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.
-

Chapter # 10: HMM System Refinement

HMM TOOL KIT HTK

Outline

- Introduction
- Using HHEd
- Constructing Context-Dependent Models
- Parameter Tying and Item Lists
- Data-Driven Clustering
- Tree-Based Clustering
- Mixture Incrementing
- Regression Class Tree Construction
- Miscellaneous Operations

Introduction

- The principle types of manipulation that can be performed by HHEd are
  - HMM cloning to form context-dependent model sets
  - Generalized parameter tying
  - Data driven and decision tree based clustering
  - Mixture component splitting
  - Adding/removing state transitions
  - Stream splitting, resizing and recasting

Using HHEd

```
HHEd -H MMF1 -H MMF2 ... -M newdir cmds.hed hmmlist
```

```
    ....
    UF smacs
    # commands to generate state macros
    ....
    UF vmacs
    # commands to generate variance macros
    ....
```

```
HHEd -H MMF1 -H MMF2 ... -w newMMF cmds.hed hmmlist
```

```
HERest -H hmm1/MMF -M hmmx -s stats hmmlist train1 train2 ....
HHEd -H hmm1/MMF -M hmm2 cmds.hed hmmlist
```

Constructing Context-Dependent Models

- The first stage of model refinement is usually to convert a set of initialized and trained context independent monophone HMMs to a set of context dependent models
- l-p+r
- To make a set of context dependent phone models, it is only necessary to construct a HMM list, called say cdlist
- CL cdlist : The effect of this command is that for each model l-p+r in cdlist it makes a copy of the monophone p

Constructing Context-Dependent Models

- To train the context dependent HMMs the training data transcriptions must be converted to use context-dependent labels
- If the HLEd TC command is used then the –n option can be used to generate the required list of context dependent HMMs automatically
- Word boundary symbols can be specified using WB command if inter-word triphone are to be used

Parameter Tying and Item Lists

- When two or more parameter sets are tied, the same set of parameter values are shared by all the owners of the tied set
- When the values of a tied parameter set are re-estimated, all of the data which would have been used to estimate each individual untied parameter are effectively pooled leading to more robust parameter estimation
- Tying is performed by executing HHEd commands

Parameter Tying and Item Lists

The basic HHEd command for tying a set of parameters

```
TI macroname itemlist
```

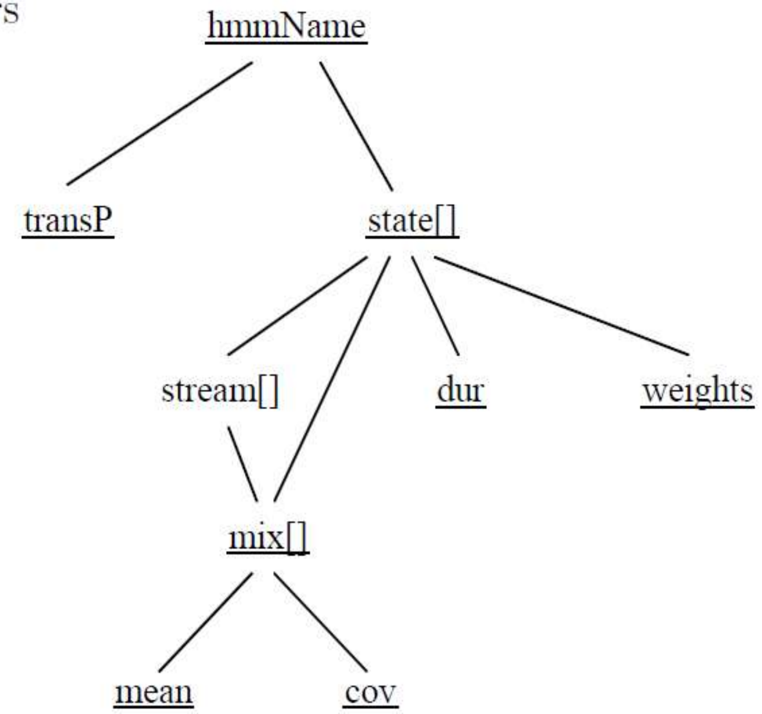
Itemlist

```
{ aa.state[2],aa.state[3],aa.state[4] }

{ aa.state[2-4] }

{ (aa+*,iy+*,eh+*).state[2-4] }

{ *.state[2-4].stream[1].mix[1,3].cov }
```



