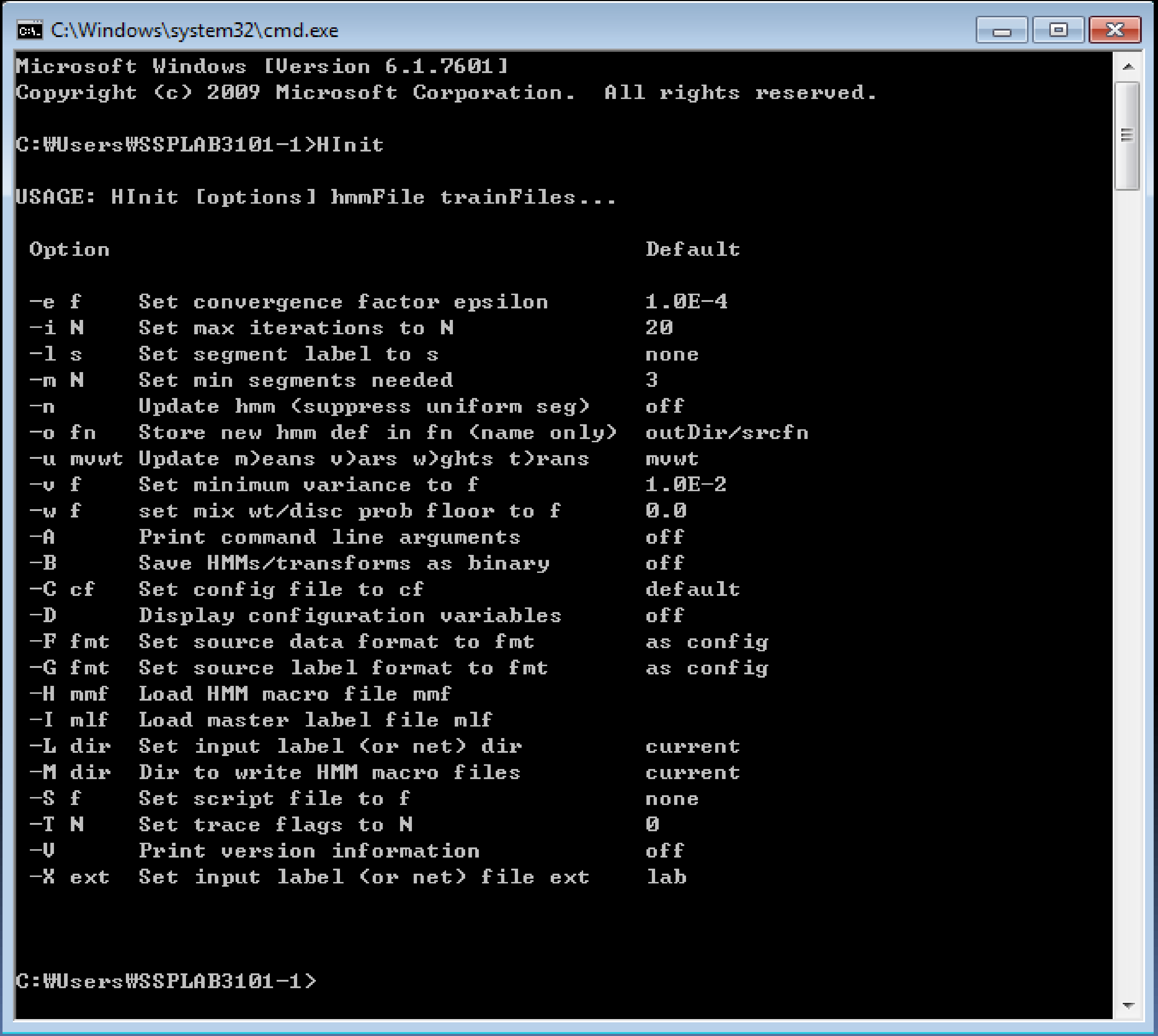
Generic Properties of HTK Tool

- HTK tools are designed to run with a traditional command-line style interface
- Each tool has a number of required arguments plus optional arguments

```
HFoo -T 1 -f 34.3 -a -s myfile file1 file2
```

- In addition to command line arguments, the operation of a tool can be controlled by parameters stored in a configuration file

```
HFoo -C config -f 34.3 -a -s myfile file1 file2
```



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:₩Users₩SSPLAB3101-1>HInit

USAGE: HInit [options] hmmFile trainFiles...

