

# Speech Recognition





## Contents:

- Introduction
- Automatic speech recognition
- How does it work?
- Recent improvements
- Current software options
- Future of Speech Recognition

- What is Speech Recognition?
  - Voice Recognition?
- Where can it be used?
  - Dictation
  - System control/navigation
  - Commercial/Industrial applications
  - Hand held digital recorders

## First success story of speech recognition



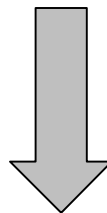
The first machine that recognized speech was a commercial toy named “Radio Rex” which was sold in the 1920’s. Rex was a celluloid dog that moved (by means of a spring) when the spring was released by 500 Hz acoustic energy. Since 500 Hz is roughly the first formant of the vowel [eh] in “Rex”, the dog seemed to come when he was called. (David, Jr. and Selfridge, 1962)

## Timeline of Speech recognition

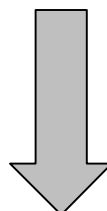
- 1936 - AT & T's Bell labs started study of speech recognition (funded by DARPA)
- 1974 - optical character recognition
- 1975 - text to speech synthesis ( Kurzweil reading machine)
- 1978 - speak and spell toy released by Texas Instruments
- 1980 - Xerox started producing reading machine Text bridge
- 1997 - Dragon Systems produces first continuous speech recognition product

# Evolution of automatic speech recognition

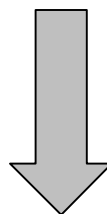
acoustic approach (pre - 1960's)



pattern recognition approach (1960's)



linguistic approach (1970's)



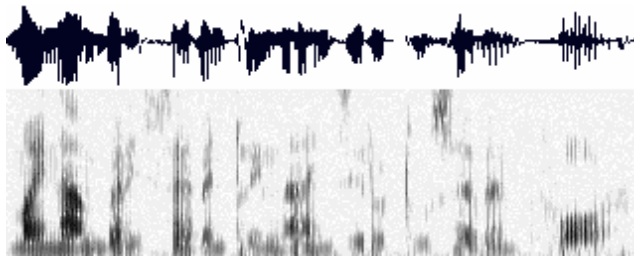
pragmatic approach (1980's)



# Continuous or Discrete?



- Continuous speech
  - dictation



- Discrete speech
  - system controls

# Types of speech recognition



- Isolated words
- Connected words
- Continuous speech
- Spontaneous speech (automatic speech recognition)
- Voice verification and identification

