Acoustic Modeling for Speech Recognition

Berlin Chen 2004

References:

1. X. Huang et. al., Spoken Language Processing, Chapter 8
2. The HTK Book (for HTK Version 3.2)

Introduction

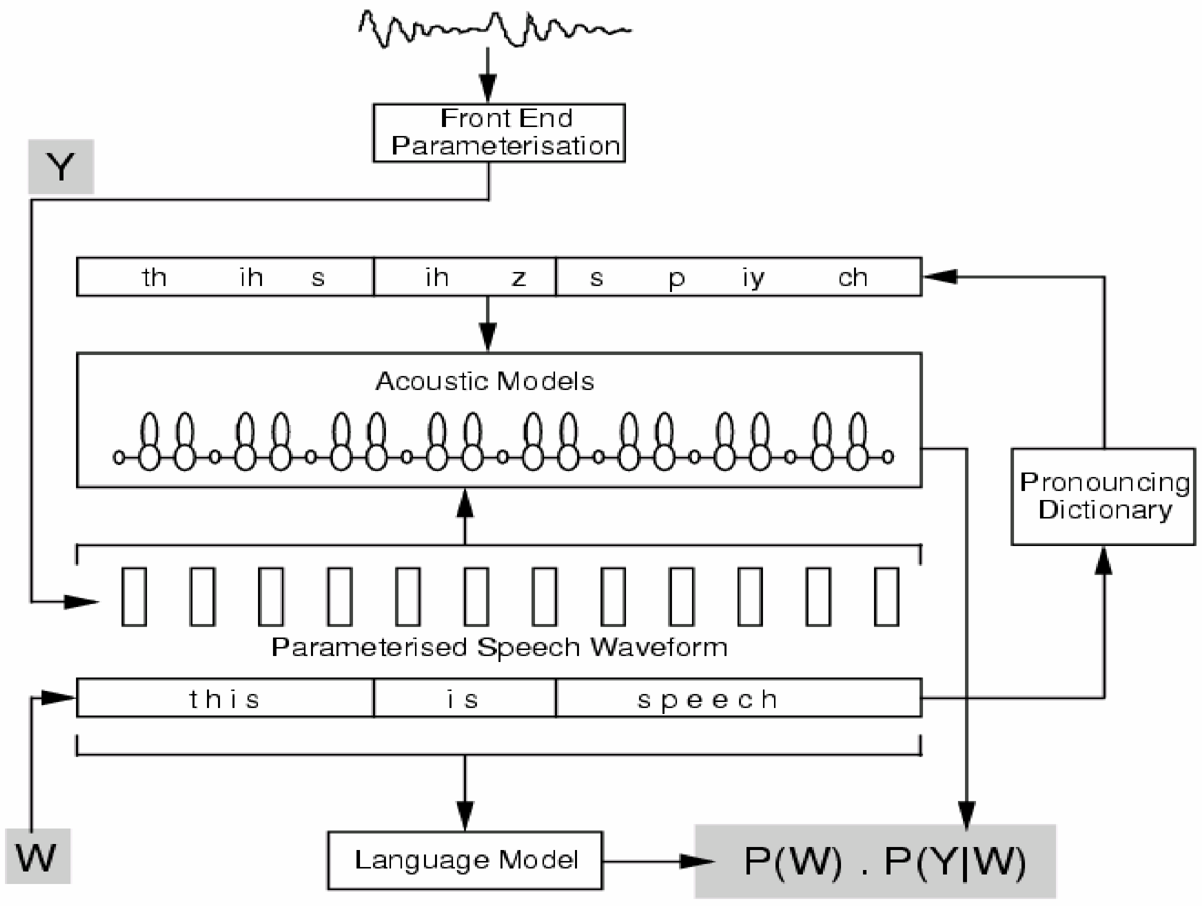
- For the given acoustic observation $\boldsymbol{X} = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, the goal of speech recognition is to find out the corresponding word sequence $\boldsymbol{W} = w_1, w_2, \ldots, w_m$ that has the maximum posterior probability $P(\boldsymbol{W} | \boldsymbol{X})$

$$
\begin{aligned}
\hat{\boldsymbol{W}} &= \arg\max_{\boldsymbol{W}} P(\boldsymbol{W} | \boldsymbol{X}) \\
&= \arg\max_{\boldsymbol{W}} \frac{P(\boldsymbol{W}) P(\boldsymbol{X} | \boldsymbol{W})}{P(\boldsymbol{X})} \\
&= \arg\max_{\boldsymbol{W}} \underbrace{P(\boldsymbol{W})}_{\text{Language Modeling}} \underbrace{P(\boldsymbol{X} | \boldsymbol{W})}_{\text{Acoustic Modeling}}
\end{aligned}
$$

$$
\boldsymbol{W} = w_1, w_2, \ldots w_i, \ldots, w_m \quad \text{where } w_i \in V : \{v_1, v_2, \ldots\ldots, v_N\}
$$

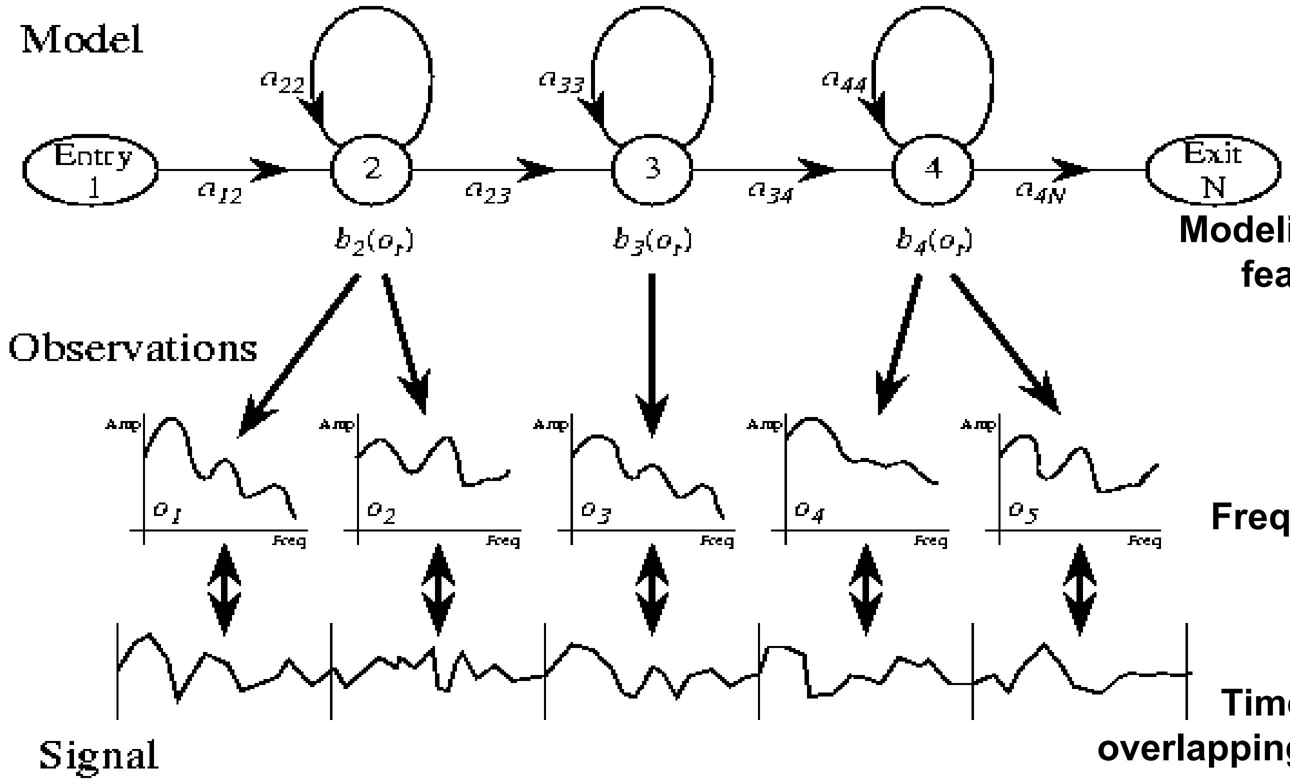
Possible variations: domain, topic, style, etc. and speaker, pronunciation, environment, context, etc. $\Rightarrow$ To be discussed later on !

Introduction (cont.)



Review: HMM Modeling

- Acoustic modeling using HMMs



Modeling the cepstral feature vectors

Frequency Domain

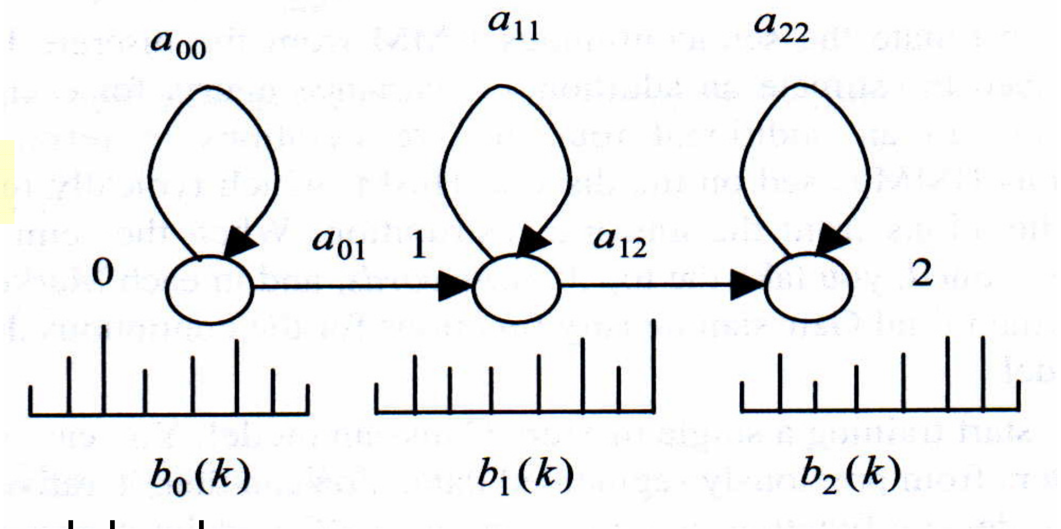
Time Domain overlapping speech frames

  – Three types of HMM state output probabilities are frequently used

Review: HMM Modeling (cont.)

1. Discrete HMM (DHMM): $b_j(\boldsymbol{v}_k)=P(\mathbf{o}_t=\boldsymbol{v}_k|s_t=j)$
   - The observations are quantized into a number of symbols
   - The symbols are normally generated by a vector quantizer
     - Each codeword is represented by a distinct symbol

A left-to-right HMM



   - With multiple codebooks

$$b_j(v_k)=\sum_{m=1}^{M} c_{jm}\, p(\boldsymbol{o}_t=\boldsymbol{v}_k | m, s_t=j) \qquad \sum_{m=1}^{M} c_{jm}=1$$

codebook index

Review: HMM Modeling (cont.)

2. Continuous HMM (CHMM)
   - The state observation distribution of HMM is modeled by multivariate Gaussian mixture density functions (*M* mixtures)

$$b_j(\boldsymbol{o}_t) = \sum_{m=1}^{M} c_{jm} b_{jm}(\boldsymbol{o}_t)$$

$$= \sum_{m=1}^{M} c_{jm} N(\boldsymbol{o}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) = \sum_{m=1}^{M} c_{jm} \left(\frac{1}{\left(\sqrt{2\pi}\right)^{L/2} \left|\boldsymbol{\Sigma}_{jm}\right|^{1/2}} \exp\left(-\frac{1}{2} (\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})^T \Sigma_{jm}^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}) \right) \right), \quad \sum_{m=1}^{M} c_{jm} = 1$$

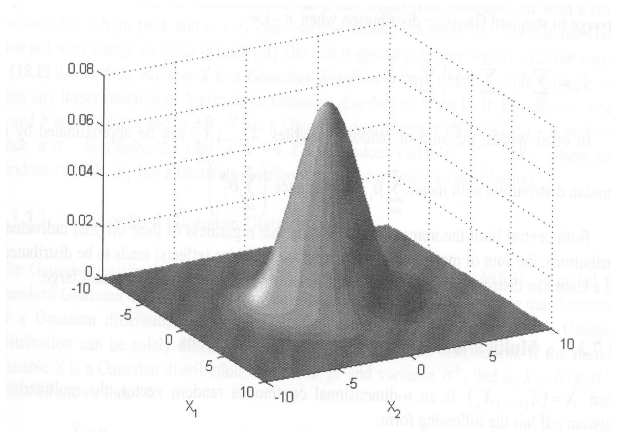


Figure 3.12 A two-dimensional multivariate Gaussian distribution with independent random variables x_1 and x_2 that have the same variance.