 Option                                          Default

 -e f     Set convergence factor epsilon         1.0E-4
 -i N     Set max iterations to N                20
 -l s     Set segment label to s                 none
 -m N     Set min segments needed                3
 -n       Update hmm (suppress uniform seg)      off
 -o fn    Store new hmm def in fn (name only)    outDir/srcfn
 -u mvwt  Update m)eans v)ars w)ghts t)rans      mvwt
 -v f     Set minimum variance to f              1.0E-2
 -w f     set mix wt/disc prob floor to f        0.0
 -A       Print command line arguments           off
 -B       Save HMMs/transforms as binary         off
 -C cf    Set config file to cf                  default
 -D       Display configuration variables        off
 -F fmt   Set source data format to fmt          as config
 -G fmt   Set source label format to fmt         as config
 -H mmf   Load HMM macro file mmf
 -I mlf   Load master label file mlf
 -L dir   Set input label (or net) dir           current
 -M dir   Dir to write HMM macro files           current
 -S f     Set script file to f                   none
 -T N     Set trace flags to N                   0
 -V       Print version information              off
 -X ext   Set input label (or net) file ext      lab



C:₩Users₩SSPLAB3101-1>
```

The Toolkit

- There are 4 main phases involved in building a sub-word based continuous speech recognizer
  - Data preparation
  - Training
  - Testing
  - Analysis