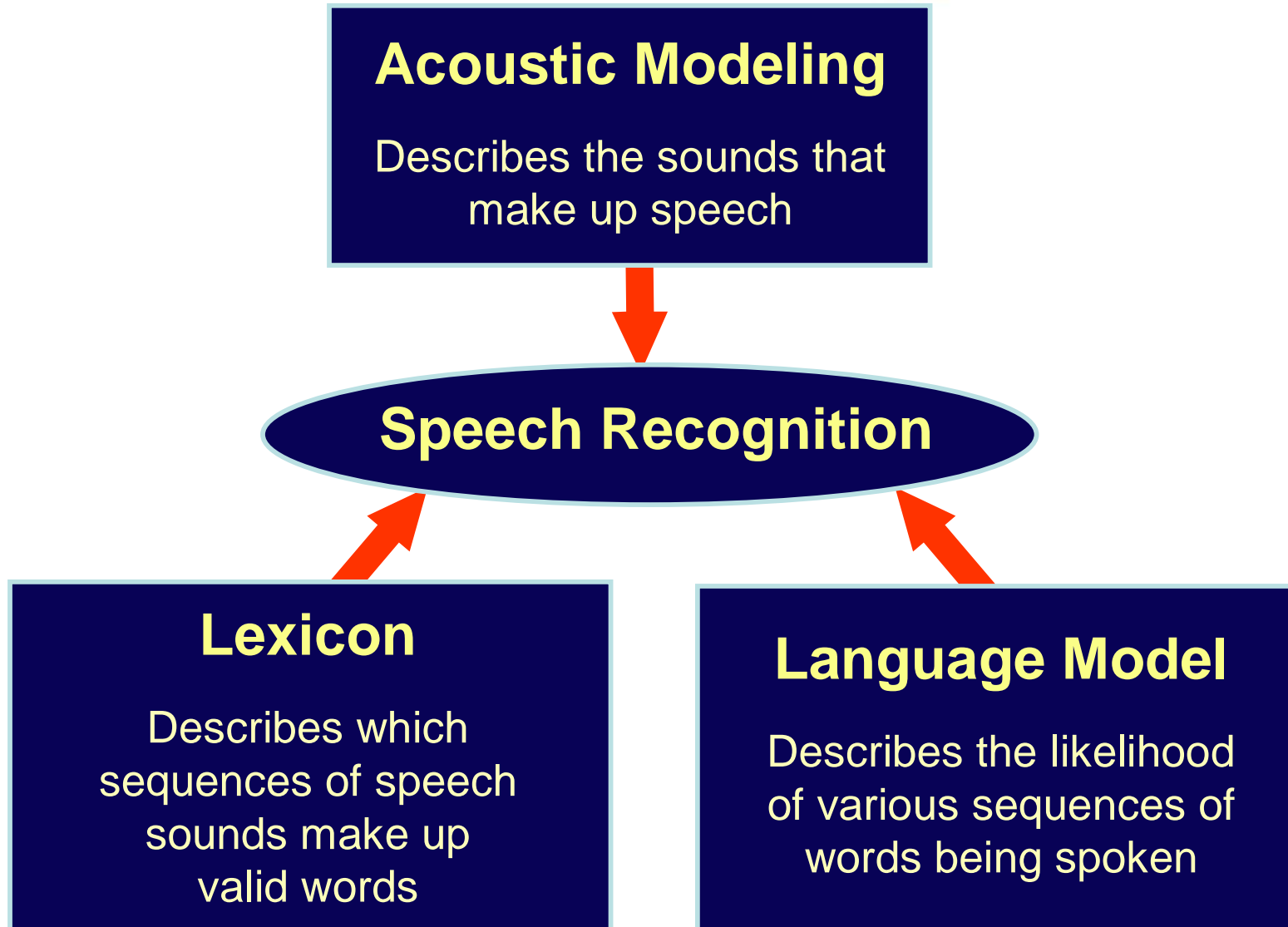


# Speech recognition - uses and applications



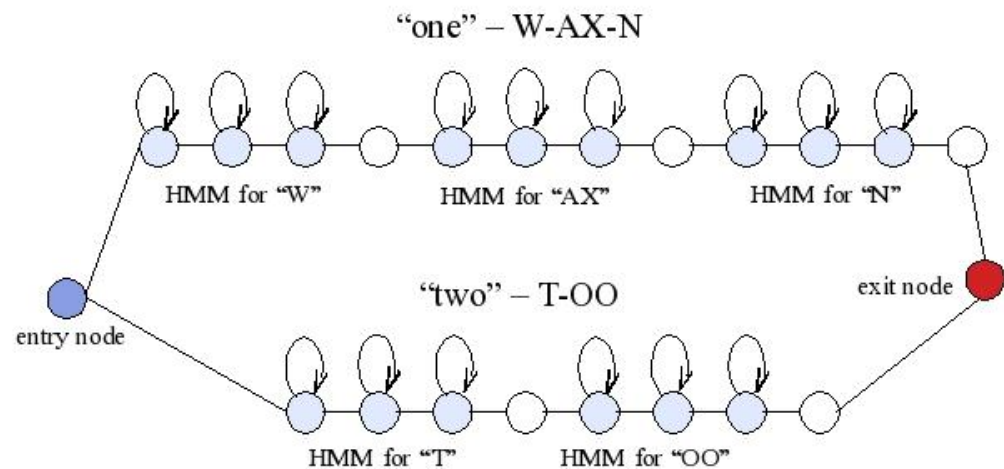
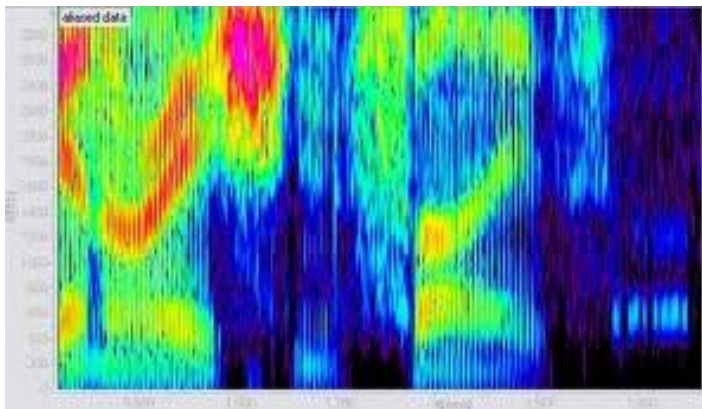
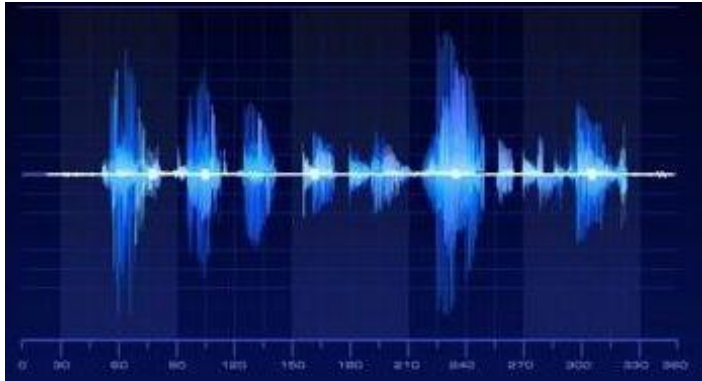
- Dictation
- Command and control
- Telephony
- Medical/disabilities

# Speech Recognition - Block Diagram



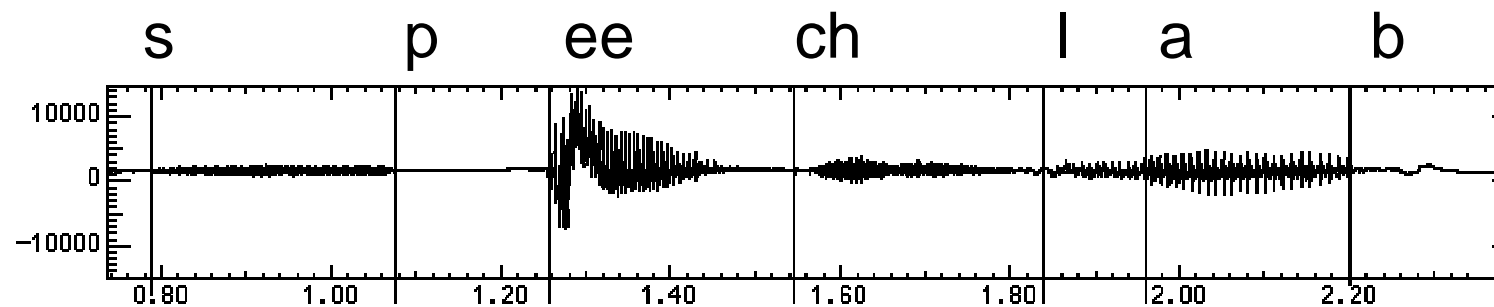
# Speech recognition - main tasks

- Recognition
- Training
- Correction
- Command/Control



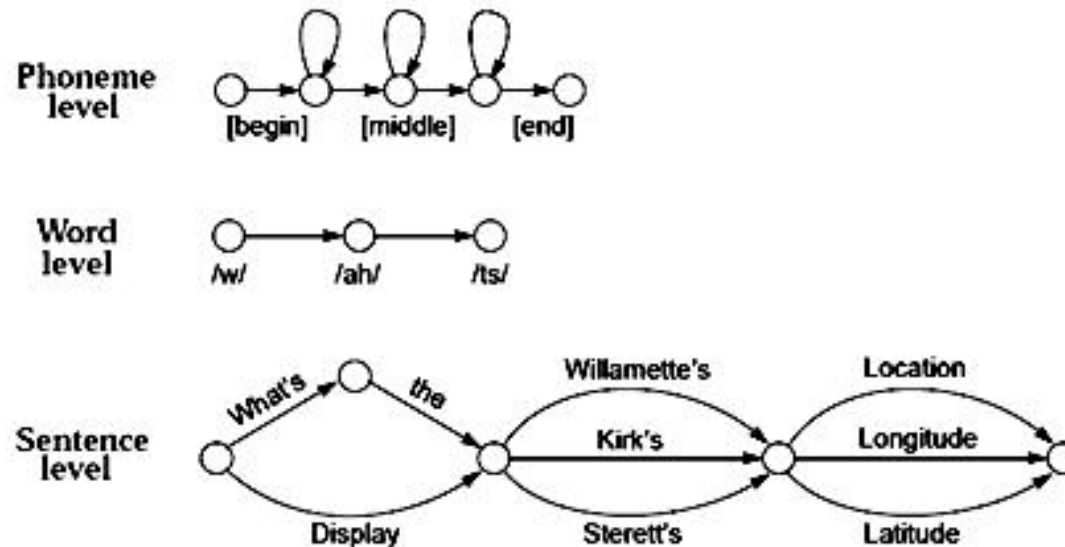
## Acoustic Modeling

- Spoken words: "I think there are....."
- Phonemes: ' ay th-in-nk-kd dh-eh-r aa-r'
- H.M.M.'s: 5 state representation
- Speech Engine



## Language Modeling

- Word context
- Word frequency
- Transition possibilities



# Speech recognition - training

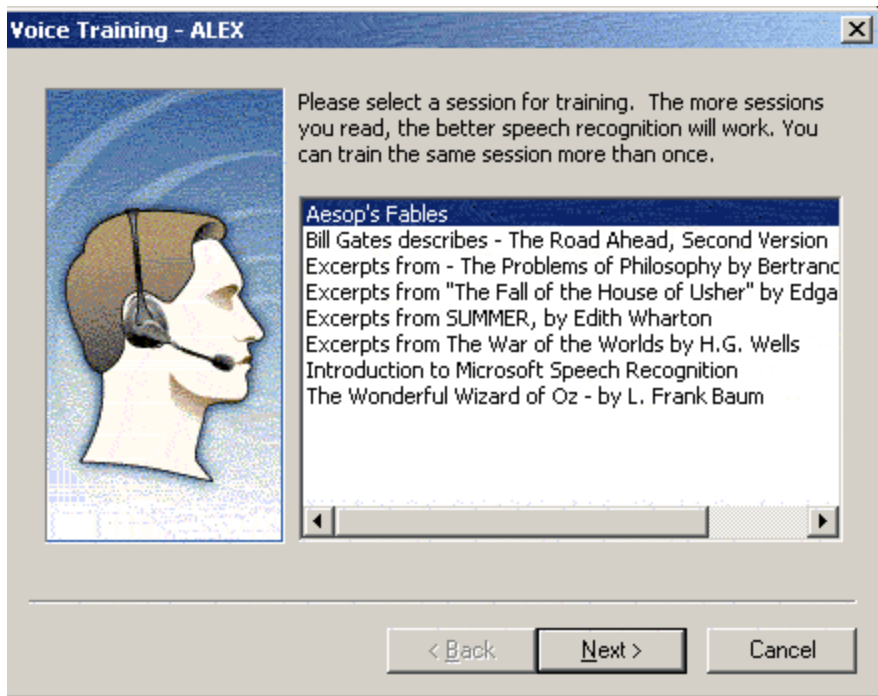
## Voice training

Can be done by:

- Predetermined text segments
- Individual words

Compare new acoustic with old and combines

- More training = better recognition



# Speech recognition - variables



## User specific Voice file

- Voice qualities
- Pronunciation
- Patterns of word use
- Preferred vocabulary



# Speech recognition - challenges

- Ease of use
- Robust performance
- Automatic learning of new words and sounds
- Grammar for spoken language
- Control of synthesized voice quality
- Integrated learning for speech recognition and synthesis





# Automatic speech recognition

- What is the task?
- What is Sound?
- How do humans do it?
- How might computers do it?
- What are the main difficulties?
- Audio processing

## What is the task?

- Getting a computer to understand spoken language
- By “understand” we might mean
  - React appropriately
  - Convert the input speech into another medium, e.g. text
- Several variables impinge on this (see later)



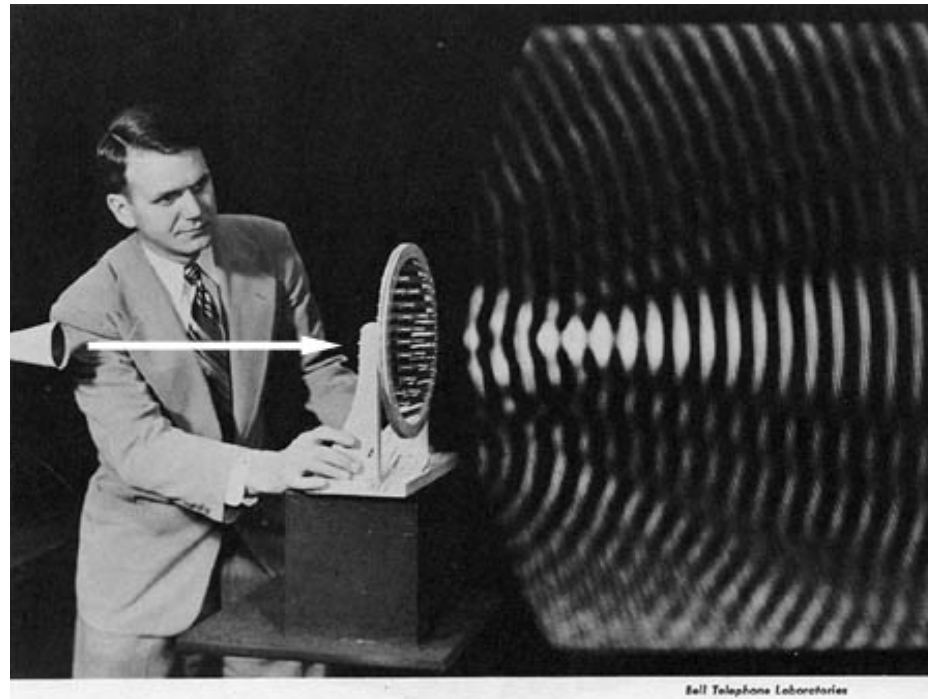
# What is Sound ?

Acoustics is the study of sound.

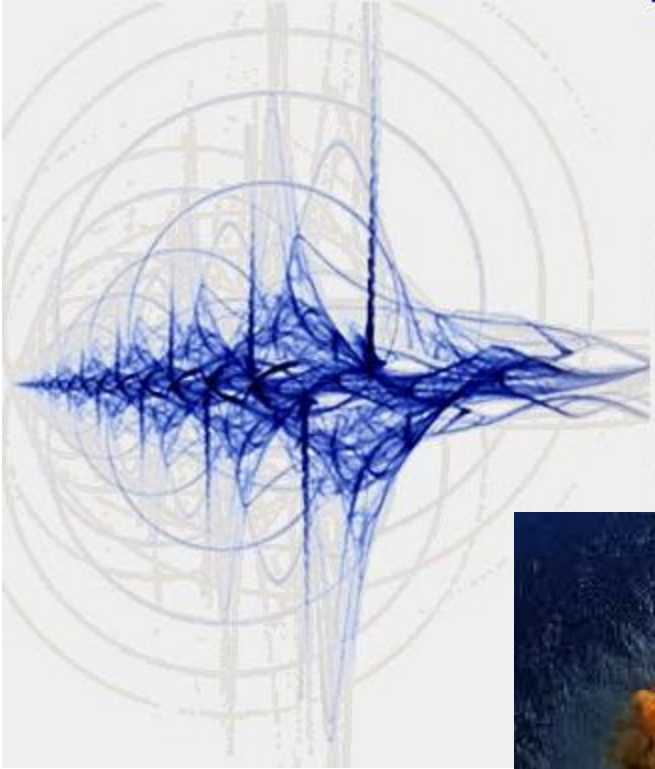
- Physical - sound as a disturbance in the air
- Psychophysical - sound as perceived by the ear
- Sound as stimulus (physical event) & sound as a sensation.
- Pressures changes (in band from 20 Hz to 20 kHz)

Physical terms

- Amplitude
- Frequency
- Spectrum



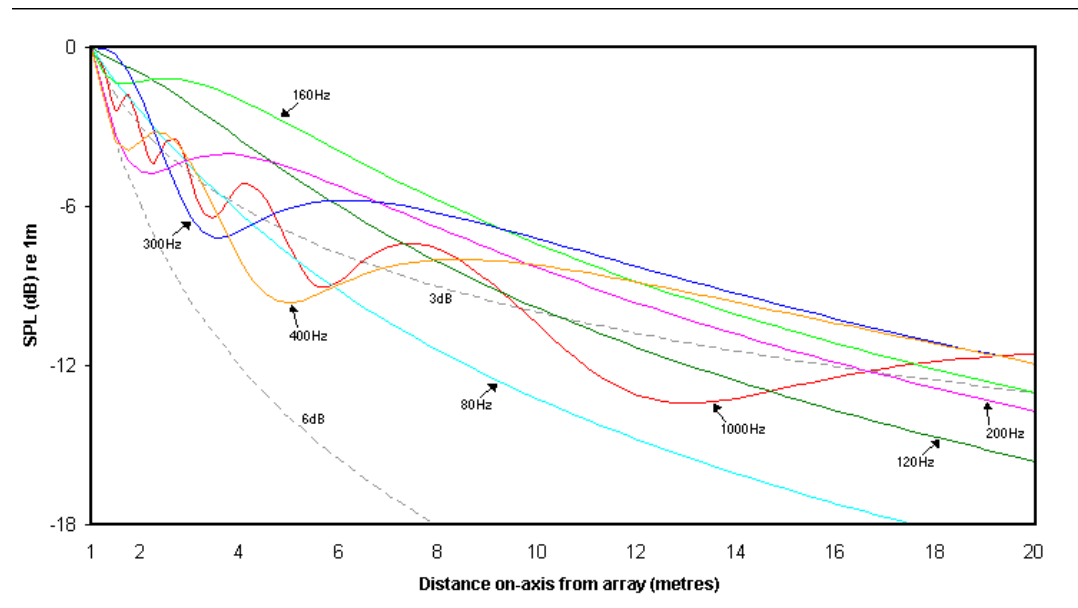
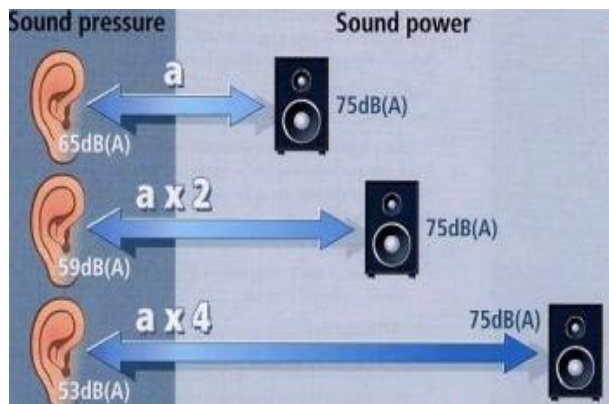
# Sound Waves