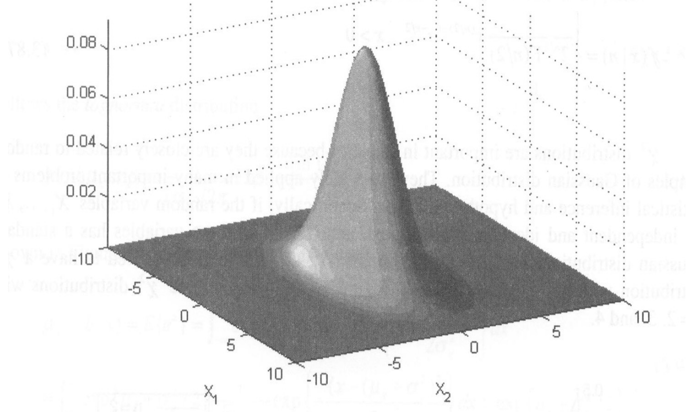


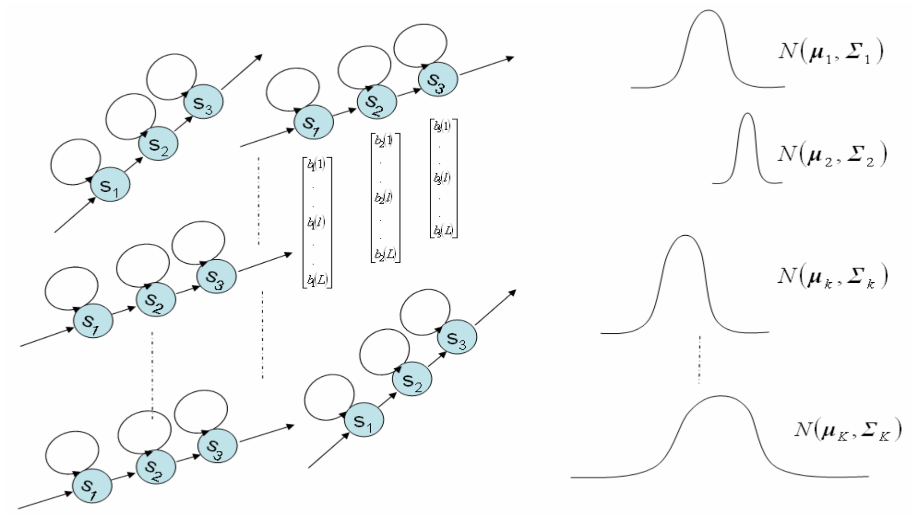
Figure 3.13 Another two-dimensional multivariate Gaussian distribution with independent random variable x_1 and x_2 which have different variances.

Review: HMM Modeling (cont.)

3. Semicontinuous or tied-mixture HMM (SCHMM)

– The HMM state mixture density functions are tied together across all the models to form a set of shared kernels (shared Gaussians)

$$b_j(\boldsymbol{o}) = \sum_{k=1}^{K} b_j(k) f(\boldsymbol{o} | v_k) = \sum_{k=1}^{K} b_j(k) N(\boldsymbol{o}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$



$N(\boldsymbol{\mu}_1, \Sigma_1)$
$N(\boldsymbol{\mu}_2, \Sigma_2)$
$N(\boldsymbol{\mu}_k, \Sigma_k)$
$N(\boldsymbol{\mu}_K, \Sigma_K)$
s_1
s_2
s_3
$b_1(1)$
$b_1(l)$
$b_1(L)$
$b_2(1)$
$b_2(l)$
$b_2(L)$
$b_3(1)$
$b_3(l)$
$b_3(L)$

– With multiple sets of shared Gaussians (or multiple codebooks)

$$b_j(\boldsymbol{o}) = \sum_{m=1}^{M} c_m \sum_{k=1}^{K} b_{jm}(k) f(\boldsymbol{o} | v_{m,k}) = \sum_{m=1}^{M} c_m \sum_{k=1}^{K} b_{jm}(k) N(\boldsymbol{o}; \boldsymbol{\mu}_{m,k}, \boldsymbol{\Sigma}_{m,k})$$

Review: HMM Modeling (cont.)

- Comparison of Recognition Performance

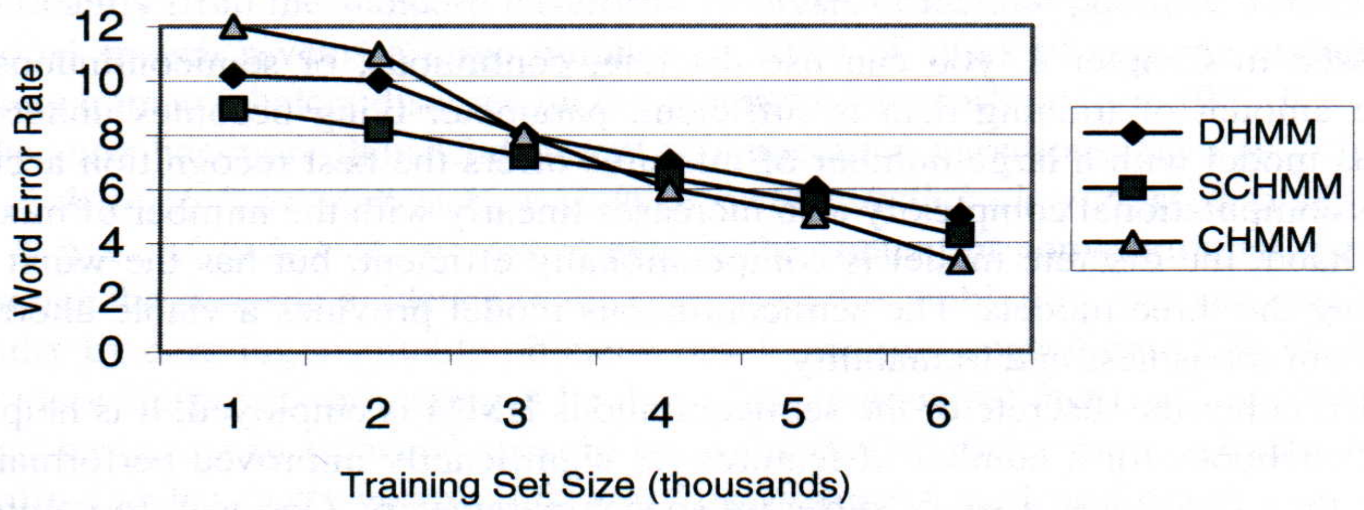


Figure 9.8 Continuous speaker-independent word recognition error rates of the discrete HMM (DHMM), SCHMM, and the continuous HMM (CHMM) with respect to the training set sizes (thousands of training sentences). Both the DHMM and SCHMM have multiple codebooks. The CHMM has 20 mixture diagonal Gaussian density functions.

Measures of ASR Performance

- Evaluating the performance of automatic speech recognition (ASR) systems is critical, and the Word Recognition Error Rate (WER) is one of the most important measures

- There are typically three types of word recognition errors
  - Substitution
    - An incorrect word was substituted for the correct word
  - Deletion
    - A correct word was omitted in the recognized sentence
  - Insertion
    - An extra word was added in the recognized sentence

- How to determine the minimum error rate?

Measures of ASR Performance (cont.)

- Calculate the WER by aligning the correct word string against the recognized word string
  - A maximum substring matching problem
  - Can be handled by dynamic programming

deleted

- Example:

  Correct : "the effect is clear"

  Recognized: "effect is not clear"

  matched inserted matched

  - Error analysis: one deletion and one insertion
  - Measures: word error rate (WER), word correction rate (WCR), word accuracy rate (WAR)

Might be higher than 100%

$$\text{Word Error Rate} = 100\% \frac{\text{Sub.} + \text{Del.} + \text{Ins. words}}{\text{No. of words in the correct sentence}} = \frac{2}{4} = 50\%$$

$$\text{Word Correction Rate} = 100\% \frac{\text{Matched words}}{\text{No. of words in the correct sentence}} = \frac{3}{4} = 75\%$$

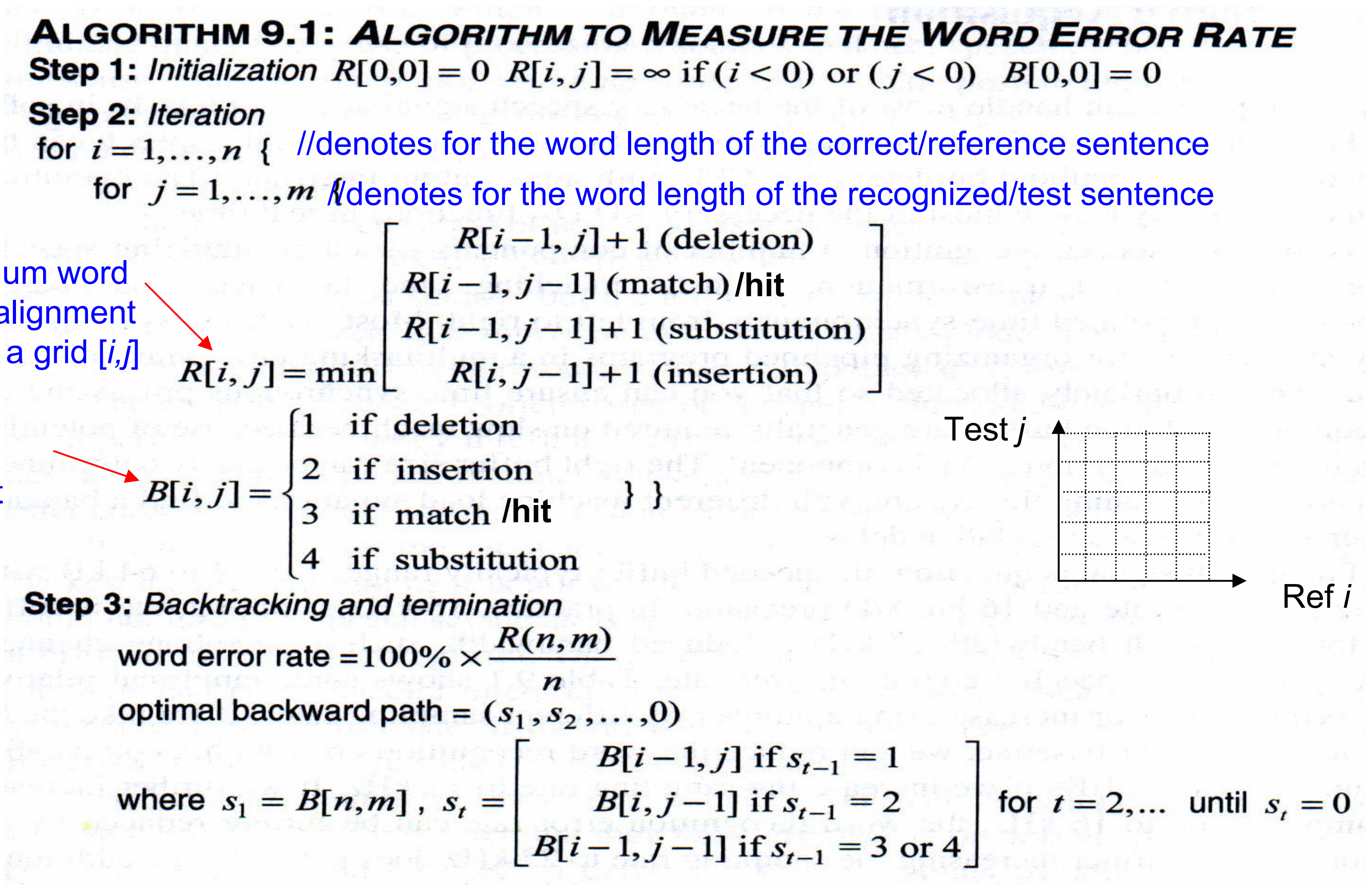
$$\text{Word Accuracy Rate} = 100\% \frac{\text{Matched - Ins. words}}{\text{No. of words in the correct sentence}} = \frac{3-1}{4} = 50\%$$

WER+ WAR =100%

Might be negative

Measures of ASR Performance (cont.)

- A Dynamic Programming Algorithm (Textbook)



ALGORITHM 9.1: *ALGORITHM TO MEASURE THE WORD ERROR RATE*

Step 1: *Initialization* $R[0,0]=0 \quad R[i,j]=\infty$ if $(i<0)$ or $(j<0)$ $\quad B[0,0]=0$

Step 2: *Iteration*

```
for $i=1,\dots,n$ {   //denotes for the word length of the correct/reference sentence
    for $j=1,\dots,m$ {   //denotes for the word length of the recognized/test sentence
        $R[i,j]=\min\begin{bmatrix} R[i-1,j]+1 \text{ (deletion)} \\ R[i-1,j-1] \text{ (match)/hit} \\ R[i-1,j-1]+1 \text{ (substitution)} \\ R[i,j-1]+1 \text{ (insertion)} \end{bmatrix}$
        $B[i,j]=\begin{cases} 1 & \text{if deletion} \\ 2 & \text{if insertion} \\ 3 & \text{if match /hit} \\ 4 & \text{if substitution} \end{cases}$
} }
```

minimum word error alignment at the a grid [*i,j*]

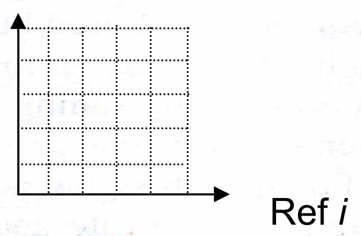
kinds of alignment

Step 3: *Backtracking and termination*

word error rate $=100\% \times \dfrac{R(n,m)}{n}$

optimal backward path = $(s_1, s_2, \dots, 0)$

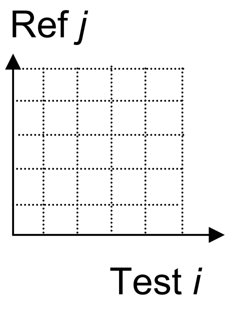
where $s_1 = B[n,m]$, $s_t = \begin{bmatrix} B[i-1,j] \text{ if } s_{t-1}=1 \\ B[i,j-1] \text{ if } s_{t-1}=2 \\ B[i-1,j-1] \text{ if } s_{t-1}=3 \text{ or } 4 \end{bmatrix}$ for $t=2,\dots$ until $s_t=0$



Measures of ASR Performance (cont.)