Parameter Tying and Item Lists

```
CL cdlist
TI T_ah {*-ah+*.transP}
TI T_eh {*-eh+*.transP}
TI T_ae {*-ae+*.transP}
TI T_ih {*-ih+*.transP}
 ... etc
```

Grand Variance HMM system can be generated by

```
TI "gvar" { *.state[2-4].mix[1].cov }

TI "silst" { sp.state[2], sil.state[3] }

UT {*-iy+*.transP}
```

Parameter Tying and Item Lists

- When states are tied, the state with the broadest variances and as few as possible zero mixture component weights is selected from the pool and used as the representative
- When mean vectors are tied, the average of all the mean vectors in the pool is used
- When variances are tied, the largest variance in the pool is used
- In all other cases, the last item in the tie-list is arbitrarily chosen as representative

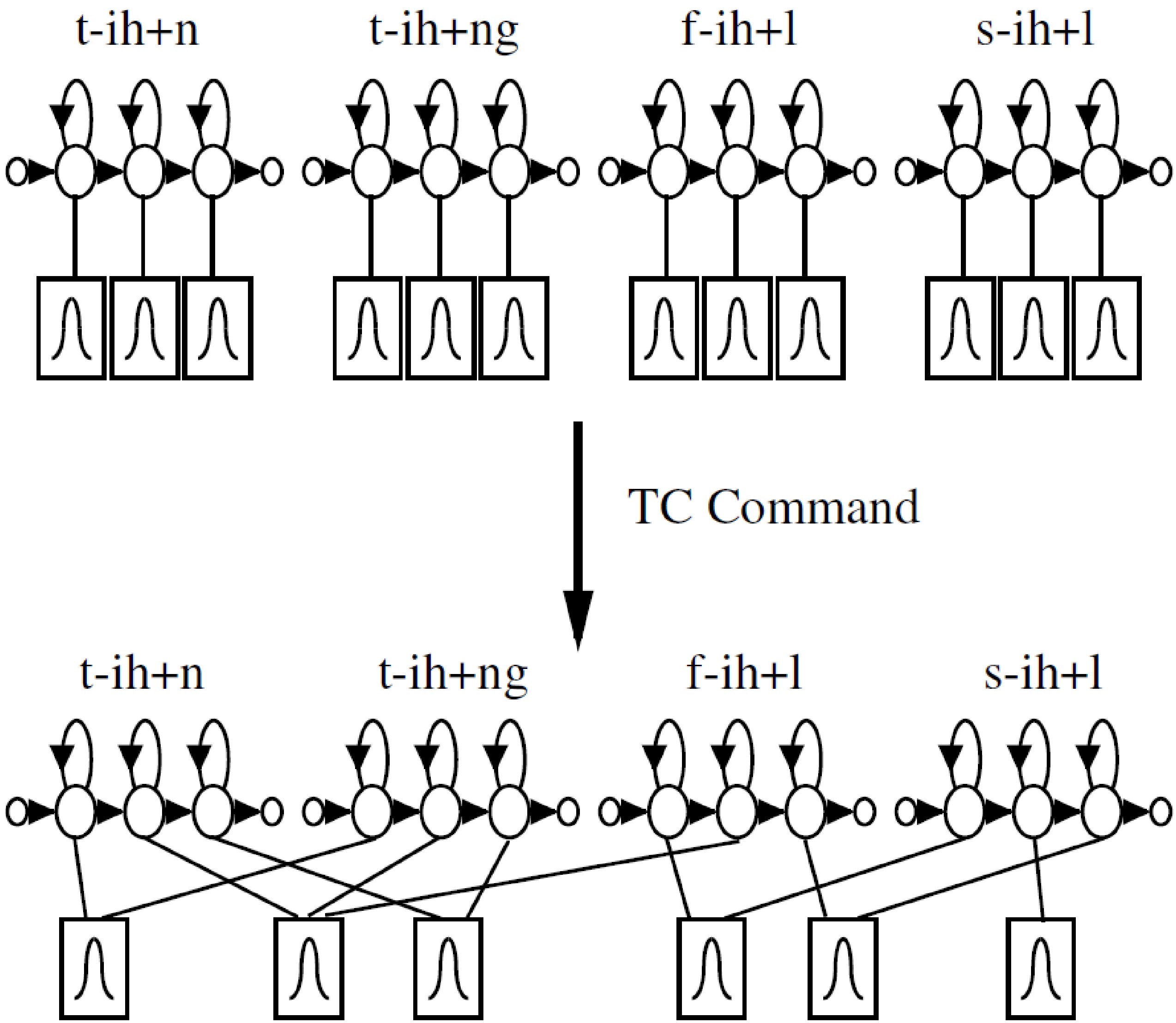
Data-Driven Clustering

- Triphone construction involves
  - Cloning all monophones
  - Re-estimating them using data for which monophone labels have been replaced by triphone labels
- This will lead to a very large set of models, and relatively little training data for each model

```
TI "iyS3" {*-iy+*.state[3]}
TI "ihS3" {*-ih+*.state[3]}
TI "ehS3" {*-eh+*.state[3]}
 .... etc
```

Data-Driven Clustering

- A much better approach is to use clustering to decide which states to tie
- HHEd provides two mechanisms for this
  - Data-driven clustering
  - Decision tree-based approach
- Data-driven clustering is performed by the TC and NC commands
- Initially all states are placed in individual clusters
- The pair of clusters which when combined would form the smallest resultant cluster are merged
- This process repeats until either the size of the largest cluster reaches the threshold set by the TC command
- Or the total number of clusters has fallen to that specified by the NC command




```
TC 100.0 "ihS2" {*-ih+*.state[2]}
TC 100.0 "ihS3" {*-ih+*.state[3]}
TC 100.0 "ihS4" {*-ih+*.state[4]}
TC 100.0 "ihS2" {(*-ih,ih+*,*-ih+*).state[2]}
```

Data-Driven Clustering

- Drawback: Outlier states will tend to form their own singleton clusters for which there is then insufficient data to properly train
- Repeatedly finding the cluster with the smallest total occupation count and merging it with its nearest neighbor can be done as follows

```
RO thresh "statsfile"
```