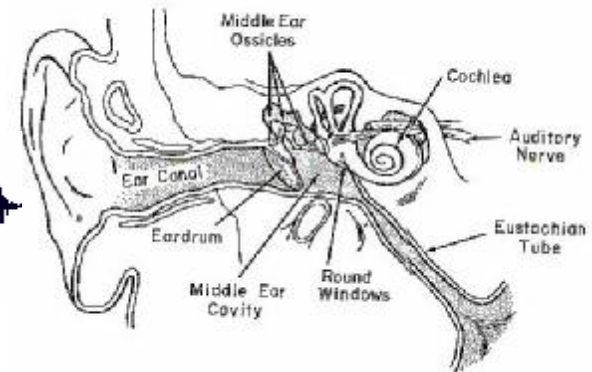
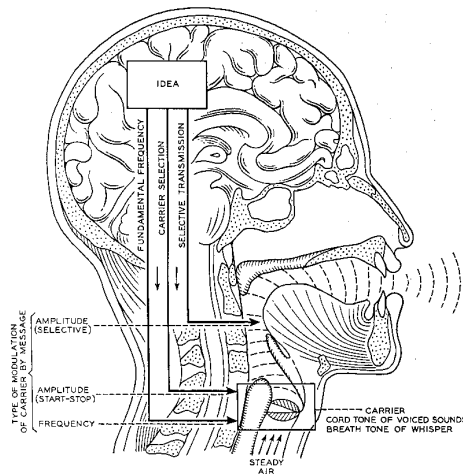


# Sound Waves

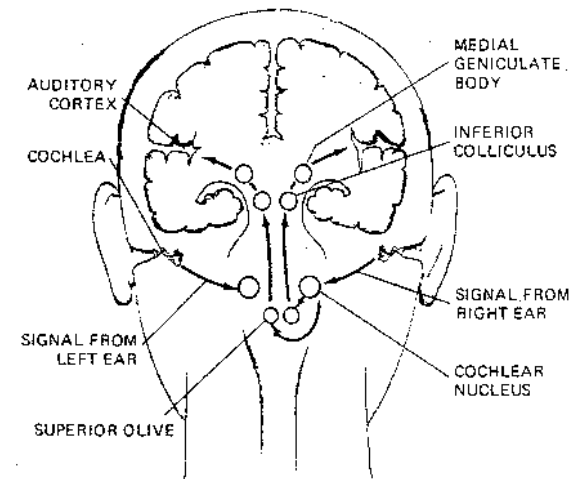
- In a free field, an ideal source of acoustical energy sends out sound of uniform intensity in all directions.  
=> Sound is propagating as a spherical wave.
- Intensity of sound is inversely proportional to the square of the distance (Inverse distance law).
- 6 dB decrease of sound pressure level per doubling the distance.



# How do humans do it?

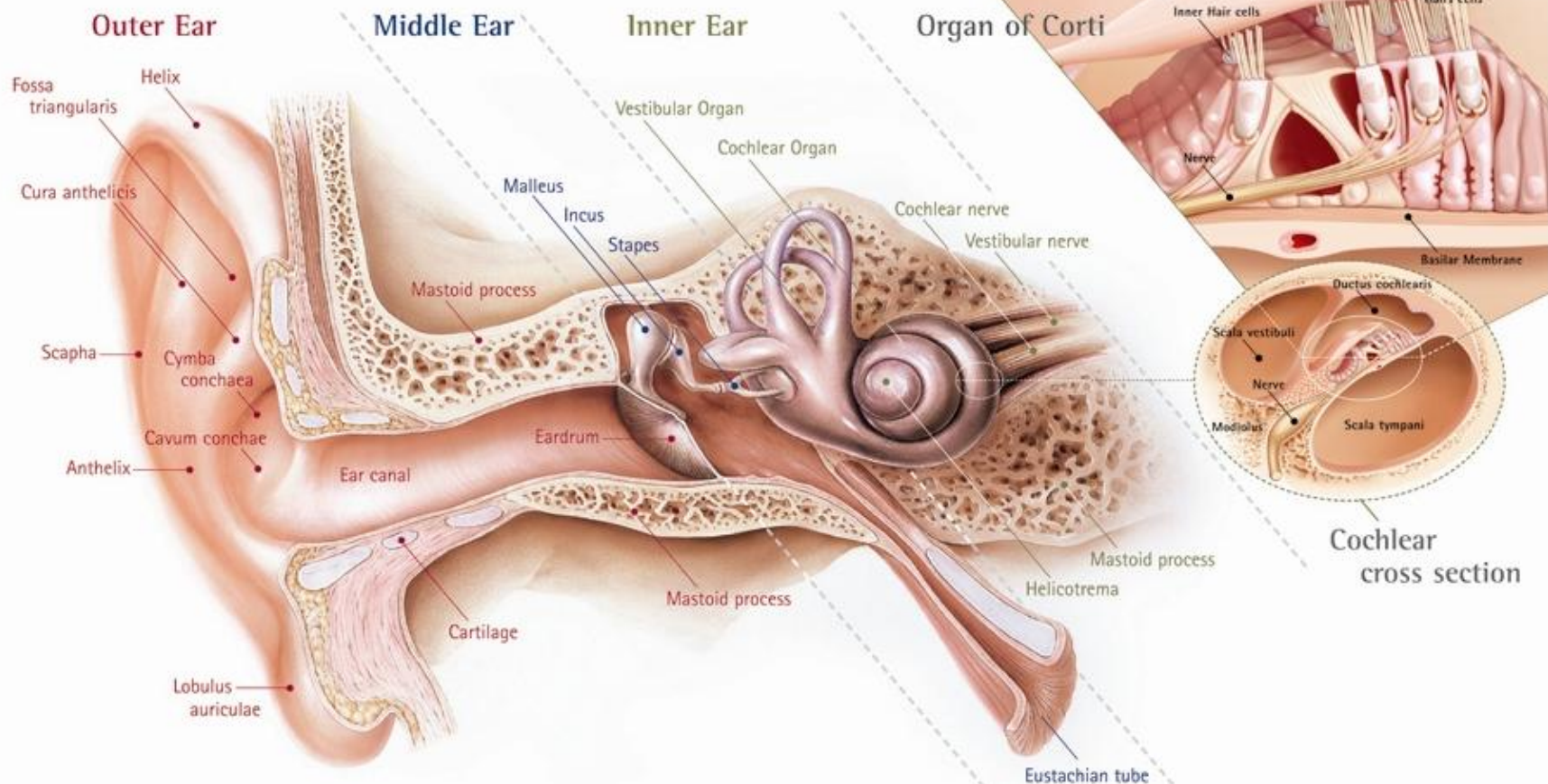


- Articulation produces
- sound waves which
- the ear conveys to the brain
- for processing