- Algorithm (by Berlin Chen)

```
Step 1: Initialization:
        $G[0][0] = 0;$
        for $i = 1,...,n$ { //test
            $G[i][0] = G[i-1][0] + 1;$
            $B[i][0] = 1;$ //Insertion
        }   (Horizontal Direction)
        for $j = 1,...,m$ { //reference
            $G[0][j] = G[0][j-1] + 1;$
            $B[0][j] = 2;$ // Deletion
        }   (Vertical Direction)
```



```
Step 2: Iteration:
    for $i = 1,...,n$ { //test
      for $j = 1,...,m$ { //reference
        $G[i][j] = \min \begin{bmatrix} G[i-1][j]+1 \text{ (Insertion)} \\ G[i][j-1]+1 \text{ (Delection)} \\ G[i-1][j-1]+1 \text{ (if } LR[j] != LT[i], \text{Substitution)} \\ G[i-1][j-1] \text{ (if } LR[j] = LT[i], \text{Match)} \end{bmatrix}$
        $B[i][j] = \begin{cases} 1; \text{//Insertion, (Horizontal Direction)} \\ 2; \text{//Deletion, (Vertical Direction)} \\ 3; \text{//Substitution (Diagonal Direction)} \\ 4; \text{//match (Diagonal Direction)} \end{cases}$

      } //for j, reference
    } //for i, test
```

```
Step 3: Measure and Backtrace:
    $\text{Word Error Rate} = 100\% \times \frac{G[n][m]}{m}$
    $\text{Word Accuracy Rate} = 100\% - \text{Word Error Rate}$
    $\text{Optimal backtrace path} = (B[n][m] \to ..... \to B[0][0])$
    if       $B[i][j] = 1$   print "        LT[i]" ; //Insertio n,   then go left
    else if  $B[i][j] = 2$   print "LR[j]        "; //Deletion ,   then go down
    else                     print "LR[j]   LR[i] "; //Hit/Matc h or Substituti on,   then go down diagonally
```

Note: the penalties for substitution, deletion and insertion errors are all set to be 1 here

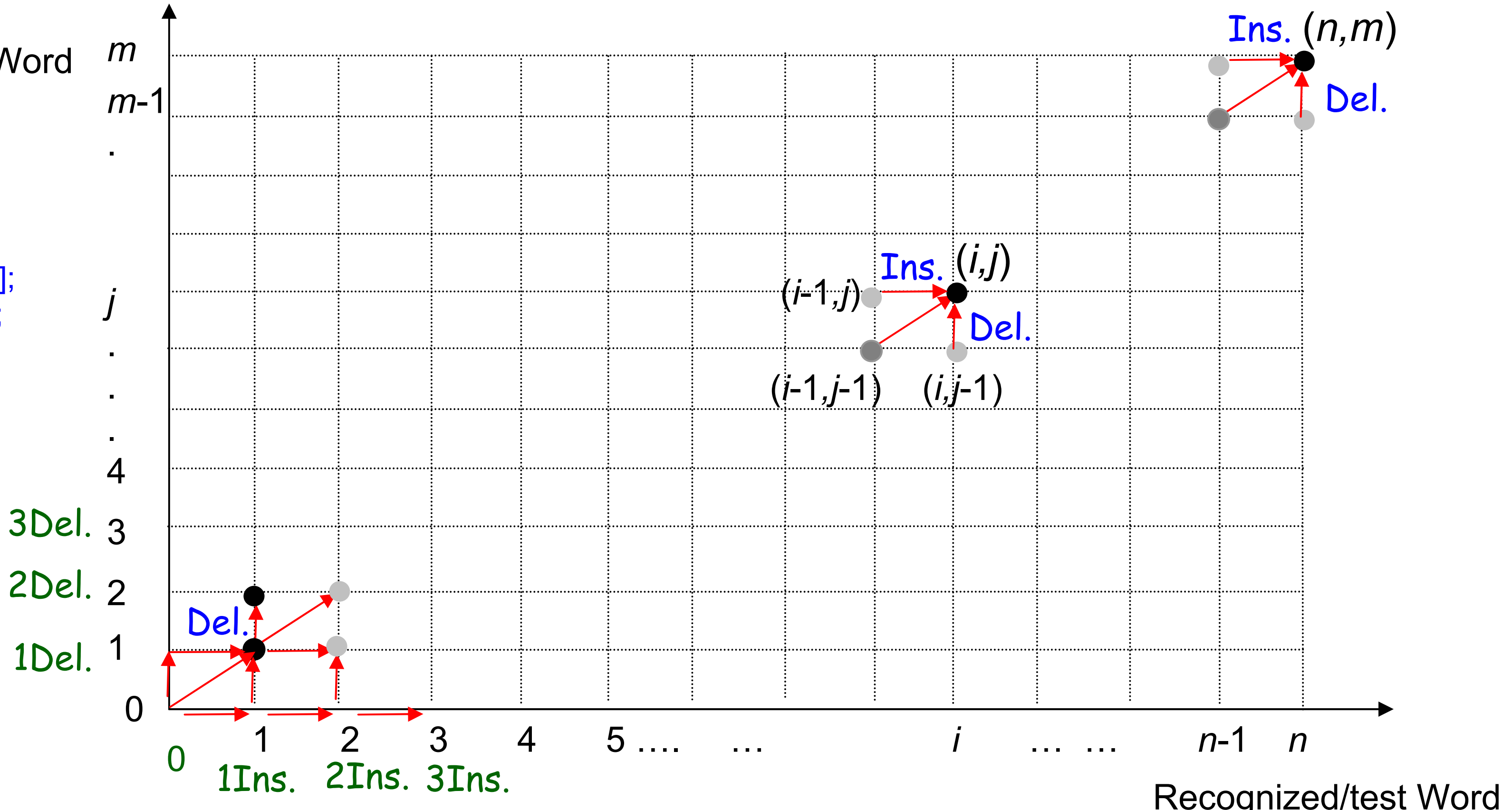
Measures of ASR Performance (cont.)

- A Dynamic Programming Algorithm
  - Initialization

```
for (j=1;j<=m;j++)
 { //reference
  grid[0][j] = grid[0][j-1];
  grid[0][j].dir = VERT;
  grid[0][j].score
       += DelPen;
  grid[0][j].del ++;

 }
```

HTK



```
grid[0][0].score = grid[0][0].ins
= grid[0][0].del = 0;
grid[0][0].sub = grid[0][0].hit = 0;
grid[0][0].dir = NIL;
```

```
for (i=1;i<=n;i++) { // test
      grid[i][0] = grid[i-1][0];
      grid[i][0].dir = HOR;
      grid[i][0].score +=InsPen;
      grid[i][0].ins ++;
}
```

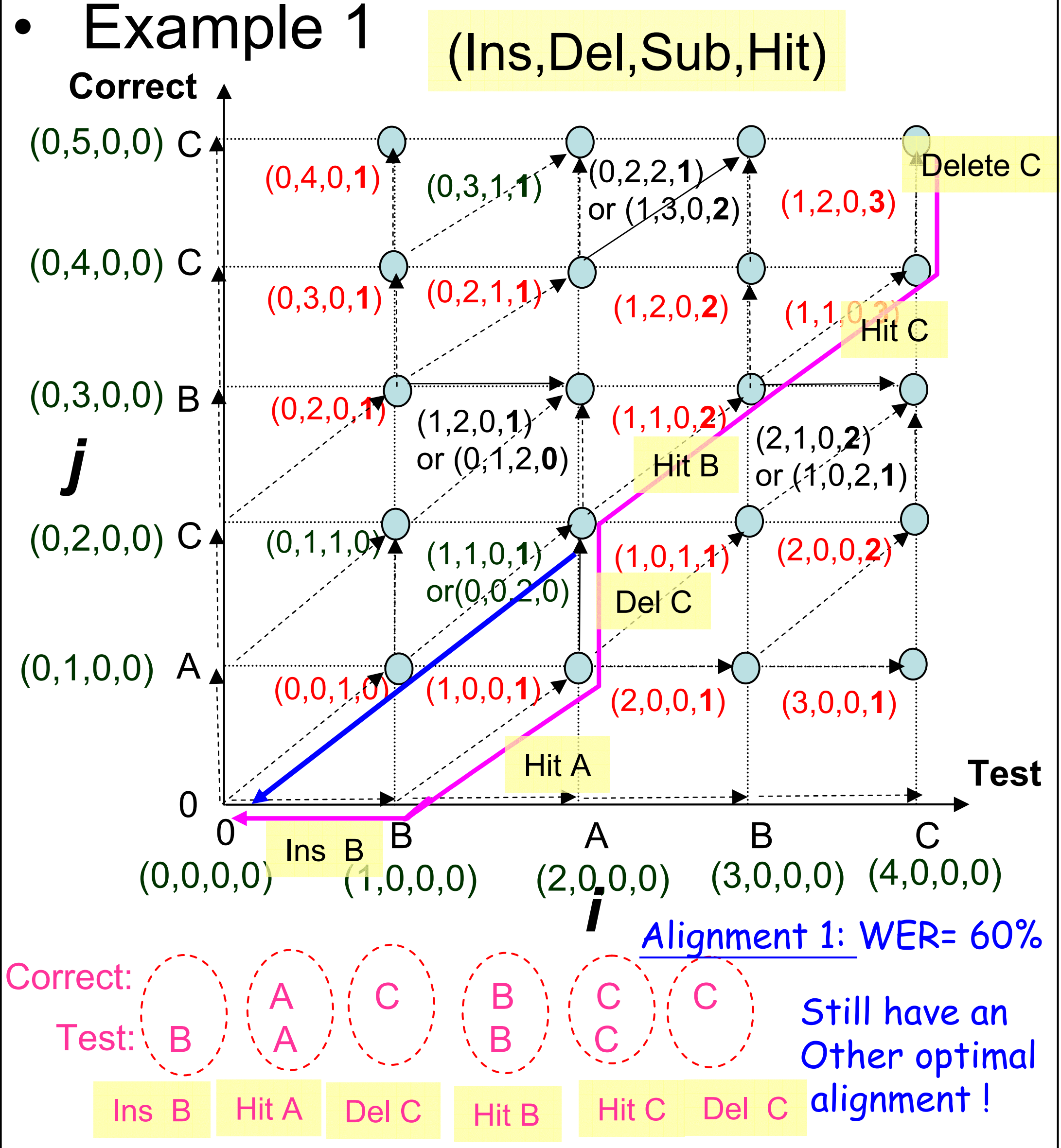
Measures of ASR Performance (cont.)

- Program

HTK

```
for (i=1;i<=n;i++) //test
 {   gridi = grid[i]; gridi1 = grid[i-1];
     for (j=1;j<=m;j++) //reference
     {       h = gridi1[j].score +insPen;
             d = gridi1[j-1].score;
             if (lRef[j] != lTest[i])
                d += subPen;
             v = gridi[j-1].score + delPen;
             if (d<=h && d<=v) { /* DIAG = hit or sub */
                gridi[j] = gridi1[j-1]; //structure assignment
                gridi[j].score = d;
                gridi[j].dir = DIAG;
                if (lRef[j] == lTest[i])    ++gridi[j].hit;
                else  ++gridi[j].sub;
             }
              else if (h<v) {        /* HOR = ins */
                gridi[j] = gridi1[j];     //structure assignment
                gridi[j].score = h;
                gridi[j].dir = HOR;
                ++ gridi[j].ins;
             }
             else {                  /* VERT = del */
                gridi[j] = gridi[j-1];   //structure assignment
                gridi[j].score = v;
                gridi[j].dir = VERT;
                ++gridi[j].del;   }
     }           /* for j */
 } /* for i */
```

- Example 1



Alignment 1: WER= 60%

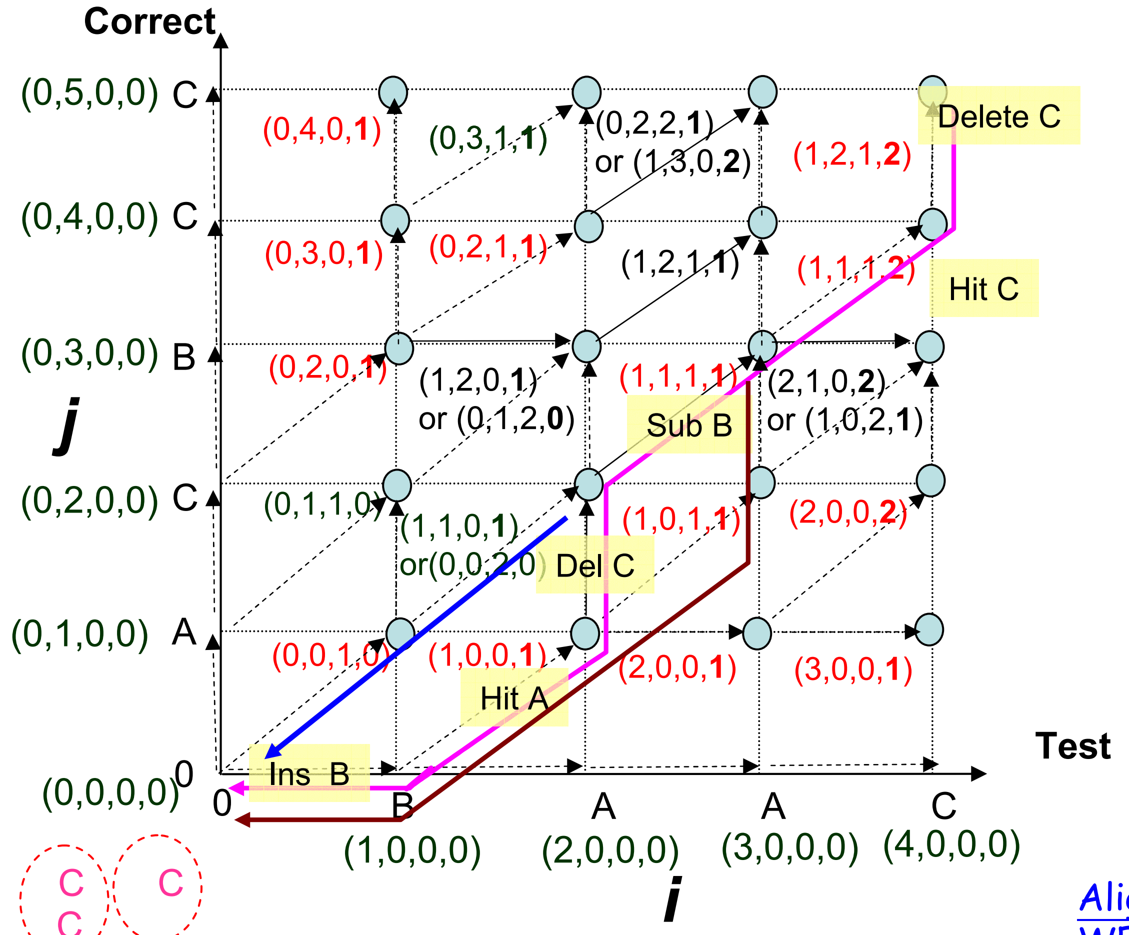
Still have an Other optimal alignment !