- Compact representation for identical hmms

```
CO newList
```

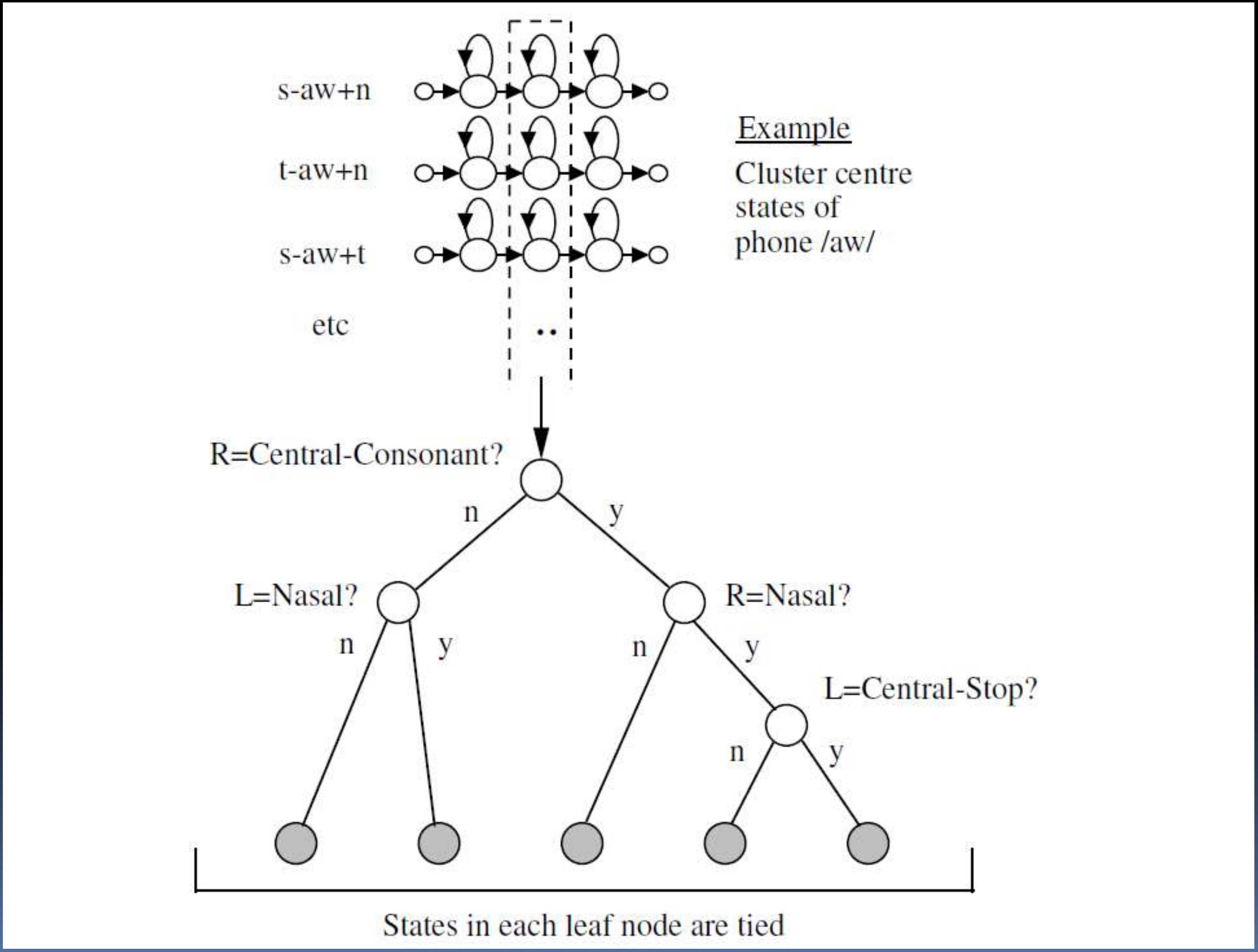
Tree-Based Clustering

- Data-driven clustering procedure described above is that it does not deal with triphones for which there are no examples in the training data
- Decision tree-based clustering offers a solution to the unseen triphone problem

```
TB thresh macroname itemlist
```

- TC uses a distance metric between states whereas TB uses a log likelihood criterion
- TC supports any type of output distribution whereas TB only supports single-Gaussian continuous density output distributions

Tree-Based Clustering



Tree-Based Clustering

- The question at each node is chosen to (locally) maximize the likelihood of the training data given the final set of state tyings
- Before any tree building can take place, all of the possible phonetic questions must be loaded into HHEd using QS commands

```
QS "L_Nasal" { ng-*,n-*,m-* }
```

- It is possible to calculate the log likelihood of the training data given any pool of states
- This can also be done without reference to the training data itself since for single Gaussian distributions the means, variances and state occupation counts form sufficient statistics???

Tree-Based Clustering

- The question at each node is chosen to (locally) maximize the likelihood of the training data given the final set of state tyings

```
TB 350.00 aw_s3 {}
 Tree based clustering
  Start   aw[3] : 28  have  LogL=-86.899 occ=864.2
  Via     aw[3] : 5   gives LogL=-84.421 occ=864.2
  End     aw[3] : 5   gives LogL=-84.421 occ=864.2
TB: Stats 28->5 [17.9%]  { 4537->285 [6.3%] total }
```

- This can also be done without reference to the training data itself since for single Gaussian distributions the means, variances and state occupation counts form sufficient statistics???

Tree-Based Clustering

```
AU hmmlist
```

- It scans the given hmmlist and synthesizes the models
- This is done by descending the previously constructed trees for that phone and answering the questions at each node based on the new unseen context
- When each leaf node is reached, the state representing that cluster is used for the corresponding state in the unseen triphone

Mixture Incrementing

- The conversion from single Gaussian HMMs to multiple mixture component HMMs is usually one of the final steps in building a system

```
MU n itemList
MU 3 {aa.state[2].mix}
MU 3 {*.state[2-4].mix}
```

Thank You