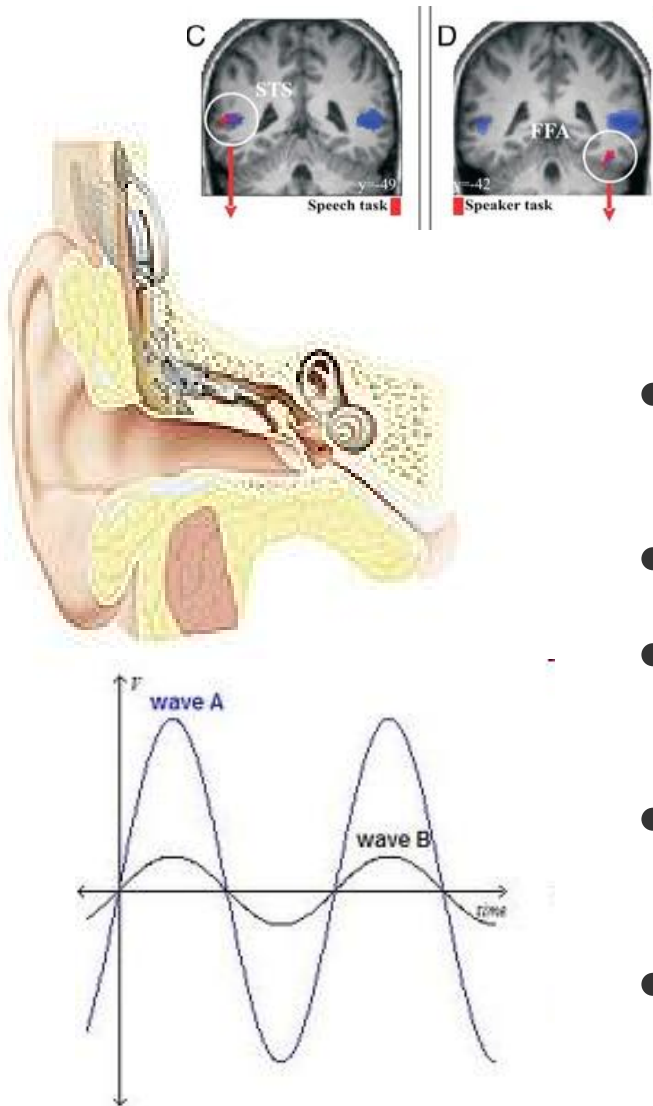


How we hear?

# Anatomy of the Ear



# How we hear?

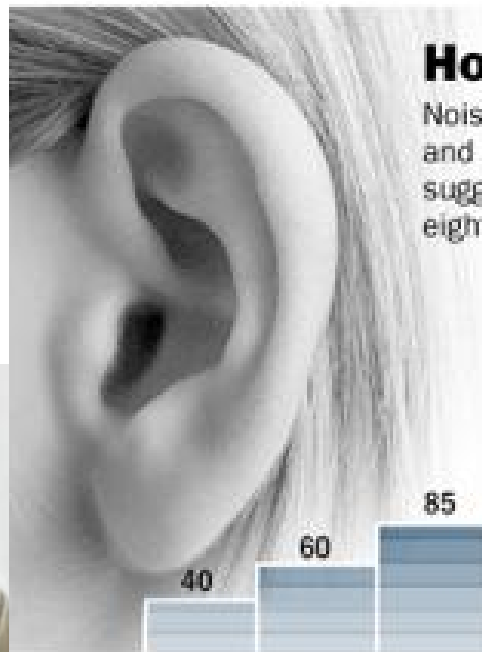


Ear connected to the brain

- left brain: speech
- right brain: music
- Ear's sensitivity to frequency is logarithmic
- Varying frequency response
- Dynamic range is about 120 dB (at 3-4 kHz)
- Frequency discrimination 2 Hz (at 1 kHz)
- Intensity change of 1 dB can be detected.

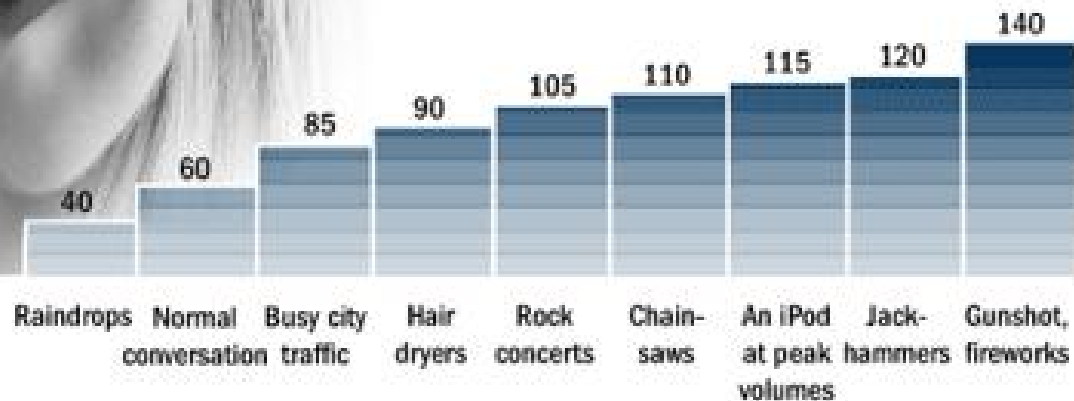


# How we hear?



## How Loud Is Too Loud?

Noise-induced hearing damage is related to the duration and volume of exposure. Government research suggests the safe exposure limit is 85 decibels for eight hours a day. Some common decibel levels:



Sources: [dangerousdecibels.org](http://dangerousdecibels.org); WSJ research



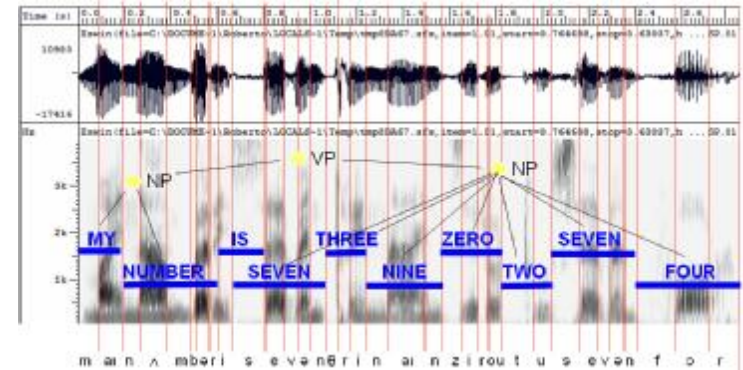
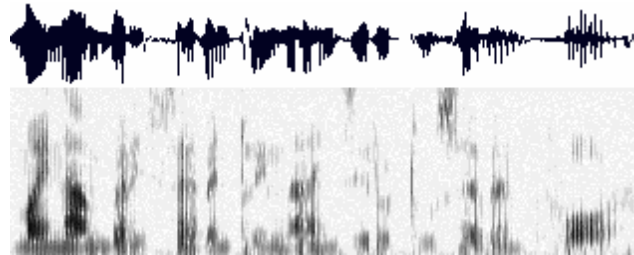
# How might computers do it?