Measures of ASR Performance (cont.)

- Example 2

Note: the penalties for substitution, deletion and insertion errors are all set to be 1 here

(Ins,Del,Sub,Hit)



Alignment 1: WER= 80%

Correct:		A	C	B	C	C
Test:	B	A		A	C	
	Ins B	Hit A	Del C	Sub B	Hit C	Del C

Correct:	A	C	B	C	C
Test:	B	A	A	C	
	Sub A	Sub C	Sub B	Hit C	Del C

Alignment 2: WER=80%

Alignment 3: WER=80%

Correct:		A	C	B	C	C
Test:	B	A	A		C	
	Ins B	Hit A	Sub C	Del B	Hit C	Del C

Measures of ASR Performance (cont.)

- Two common settings of different penalties for substitution, deletion, and insertion errors

```
/* HTK error penalties */
 subPen = 10;
 delPen = 7;
 insPen = 7;

/* NIST error penalties*/
 subPenNIST = 4;
 delPenNIST = 3;
 insPenNIST = 3;
```

Choice of Appropriate Units for HMMs

- Issues for HMM Modeling units
  - **Accurate**: accurately represent the acoustic realization that appears in different contexts
  - **Trainable**: have enough data to estimate the parameters of the unit (or HMM model)
  - **Generalizable**: any new word can be derived from a predefined unit inventory for task-independent speech recognition

Choice of Appropriate Units for HMMs (cont.)

- Comparison of different units
  - **Word**:
    - Semantic meaning, capturing within-word coarticulation, can be accurately trained for small-vocabulary speech recognition, but not generalizable for modeling unseen words and interword coarticulation

subword

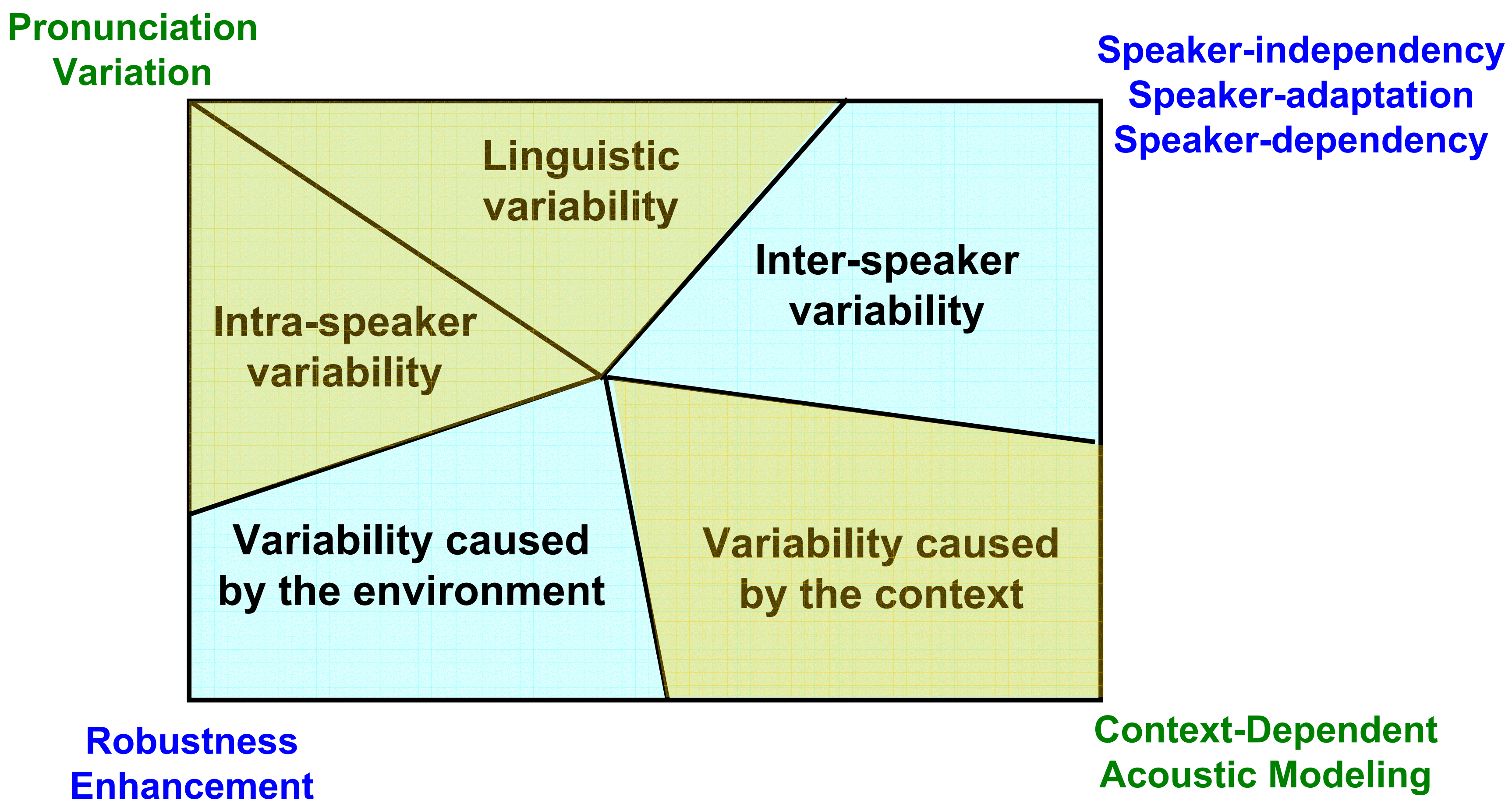
  - **Phone**:
    - More trainable and generalizable, but less accurate
    - There are only about 50 context-independent phones in English and 30 in Mandarin Chinese
    - Drawbacks: the realization of a phoneme is strongly affected by immediately neighboring phonemes (e.g., /t s/ and /t r/)
  - **Syllable**:
    - A compromise between the word and phonetic models. Syllables are larger than phone
    - There only about 1,300 tone-dependent syllables in Chinese and 50 in Japanese. However, there are over 30,000 in English

Choice of Appropriate Units for HMMs (cont.)

- Phonetic Structure of Mandarin Syllables

Syllables (1,345)		
Base-syllables (408)		**Tones (4+1)**
INITIAL’s (21)	**FINAL’s (37)**	
Phone-like Units/Phones (33)		

Variability in the Speech Signals


Pronunciation Variation
Speaker-independency
Speaker-adaptation
Speaker-dependency
Linguistic variability
Inter-speaker variability
Intra-speaker variability
Variability caused by the environment
Variability caused by the context
Robustness Enhancement
Context-Dependent Acoustic Modeling

Variability in the Speech Signals (cont.)

- Context Variability
  - Context variability at word/sentence level
    - E.g., "*Mr. Wright should write to Ms. Wright right away about his Ford or four door Honda*"
    - Same pronunciation but different meaning (*Wright , write , right*)
    - Phonetically identical and semantically relevant *(Ford or, four door)*

Pause or intonation information is needed

  - Context variability at phonetic level
    - The acoustic realization of phoneme */ee/* for word *peat* and *wheel* depends on its left and right context

the effect is more important in fast speech or spontaneous conversations, since many phonemes are not fully realized!

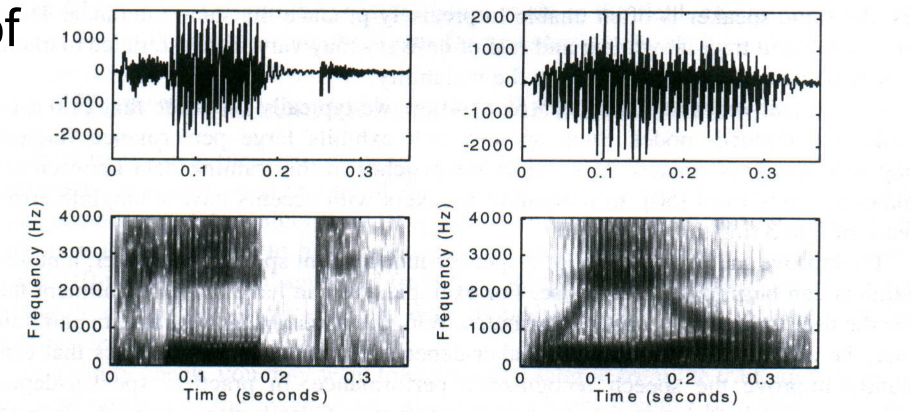


Figure 9.1 Waveforms and spectrograms for words *peat* (left) and *wheel* (right). The phoneme /ee/ is illustrated with two different left and right contexts. This illustrates that different contexts may have different effects on a phone.

Variability in the Speech Signals (cont.)

- Style Variability (also including intra-speaker and linguistic variability)
  - Isolated speech recognition
    - Users have to pause between each word (a clear boundary between words)
    - Errors such as "*Ford or*" and "*four door*" can be eliminated
    - But unnatural to most people
  - Continuous speech recognition
    - Causal, spontaneous, and conversational
    - Higher speaking rate and co-articulation effects
    - Emotional changes also introduce more significantly variations

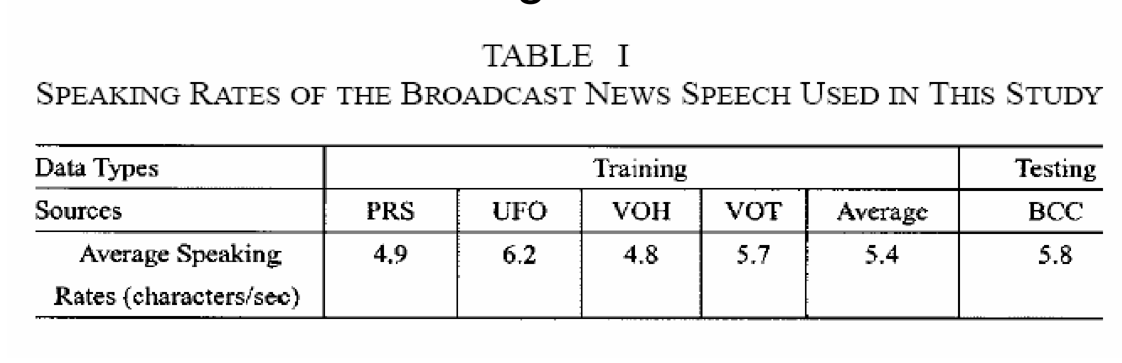


TABLE I

SPEAKING RATES OF THE BROADCAST NEWS SPEECH USED IN THIS STUDY

Data Types	Training					Testing
Sources	PRS	UFO	VOH	VOT	Average	BCC
Average Speaking Rates (characters/sec)	4.9	6.2	4.8	5.7	5.4	5.8

Statistics of the speaking rates of the broadcast new speech collected in Taiwan

Variability in the Speech Signals (cont.)

- Speaker Variability
  - Interspeaker
    - Vocal tract size, length and width of the neck and a range of physical characteristics
    - E.g., gender, age, dialect, health, education, and personal style
  - Intraspeaker
    - The same speaker is often unable to precisely produce the same utterance
    - The shape of the vocal tract movement and rate of delivery may vary from utterance to utterance

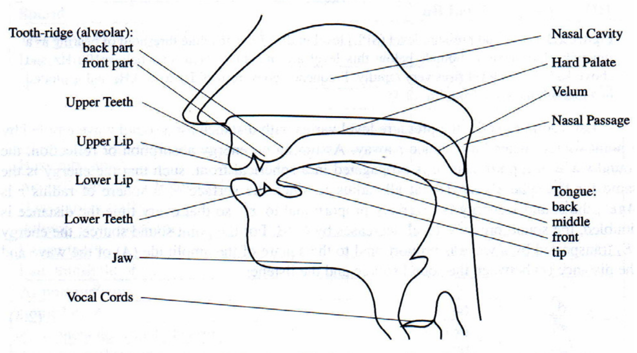


Figure 2.4 A schematic diagram of the human speech production apparatus.

  - Issues for acoustic modeling
    - Speaker-dependent (SD), speaker-independent (SI) and speaker-adaptive (SA) modeling
    - Typically an SD system can reduce WER by more than 30% as compared with a comparable SI one

Variability in the Speech Signals (cont.)