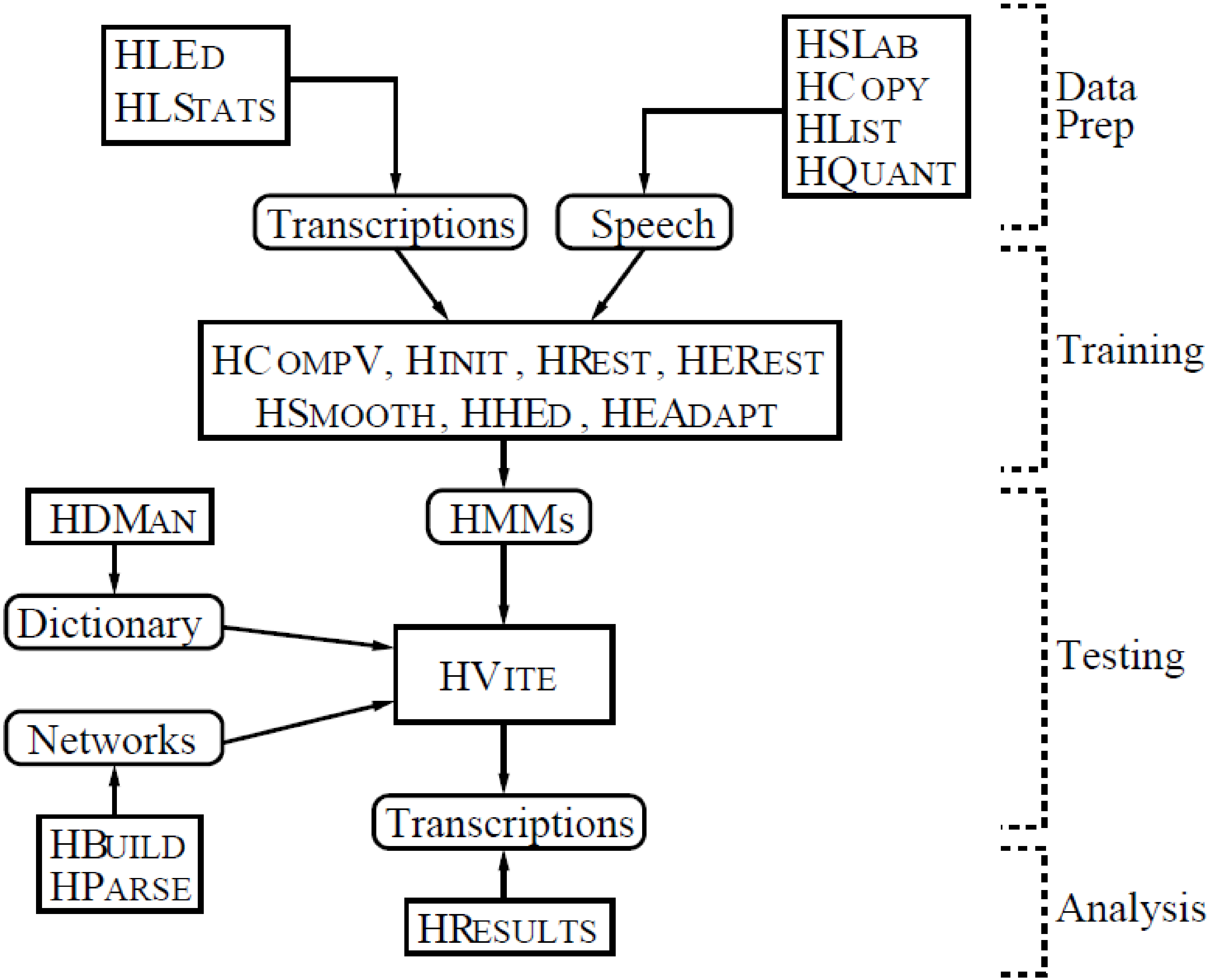


Fig. 2.2 HTK Processing Stages

Data preparation

- In order to build a set of HMMs, a set of speech data files and their associated transcriptions are required
- Before the speech can be used in training, it must be converted into the appropriate parametric form and any associated transcriptions must be converted to have the correct format and use the required phone or word labels
- It is usually better to parameterize the data just once and HCopy for copying
- Transcriptions will also need preparing (e.g. because of differences in the phone sets used)
- HLEd is a script-driven label editor which is designed to make the required transformations to label files
- HQuant can be used to build a VQ codebook

Training

- Simple text files can be used to define the topology required for each HMM by writing a prototype definition
- The purpose of the prototype definition is only to specify the overall characteristics and topology of the HMM

Tra

- Si fine the
  to MM by
  wr
- Th nition is
  or teristics
  an

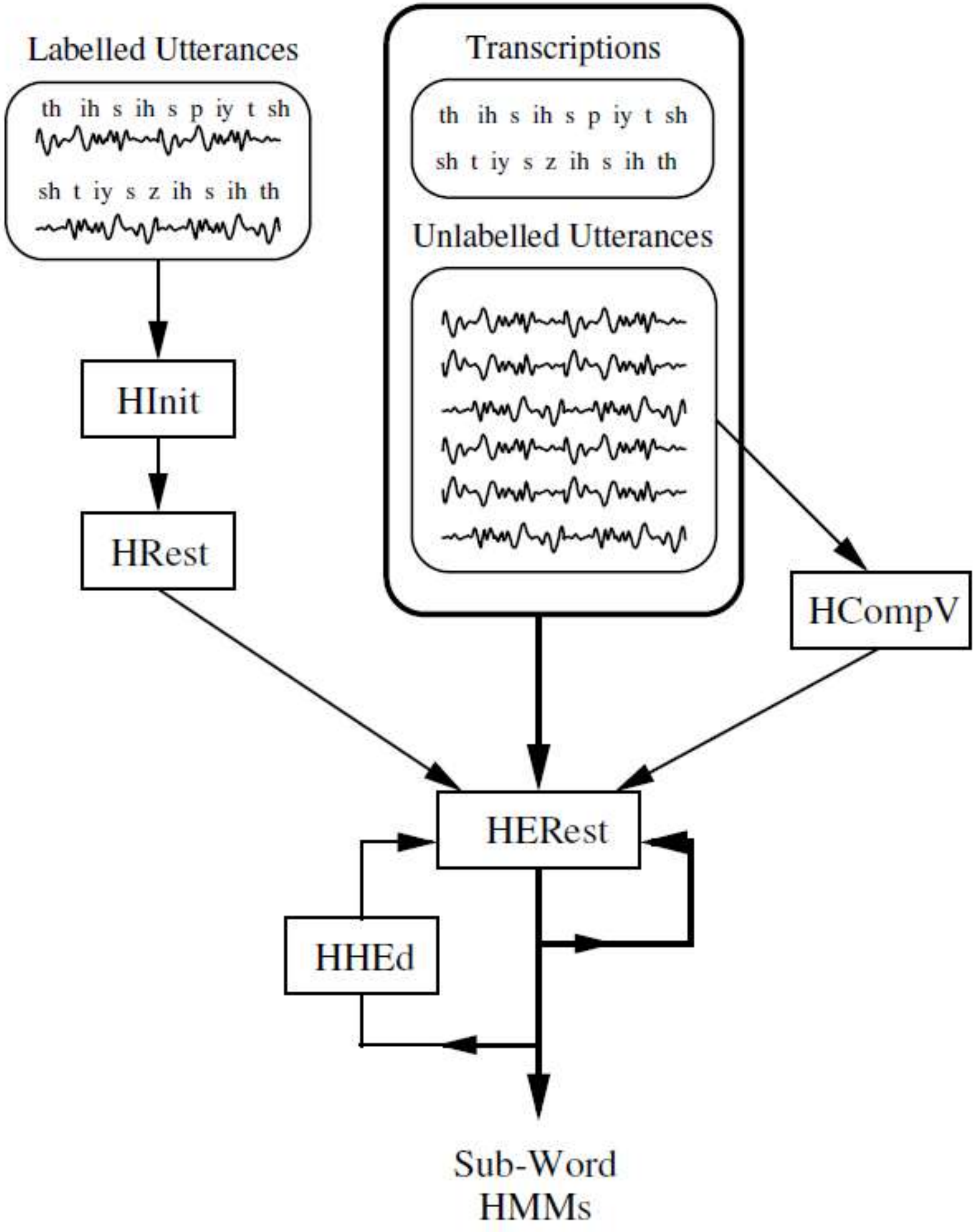


Fig. 2.3 Training Sub-word HMMs

Training

- The philosophy of system construction in HTK is that HMMs should be refined incrementally
- Thus, 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

Recognition Tools

- HVite
  - HTK provides a recognition tool called HVite which uses the token passing algorithm to perform Viterbi-based speech recognition
  - HVite takes as input a network describing the allowable word sequences, a dictionary defining how each word is pronounced and a set of HMMs
  - It operates by converting the word network to a phone network and then attaching the appropriate HMM definition to each phone instance
  - Recognition can then be performed on either a list of stored speech files or on direct audio input

Recognition Tools

- HLRescore
  - HLRescore is a tool for manipulating lattices
  - it reads lattices in standard lattice format and applies one of the following operations on them:
  - finding 1-best path through lattice
  - expanding lattices with new language model
  - converting lattices to equivalent word networks
  - calculating various lattice statistics
  - pruning lattice using forward-backward scores

Recognition Tools

- HDecode
  - Similar to HVite, HDecode transcribes speech files using a HMM model set and a dictionary (vocabulary)
  - The best transcription hypothesis will be generated in the Master Label File (MLF) format

Analysis Tools

- Once the HMM-based recognizer has been built, it is necessary to evaluate its performance
- This is usually done by using it to transcribe some pre-recorded test sentences and match the recognizer output with the correct reference transcriptions (HResults)
- Hresults uses dynamic programming to align the two transcriptions and then count substitution, deletion and insertion errors
- It can also compute Figure of Merit (FOM) scores and Receiver Operating Curve (ROC) information

Thank You