Acoustic waveform



Acoustic signal



Speech recognition

- Digitization
- Acoustic analysis of the speech signal
- Linguistic interpretation

# What are the main difficulties?

- Digitization
  - Converting analogue signal into digital representation
- Signal processing
  - Separating speech from background noise
- Phonetics
  - Variability in human speech
- Phonology
  - Recognizing individual sound distinctions (similar phonemes)
- Lexicology and syntax
  - Disambiguating homophones
  - Features of continuous speech
- Syntax and pragmatics
  - Interpreting prosodic features
- Pragmatics
  - Filtering of performance errors (disfluencies)

# Audio Processing

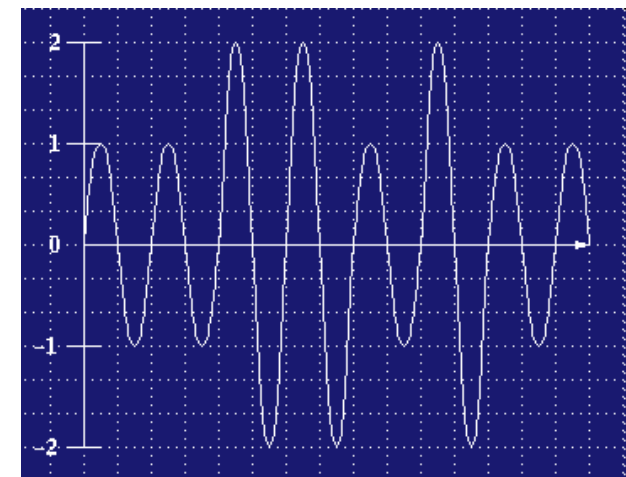
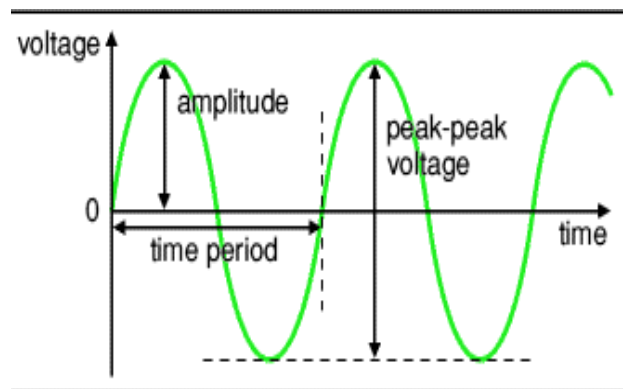


Acoustic pressure waves



Microphone

Varying electrical voltage

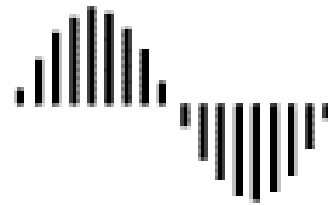


- Analogue to digital conversion
- Sampling and quantizing
- Use filters to measure energy levels for various points on the frequency spectrum
- Knowing the relative importance of different frequency bands (for speech) makes this process more efficient
- E.g. high frequency sounds are less informative, so can be sampled using a broader bandwidth (log scale)

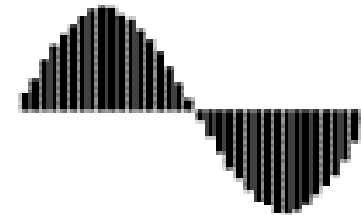
# Digital Sampling



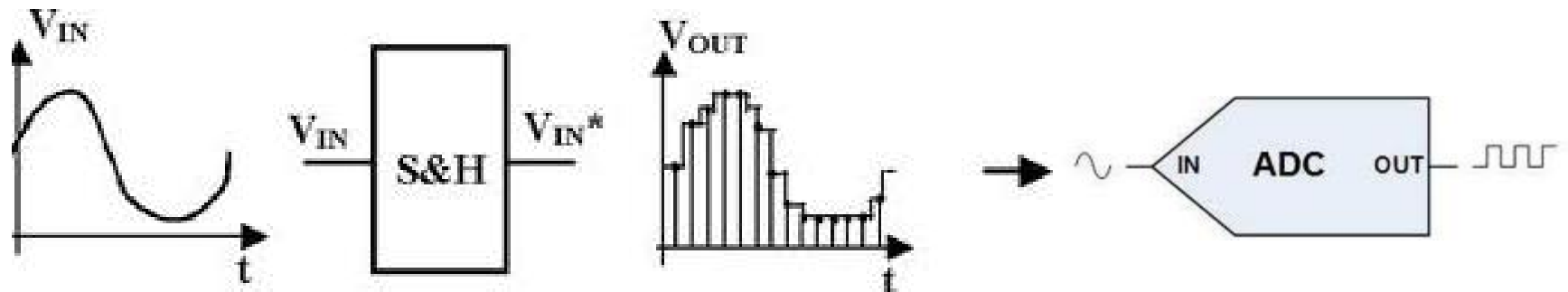
Analog Signal



Digital Signal at 64 kbps

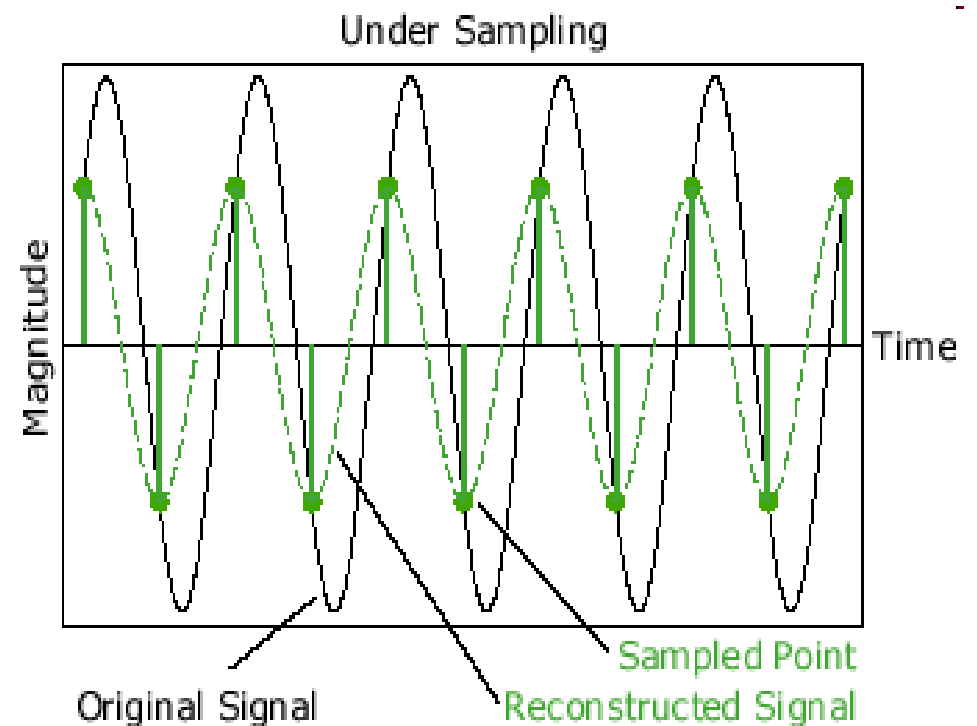
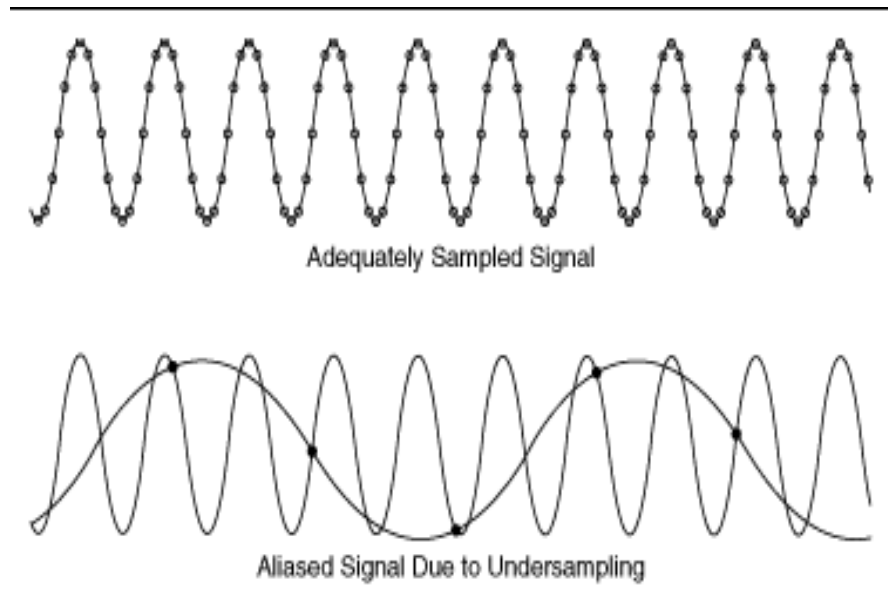