- Environment Variability
  - The world we live in is full of sounds of varying loudness from different sources
  - Speech recognition in hands-free or mobile environments remain one of the most severe challenges
    - The spectrum of noises varies significantly
  - Noise may also be present from the input device itself, such as microphone and A/D interface noises
  - We can reduce the error rates by using multi-style training or adaptive techniques
  - Environment variability remains as one of the most severe challenges facing today's state-of-the-art speech systems

Context Dependency

- Review: Phone and Phoneme
  - In speech science, the term **phoneme** is used to denote any of the minimal units of speech sound in a language that can serve to distinguish one word from another
  - The term **phone** is used to denote a phoneme’s acoustic realization
  - E.g., English phoneme ***/t/*** has two very different acoustic realizations in the word *sa**t*** and *me**t**er*
    - We have better treat them as two different phones when building a spoken language system

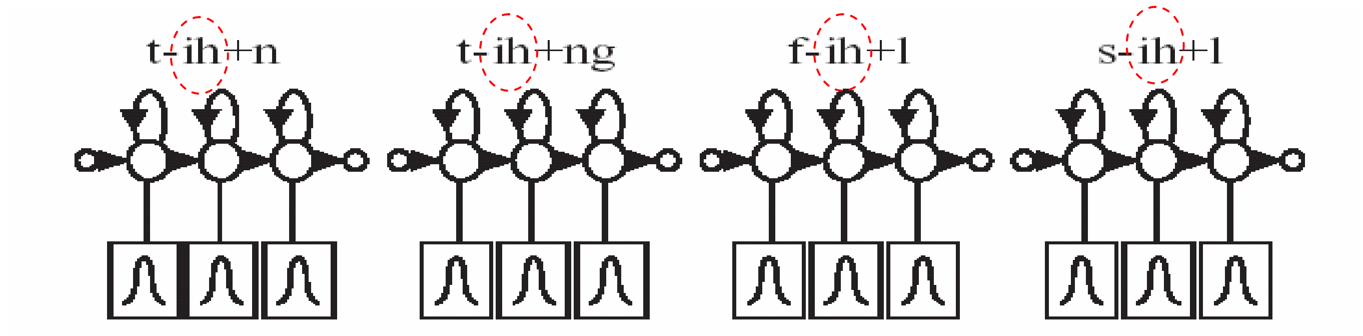
Context Dependency (cont.)

- Why Context Dependency
  - If we make unit context dependent, we can significantly improve the recognition accuracy, provided there are enough training data for parameter estimation
  - A context usually refers to the immediate left and/or right neighboring phones
  - Context-dependent (CD) phonemes have been widely used for LVCSR systems

Context Dependency (cont.)

allophones: different realizations of a phoneme is called allophones
→**Triphones** are examples of allophones

- Triphone (Intra-word triphone)
  - A triphone model is a phonetic model that takes into consideration both the left and right neighboring phones
    - It captures the most important coarticulatory effects
  - Two phones having the same identity but different left and right context are considered different triphones
  - Challenging issue: Need to balance trainability and accuracy with a number of parameter-sharing techniques



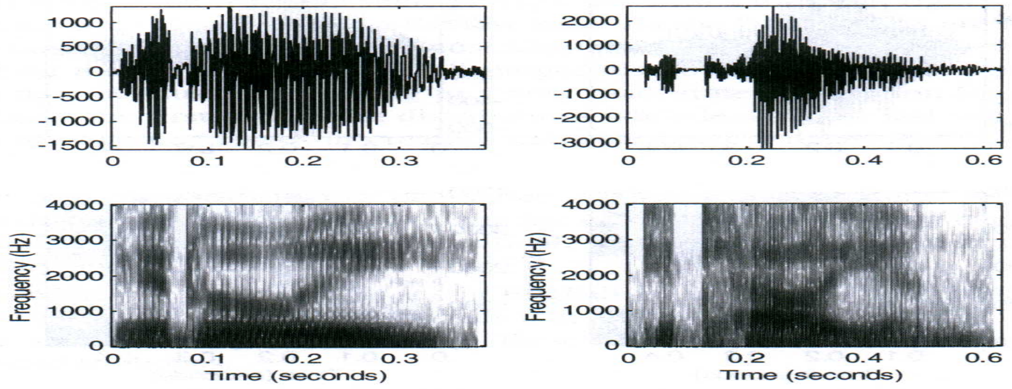
Context Dependency (cont.)

- Modeling inter-word context-dependent phone (like triphones) is complicated
  - Although the juncture effect on word boundaries is one of the most serious coarticulation phenomena in continuous speech recognition
    - E.g., speech /*s p iy ch*/→ /*s*/ and /*ch*/ are depending on the preceding and following words in actual sentences
  - Should be taken into consideration with the decoding/search scheme adopted
- Even with the same left/right context, a phone may have significant different realizations at different word positions
  - E.g., that rock /*t*/→ extinct! , theatrical /*t*/→/*ch*/

Context Dependency (cont.)

- Stress information for context dependency
  - **Word-level stress** (free stress)
    - The stress information: longer duration, higher pitch and more intensity for stressed vowels
    - E.g., ***im**port* (n) vs. *im**port*** (v), ***con**tent* (n) vs. *con**tent*** (v)

***It**aly*



*I**ta**lian*

Figure 9.3 The importance of stress is illustrated in *Italy* vs. *Italian* for phone /t/. The realizations are quite different, even though they share the same left and right context.

  - **Sentence-level stress** (including contrastive and emphatic stress)
    - Sentence-level stress is very hard to model without incorporate semantic and pragmatic knowledge
    - Contrastive: e.g., "I said **import** records not **export**"
    - Emphatic: e.g., "I **did** have dinner"

Clustered Acoustic-Phonetic Units

- Triphone modeling assumes that every triphone context is different. Actually, many phones have similar effects on the neighboring phones
  - /b/ and /p/ (labial stops) (or, /r/ and /w/ (liquids)) have similar effects on the following vowel
- It is desirable to find instances of similar contexts and merge them
  - A much more manageable number of models that can be better trained

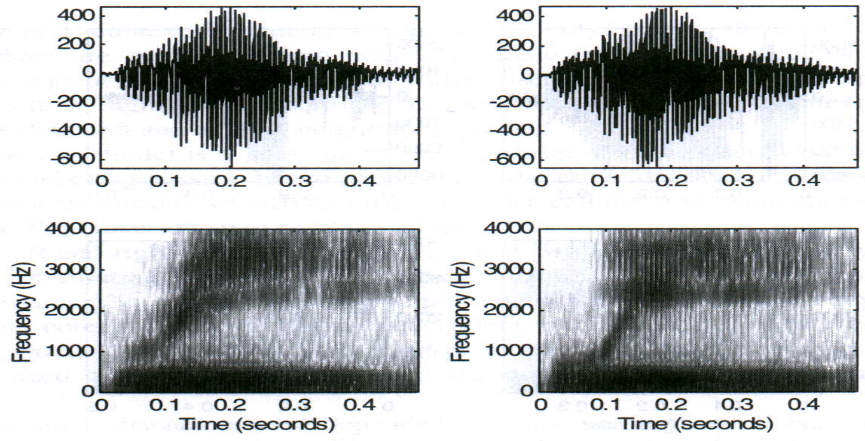
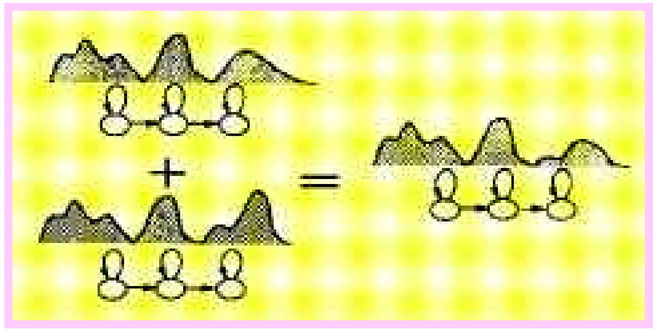


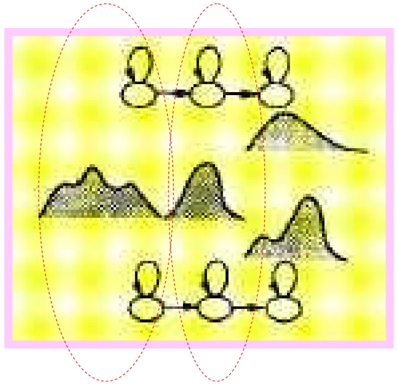
Figure 9.4 The spectrograms for the phoneme /*iy*/ with two different *left-contexts* are illustrated. Note that /*r*/ and /*w*/ have similar effects on /*iy*/. This illustrates that different left-contexts may have similar effects on a phone.

Clustered Acoustic-Phonetic Units (cont.)

- Model-based clustering



- State-based clustering (state-tying)
  - Keep the dissimilar states of two models apart while the other corresponding states are merged



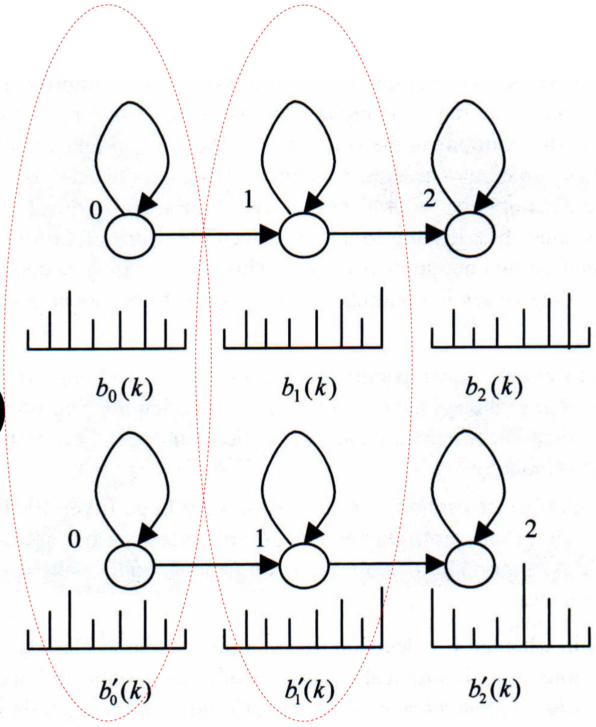
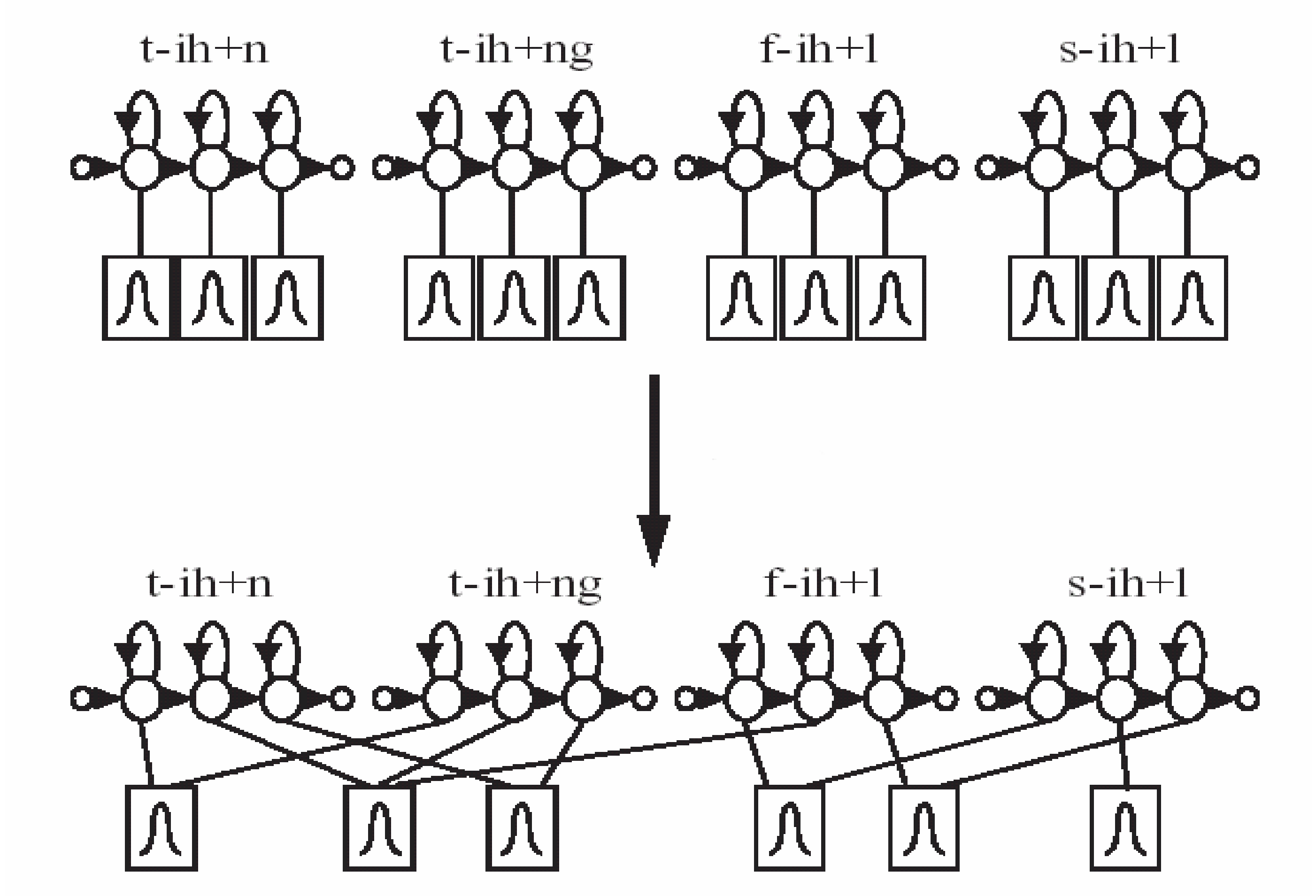


Figure 9.5 State-based vs. model-based clustering. These two models are very similar, as both the first and the second output distributions are almost identical. The key difference is the output distribution of the third state. If we measure the overall model similarity, which is often based on the accumulative output distribution similarities of all states, these two models may be clustered, leading to a very inaccurate distribution for the last state. If we cluster output distributions at state level, we can cluster the first two output distributions while leaving the last ones intact, leading to more accurate representations.