Digital Signal at 128 kbps

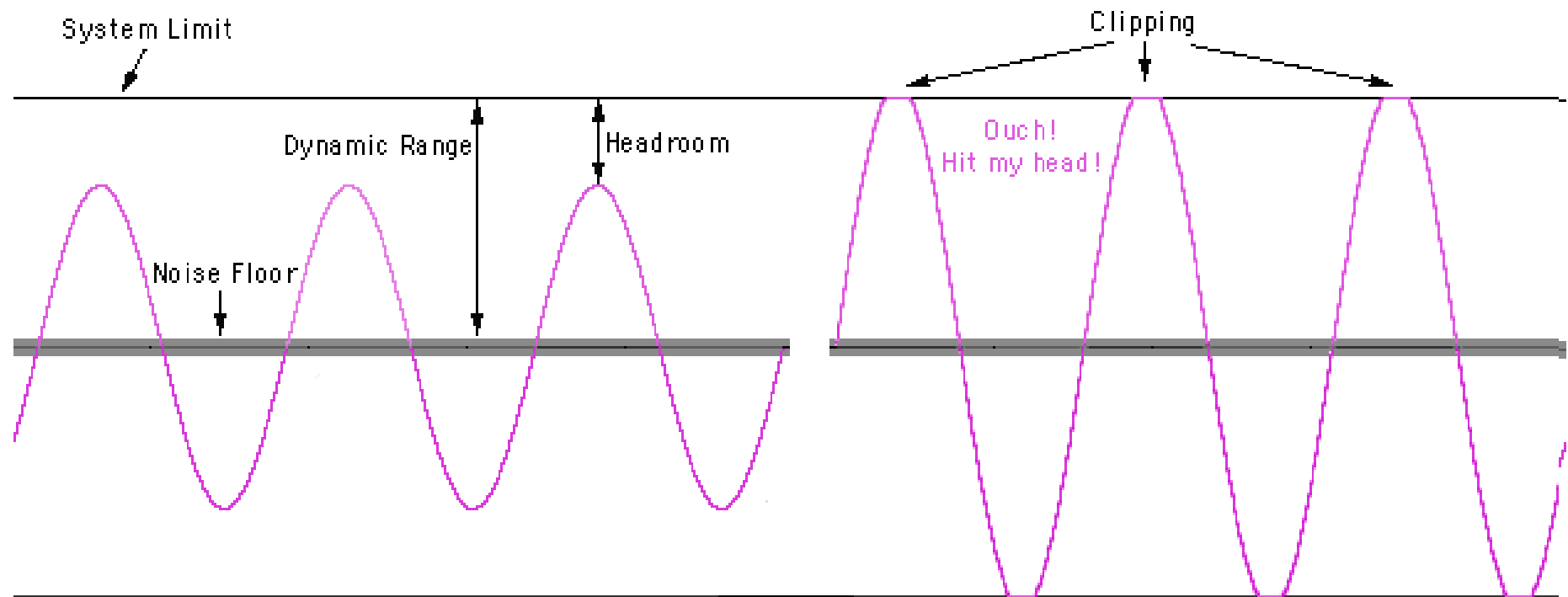




# Undersampling

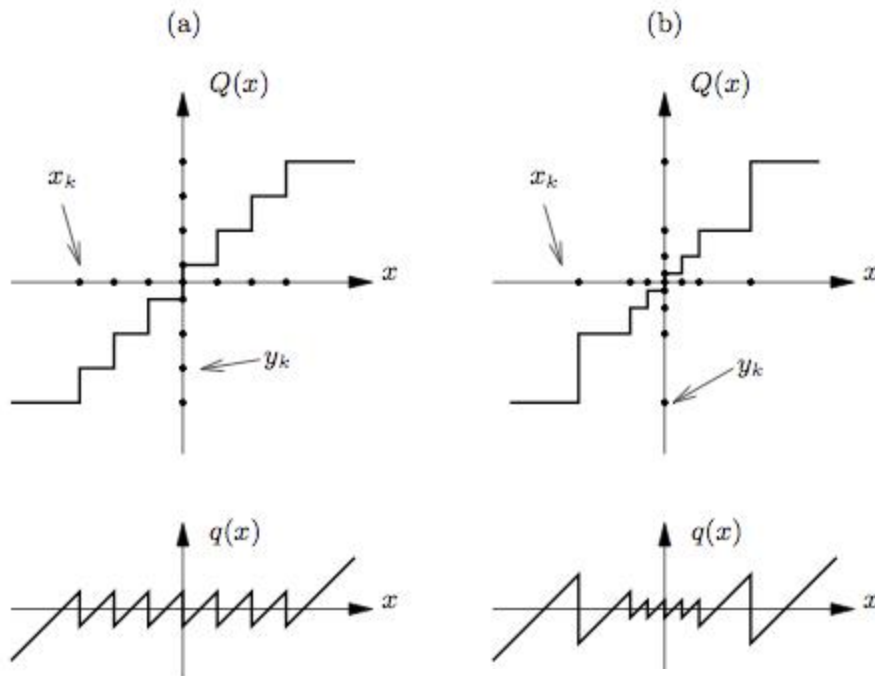


# Clipping

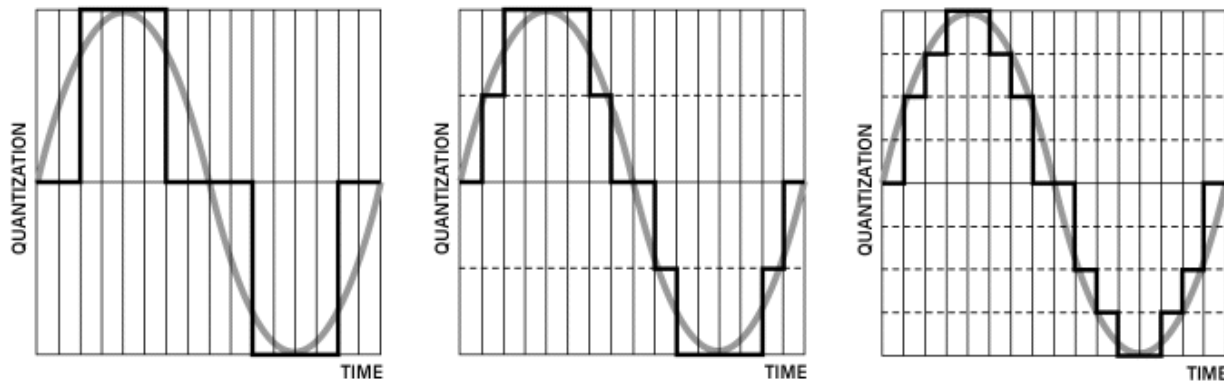




# Quantization



(a) Uniform and  
(b) non-uniform quantization  
 $Q(x)$  and quantization error  $q(x)$



- *Sampling* is dictated by the Nyquist sampling theorem which states how quickly samples must be taken to ensure an accurate representation of the analog signal.

$$f_s \geq 2f$$

or

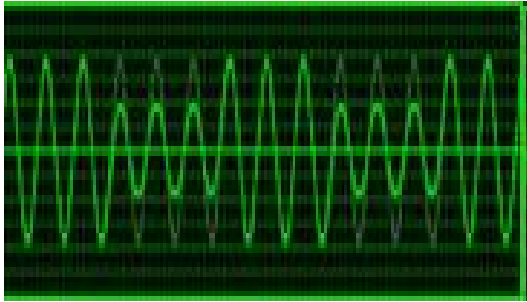
$$T_s \leq \frac{T}{2}$$

- The Nyquist sampling theorem states that the sampling frequency must be **two** times greater than the highest frequency in the original analog signal.