Clustered Acoustic-Phonetic Units (cont.)

- State-tying of triphones

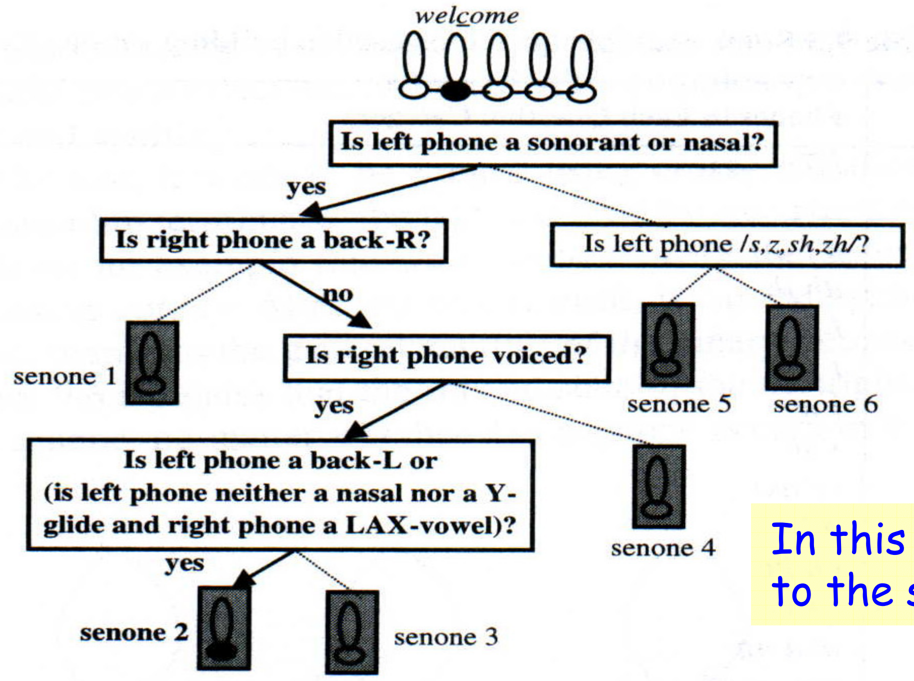


Clustered Acoustic-Phonetic Units (cont.)

- Two key issues for CD phonetic or subphonetic modeling
  - Tying the phones with similar contexts to improve trainability and efficiency
    - Enable better parameter sharing and smoothing
  - Mapping the unseen triphones (in the test) into appropriately trained triphones is important
    - Because the possible of triphones could be very lagre
    - E.g., English has over 100,000 triphones

Clustered Acoustic-Phonetic Units (cont.)

- Microsoft's approach - State-based clustering
  - Generate clustering to the state-dependent output distributions across different phonetic models
  - Each cluster represents a set of similar HMM states and is called ***senone***
  - A subword model is composed of a sequence of senons

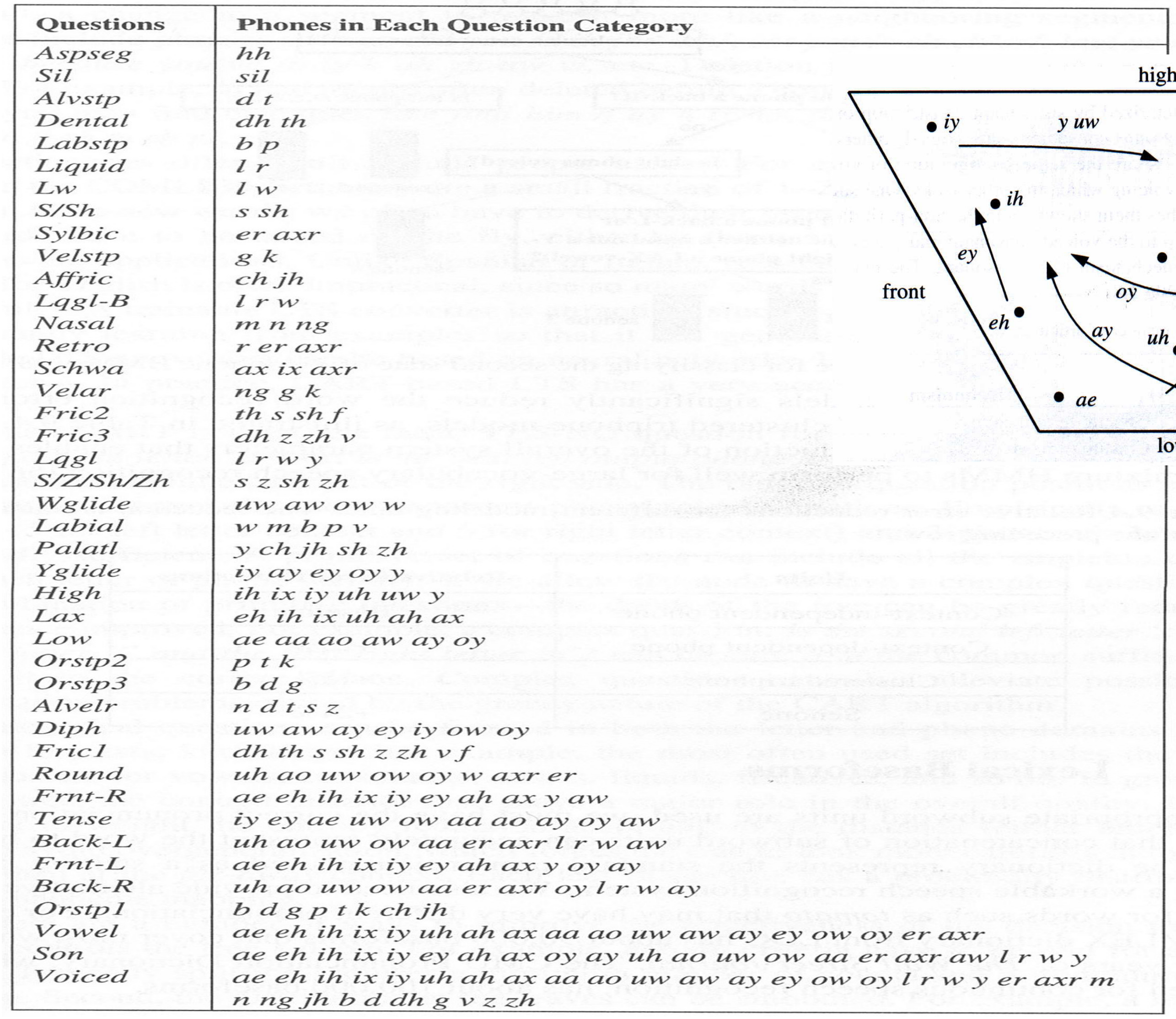

welcome
Is left phone a sonorant or nasal?
yes
Is right phone a back-R?
Is left phone /s,z,sh,zh/?
no
senone 1
Is right phone voiced?
senone 5
senone 6
yes
Is left phone a back-L or (is left phone neither a nasal nor a Y-glide and right phone a LAX-vowel)?
senone 4
yes
senone 2
senone 3

In this example, the tree can be applied to the second state of any /k/ triphone

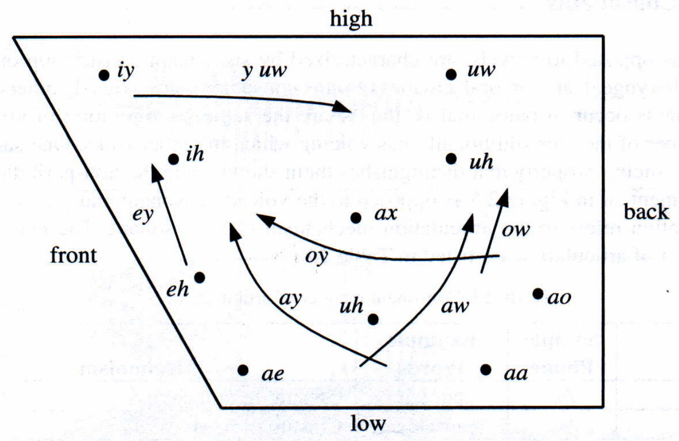
Figure 9.6 A decision tree for classifying the second state of *K*-triphone HMMs [48].

Clustered Acoustic-Phonetic Units (cont.)

- Some example questions used in building senone trees



Questions	Phones in Each Question Category
Aspseg	hh
Sil	sil
Alvstp	d t
Dental	dh th
Labstp	b p
Liquid	l r
Lw	l w
S/Sh	s sh
Sylbic	er axr
Velstp	g k
Affric	ch jh
Lqgl-B	l r w
Nasal	m n ng
Retro	r er axr
Schwa	ax ix axr
Velar	ng g k
Fric2	th s sh f
Fric3	dh z zh v
Lqgl	l r w y
S/Z/Sh/Zh	s z sh zh
Wglide	uw aw ow w
Labial	w m b p v
Palatl	y ch jh sh zh
Yglide	iy ay ey oy y
High	ih ix iy uh uw y
Lax	eh ih ix uh ah ax
Low	ae aa ao aw ay oy
Orstp2	p t k
Orstp3	b d g
Alvelr	n d t s z
Diph	uw aw ay ey iy ow oy
Fric1	dh th s sh z zh v f
Round	uh ao uw ow oy w axr er
Frnt-R	ae eh ih ix iy ey ah ax y aw
Tense	iy ey ae uw ow aa ao ay oy aw
Back-L	uh ao uw ow aa er axr l r w aw
Frnt-L	ae eh ih ix iy ey ah ax y oy ay
Back-R	uh ao uw ow aa er axr oy l r w ay
Orstp1	b d g p t k ch jh
Vowel	ae eh ih ix iy uh ah ax aa ao uw aw ay ey ow oy er axr
Son	ae eh ih ix iy ey ah ax oy ay uh ao uw ow aa er axr aw l r w y
Voiced	ae eh ih ix iy uh ah ax aa ao uw aw ay ey ow oy l r w y er axr m n ng jh b d dh g v z zh



Clustered Acoustic-Phonetic Units (cont.)

- Comparison of recognition performance for different acoustic modeling

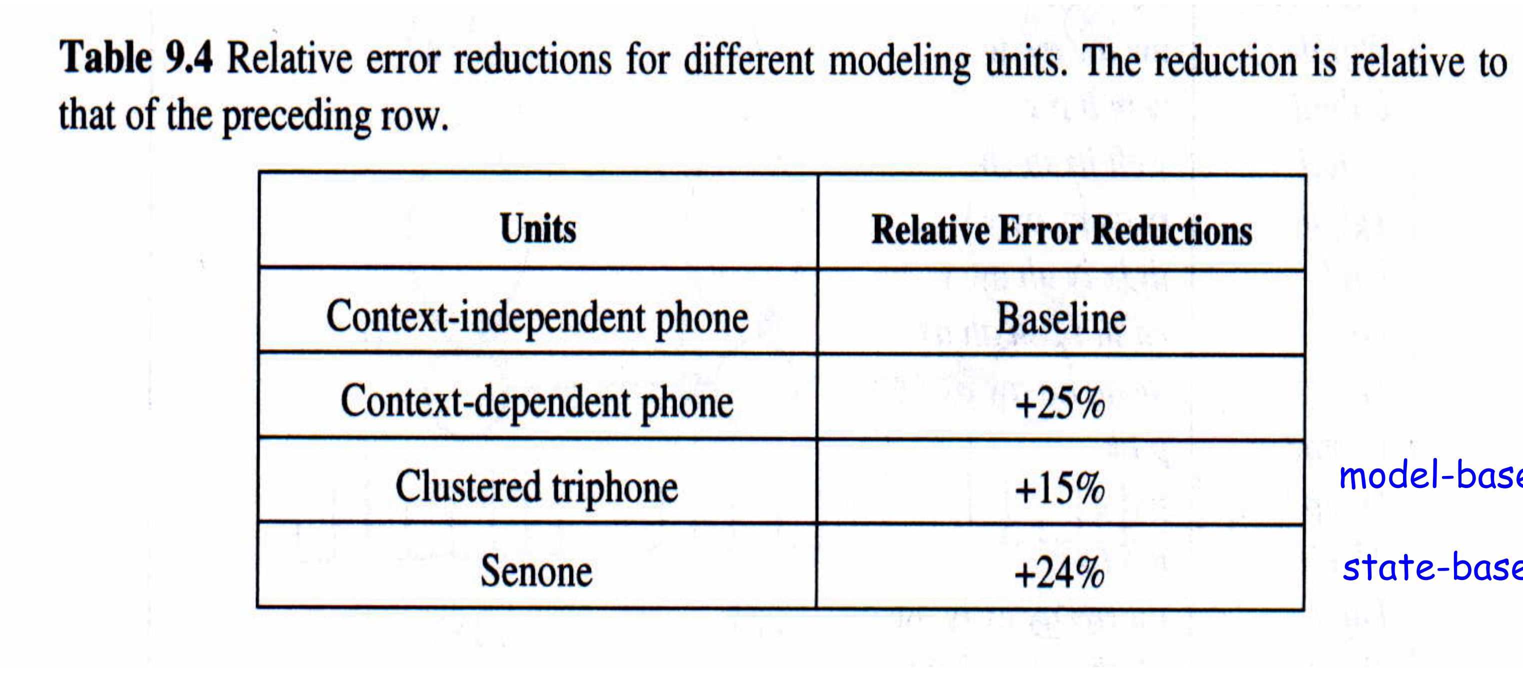


Table 9.4 Relative error reductions for different modeling units. The reduction is relative to that of the preceding row.

Units	Relative Error Reductions
Context-independent phone	Baseline
Context-dependent phone	+25%
Clustered triphone	+15%
Senone	+24%

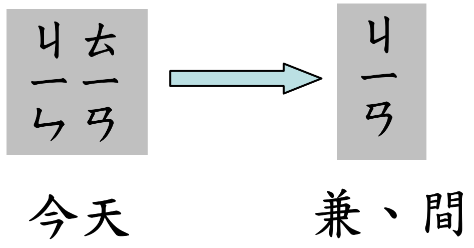
model-based clustering

state-based clustering

Pronunciation Variation

- We need to provide alternative pronunciations for words that may have very different pronunciations
  - In continuous speech recognition, we must handle the modification of **interword pronunciations** and **reduced sounds**
- Variation kinds
  - **Co-articulation** (Assimilation)

    “*did you*” */d ih jh y ah/*, “*set you*” */s eh ch er/*
    - Assimilation: a change in a segment to make it more like a neighboring segment
  - **Deletion**
    - */t/* and */d/* are often deleted before a consonant
- Variation can be drawn between
  - Inter-speaker variation (social)
  - Intra-speaker variation (stylistic)



Pronunciation Variation (cont.)

- Pronunciation Network (a probabilistic finite state machine)

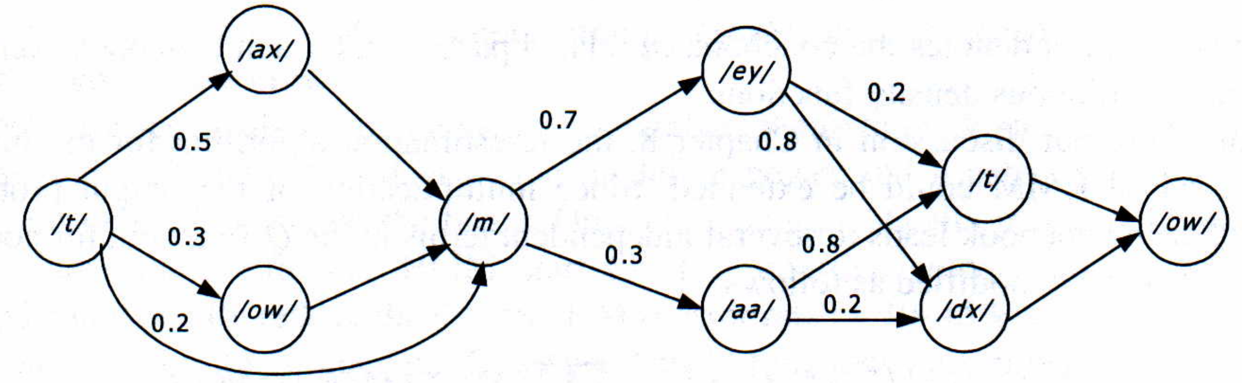


Figure 9.7 A possible pronunciation network for word *tomato*. The vowel /*ey*/ is more likely to flap, thereby having a higher transition probability into /*dx*/.

- Examples:
  - E. g., word "*that*" appears 328 times in one corpus, with 117 different tokens of the 328 times (only 11% of the tokens are most frequent)

    Greenberg, 1998
  - Cheating experiments show big performance improvements achieved if the tuned pronunciations were applied to those in test data (e.g. Switchboard WER goes from 40% to 8%)

    McAllaster et al., 1998

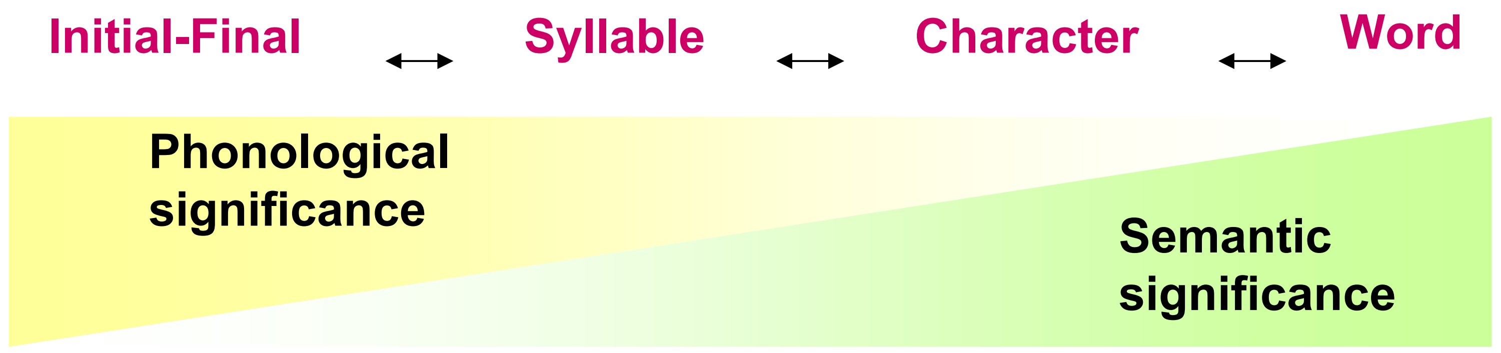
Pronunciation Variation (cont.)

- Adaptation of Pronunciations
  - Dialect-specific pronunciations
  - Native vs. non-native pronunciations
  - Rate-specific pronunciations

- Side Effect
  - Adding more and more variants to the pronunciation lexicon increases size and confusion of the vocabulary
    - Lead to increased ASR WER

Characteristics of Mandarin Chinese

- **Four levels of linguistic units**

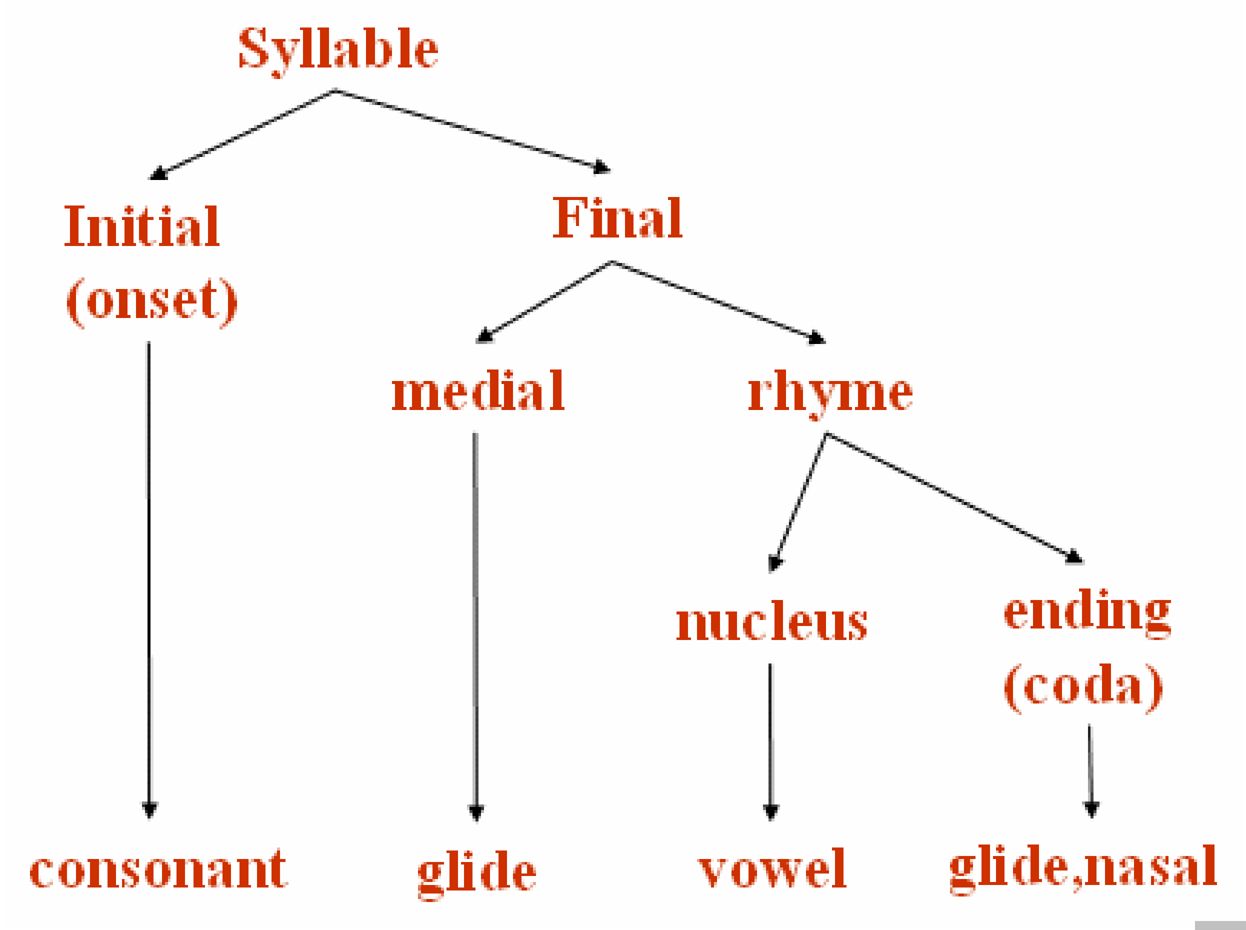

Initial-Final ↔ Syllable ↔ Character ↔ Word
Phonological significance
Semantic significance

- **A monosyllabic-structure language**
  - All characters are monosyllabic
- **Most characters are morphemes** (詞素)
- **A word is composed of one to several characters**
- **Homophones**
  - Different characters sharing the same syllable

from Ming-yi Tsai

Characteristics of Mandarin Chinese (cont.)

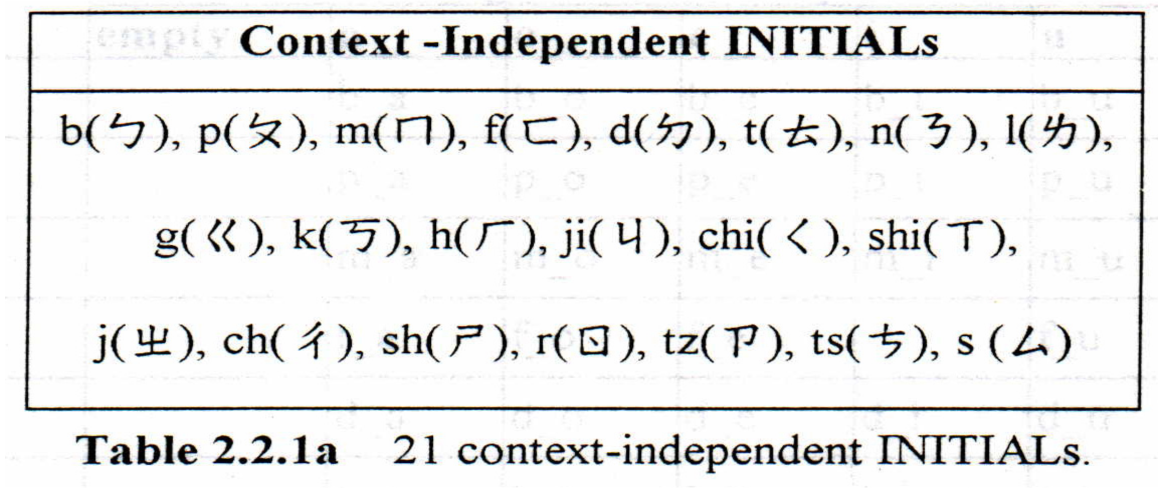
- **Chinese syllable structure**



from Ming-yi Tsai

Characteristics of Mandarin Chinese (cont.)

- Sub-syllable HMM Modeling
  - INITIALs

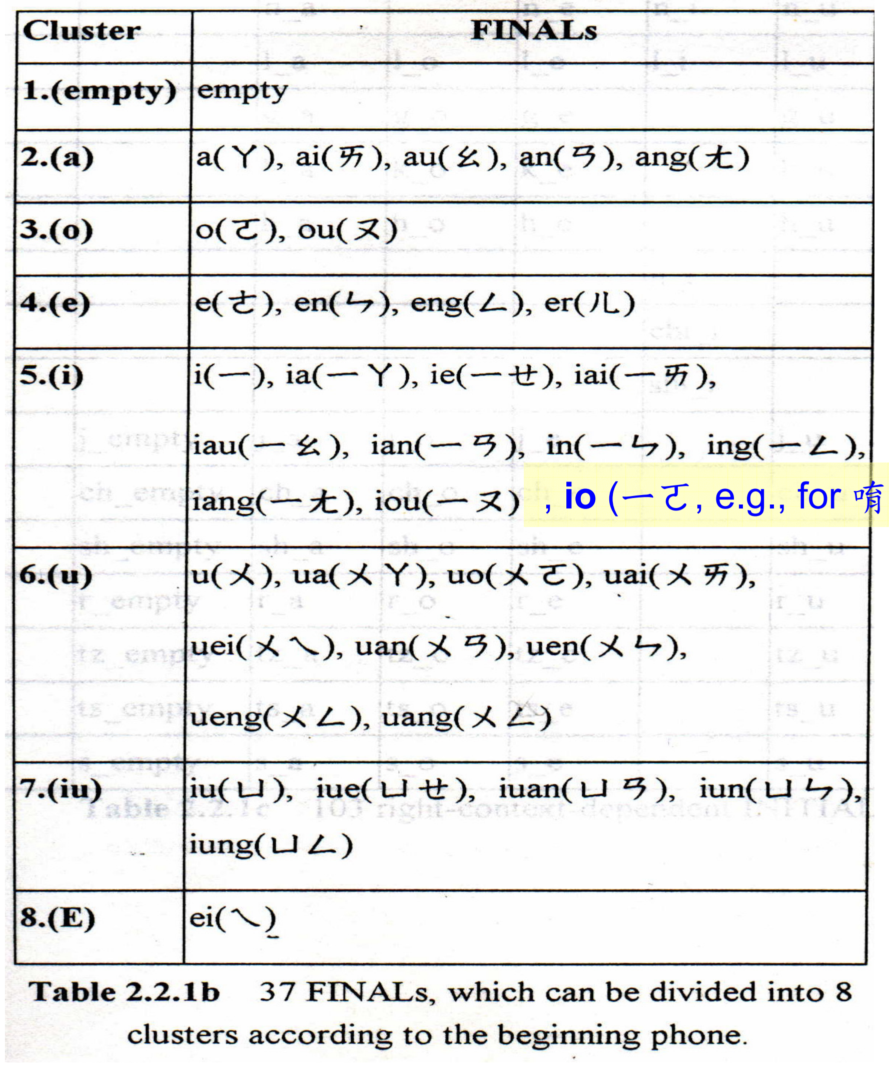


Context -Independent INITIALs
b(ㄅ), p(ㄆ), m(ㄇ), f(ㄈ), d(ㄉ), t(ㄊ), n(ㄋ), l(ㄌ),
g(ㄍ), k(ㄎ), h(ㄏ), ji(ㄐ), chi(ㄑ), shi(ㄒ),
j(ㄓ), ch(ㄔ), sh(ㄕ), r(ㄖ), tz(ㄗ), ts(ㄘ), s (ㄙ)

Table 2.2.1a 21 context-independent INITIALs.

Sub-Syllable HMM Modeling (cont.)

- Sub-syllable HMM Modeling
  - FINALs



Cluster	FINALs
1.(empty)	empty
2.(a)	a(ㄚ), ai(ㄞ), au(ㄠ), an(ㄢ), ang(ㄤ)
3.(o)	o(ㄛ), ou(ㄡ)
4.(e)	e(ㄜ), en(ㄣ), eng(ㄥ), er(ㄦ)
5.(i)	i(ㄧ), ia(ㄧㄚ), ie(ㄧㄝ), iai(ㄧㄞ), iau(ㄧㄠ), ian(ㄧㄢ), in(ㄧㄣ), ing(ㄧㄥ), iang(ㄧㄤ), iou(ㄧㄡ) , **io** (ㄧㄛ, e.g., for 唷 was ignored here)
6.(u)	u(ㄨ), ua(ㄨㄚ), uo(ㄨㄛ), uai(ㄨㄞ), uei(ㄨㄟ), uan(ㄨㄢ), uen(ㄨㄣ), ueng(ㄨㄥ), uang(ㄨㄤ)
7.(iu)	iu(ㄩ), iue(ㄩㄝ), iuan(ㄩㄢ), iun(ㄩㄣ), iung(ㄩㄥ)
8.(E)	ei(ㄟ)

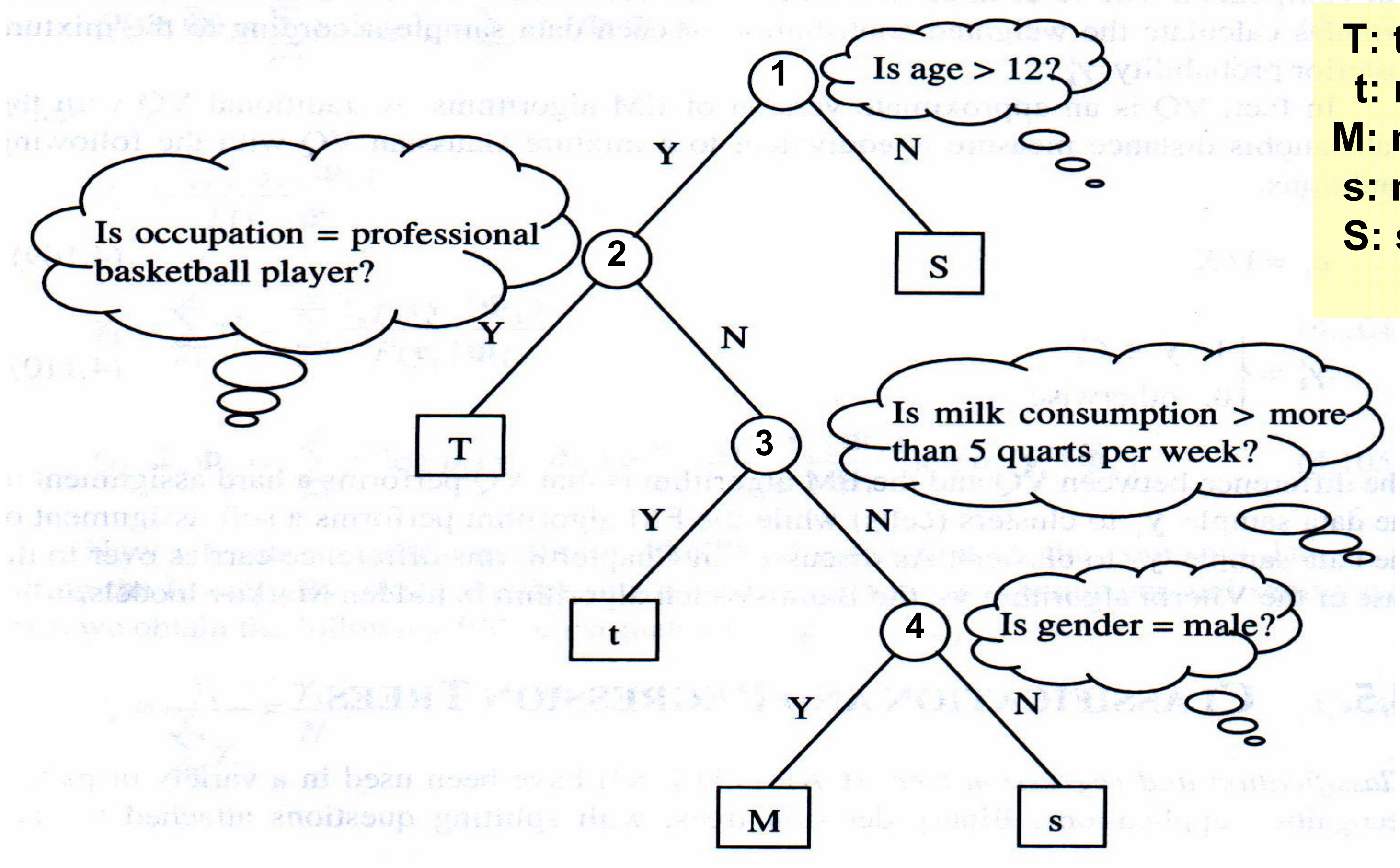
Table 2.2.1b 37 FINALs, which can be divided into 8 clusters according to the beginning phone.

Classification and Regression Trees (CART)

- CART are binary decision trees, with splitting questions attached to each node
  - Act like a rule-based system where the classification carried out by a sequence of decision rules
- CART provides an easy representation that interprets and predicates the structure of a set of data
  - Handle data with high dimensionality, mixed data type and nonstandard data structure
- CART also provides an automatic and data-driven framework to construct the decision process based on objective criteria, not subjective criteria
  - E.g., the choice and order of rules
- CART is a kind of clustering/classification algorithms

Classification and Regression Trees (cont.)

- Example: height classification
  - Assign a person to one of the following five height classes



1
Is age > 12?
Y
N
S
2
Is occupation = professional basketball player?
Y
N
T
3
Is milk consumption > more than 5 quarts per week?
Y
N
t
4
Is gender = male?
Y
N
M
s

Figure 4.14 A binary tree structure for height classification.

T: tall
t: medium-tall
M: medium
s: medium-sort
S: short

Classification and Regression Trees (cont.)

- Example: height classification (cont.)
  - Can easy predict the height class for any new person with all the measured data (age, occupation, milk-drinking, etc.) but no height information, by traversing the binary tree (based on a set of questions)
  - "No": right branch, "Yes" left branch
  - When reaching a leaf node, we can use its attached label as the height class for the new person
  - Also can use the average height in the leaf node to predict the height of the new person

CART Construction using Training Samples

- **Steps**
  1. First, find a set of questions regarding the measured variable
     - E.g., "Is age>12?", "Is gender=male?", etc.
  2. Then, place all the training samples in the root of the initial tree
  3. Choose the best question from the question set to split the root into two nodes (need some measurement !)
  4. Recursively split the most promising node with the best question until the right-sized tree is obtained

How to choose the best question?
- E.g., reduce the uncertainty of the event being decided upon
  i.e., find the question which gives the **greatest entropy reduction**

CART Construction using Training Samples (cont.)

- **Splitting Criteria (for discrete pdf)**
  - How to find the best question for a node split ?
    - I.e., find the best split for the data samples of the node
  - Assume training samples have a probability (density) function $P(\omega|t)$ at each node *t*
    - E.g., $P(\omega_i|t)$ is the percentage of data samples for class *i* at a node *t* and $\sum_i P(\omega_i|t) = 1$

CART Construction using Training Samples (cont.)

- **Splitting Criteria (for discrete pdf)**
  - Define the weighted entropy for any tree node *t*

$$\overline{H}_t(Y) = H_t(Y)P(t)$$

$$H_t(Y) = -\sum_i P(\omega_i|t)\log P(\omega_i|t), \text{ Entropy: average amount of information}$$

- Y is the random variable for classification decision
- $P(t)$ is the prior probability of visiting node *t* (ratio of numbers of samples in a node *t* and the total number of samples)

CART Construction using Training Samples (cont.)

- **Splitting Criteria (for discrete pdf)**
  - Entropy reduction for a question *q* to split a node *t* into nodes *l* and *r*
    - Pick the question with the greatest entropy reduction

$$\Delta \overline{H}_t(q) = \overline{H}_t(Y) - \left(\overline{H}_l(Y) + \overline{H}_r(Y)\right) = \overline{H}_t(Y) - \overline{H}_t(Y|q)$$

$$q^* = \arg\max_q \left[\Delta \overline{H}_t(q)\right]$$

Review: Fundamentals in Information Theory

- Three interpretations for quantity of information
  1. The amount of **uncertainty** before seeing an event
  2. The amount of **surprise** when seeing an event
  3. The amount of **information** after seeing an event

- The definition of information:

$$I(x_i) = \log \frac{1}{P(x_i)} = -\log P(x_i)$$

  – $P(x_i)$ the probability of an event x_i

- Entropy: the average amount of information

$$H(X) = E[I(X)]_X = E[-\log P(x_i)]_X = \sum_{x_i} -P(x_i) \cdot \log P(x_i)$$

$$\text{where } S = \{x_1, x_2, \ldots, x_i, \ldots\}$$

  – Have maximum value when the probability (mass) function is a uniform distribution

CART Construction using Training Samples (cont.)

- **Splitting Criteria (for discrete pdf)**
  - **Example**

X:{x_i=1, 1, 3, 3, 8, 8, 9, 9}
P(x=1)=1/4
P(x=3)=1/4
P(x=8)=1/4
P(x=9)=1/4

H=-4*(1/4)$\log_2$(1/4)=2

1

Y:{y_i=1, 1}
P(y=1)=1

Z:{z_i=3, 3, 8, 8, 9, 9}
P(z=3)=1/3
P(z=8)=1/3
P(z=9)=1/3

H_l=-1*(1)$\log_2$(1)=0;

H_r=-3*(1/3)$\log_2$(1/3)=1.6;

$$\overline{H}_l = H_l \cdot P(\text{Node}_l) = 0 \cdot 1/4 = 0$$

$$\overline{H}_r = H_r \cdot P(\text{Node}_r) = 1.6 \cdot 3/4 = 1.2$$

$$\overline{H} = \overline{H}_1 + \overline{H}_2 = 1.2$$

2

Y:{y_i=1, 1, 3, 3}
P(y=1)=1/2
P(y=3)=1/2

Z:{z_i=8, 8, 9, 9}
P(z=8)=1/2
P(z=9)=1/2

H_l=-2*(1/2)$\log_2$(1/2)=1;

H_r=-2*(1/2)$\log_2$(1/2)=1;

$$\overline{H}_l = H_l \cdot P(\text{Node}_l) = 1 \cdot 1/2 = 1/2 \quad \overline{H}_r = H_r \cdot P(\text{Node}_r) = 1 \cdot 1/2 = 1/2$$

$$\overline{H} = \overline{H}_l + \overline{H}_r = 1.0$$

CART Construction using Training Samples (cont.)

- **Entropy for a tree**
  - the sum of weighted entropies for all terminal nodes

$$\overline{H}(T) = \sum_{t \text{ is terminal}} \overline{H}_t(Y)$$

  - It can be show that the above tree-growing (splitting) procedure repeatedly reduces the entropy of the tree
  - The resulting tree has a better classification power

CART Construction using Training Samples (cont.)

- **Splitting Criteria** (for continuous pdf)
  - the likelihood gain is often used instead of the entropy measure
  - Suppose one split divides the data $\boldsymbol{X}$ into two groups $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, which can be respectively represented as two Gaussian distributions $N_1(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $N_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$

$$L_1(\boldsymbol{X}_1 | N_1) = \log \prod_{\boldsymbol{x}_1} N(\boldsymbol{x}_1; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$

$$L_2(\boldsymbol{X}_2 | N_2) = \log \prod_{\boldsymbol{x}_2} N(\boldsymbol{x}_2; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

$$\text{Log likelihood gain at node } t$$

$$\Delta \overline{L}_t(q) = L_1(\boldsymbol{X}_1 | N_1) + L_2(\boldsymbol{X}_2 | N_2) - L_X(\boldsymbol{X} | N)$$

$$= \ldots$$

$$= (a + b)\log|\Sigma| - a \log|\Sigma_1| - b \log|\Sigma_2|$$

a, *b* are the sample counts for $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$