# Dithering a Sampled Signal

- Analog signal added to the signal to remove the artifacts of quantization error.
- Dither causes the audio signal to always move between quantization levels.
- Otherwise, a low level signal would be encoded as a square wave  
=> granulation noise.
- Dithered, the A/D converter output is signal + noise  
=> perceptually preferred,  
since noise is better tolerated than distortion.
- Amplitude of dither signal:  
high dither amplitudes more easily remove quantization artifacts  
too much dither decreases the signal-to-noise ratio

# Common Sound Sampling Parameters



- Common Sampling Rates
  - 8KHz (Phone) or 8.012820513kHz (Phone, NeXT)
  - 11.025kHz (1/4 CD std)
  - 16kHz (G.722 std)
  - 22.05kHz (1/2 CD std)
  - 44.1kHz (CD, DAT)
  - 48kHz (DAT)
- Bits per Sample
  - 8 or 16
- Number of Channels
  - mono/stereo/quad/ etc.

## Sampling Audio Data Rates

Quality	Format (examples)	Transfer Rate	Disk Space 1 hour	Disk Space 100,000 hours
Netcasting	RealAudio	20 Kbit/s	8.8 MByte	0.9 TByte
Preview	RealAudio	80 Kbit/s	35.2 MByte	3.5 Tbyte
Preview	MPEG Layer 3 (MP3)	192 Kbit/s	84.4 MByte	8.4 Tbyte
Broadcasting or Editing	MPEG Layer 2	384 Kbit/s	168.8 MByte	16.9 Tbyte
Archive (uncompressed)	Waveform PCM	1538 Kbit/s	675.9 MByte	67.6 TByte

# Space/Storage Requirements

## 1 Minute of Sound



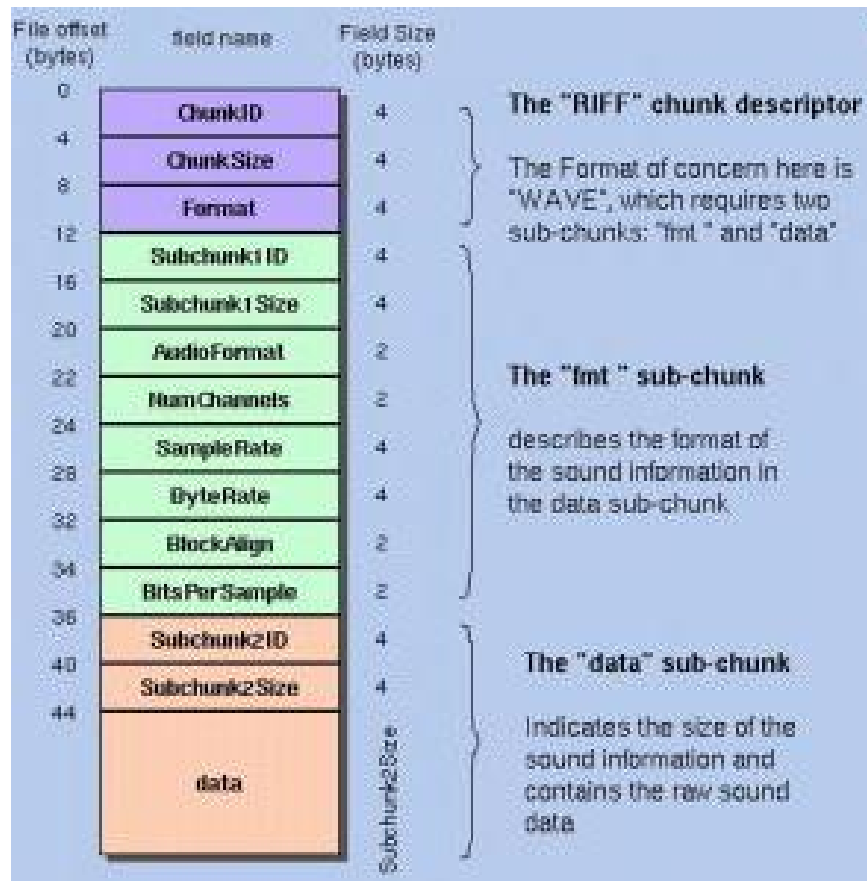
Type	Mono	Mono	Stereo	Stereo
Resolution	8 bit	16 bit	8 bit	16 bit
Sampling Rate				
44.1k	2646k	5292k	5292k	10584k
22.05k	1323k	2646k	2646k	5292k
11.025k	661.5k	1323k	1323k	2646k
8k	480k	960k	960k	1920k

# Sound File Formats



- Mulaw (Sun, NeXT) .au
- RIFF (Resource Interchange File Format)
  - MS WAV and .AVI
- MPEG Audio Layer (MPEG) .mpa .mp3
- AIFF (Apple, SGI) .aiff .aif
- HCOM (Mac) .hcom
- SND (Sun, NeXT) .snd
- VOC (Soundblaster card proprietary standard) .VOC
- AND MANY OTHERS!

# What's in a Sound File Format



## - Header Information

- Magic Cookie
- Sampling Rate
- Bits/Sample
- Channels
- Byte Order
- Endian
- Compression type

## - Data



# Example File Format (NIST SPHERE)

NIST\_1A

1024

sample\_rate -i 16000

channel\_count -i 1

sample\_n\_bytes -i 2

sample\_byte\_format -s2 10

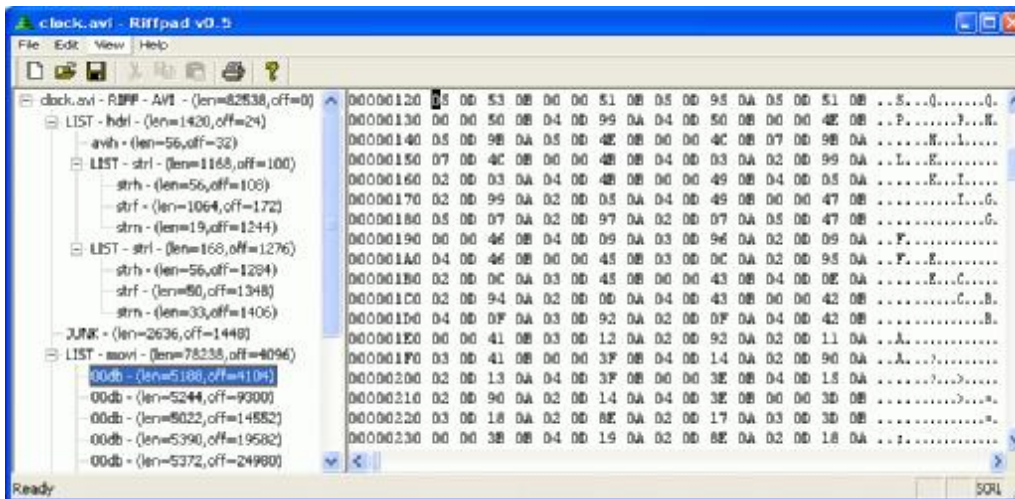
sample\_sig\_bits -i 16

sample\_count -i 594400

sample\_coding -s3 pcm

sample\_checksum -i 20129

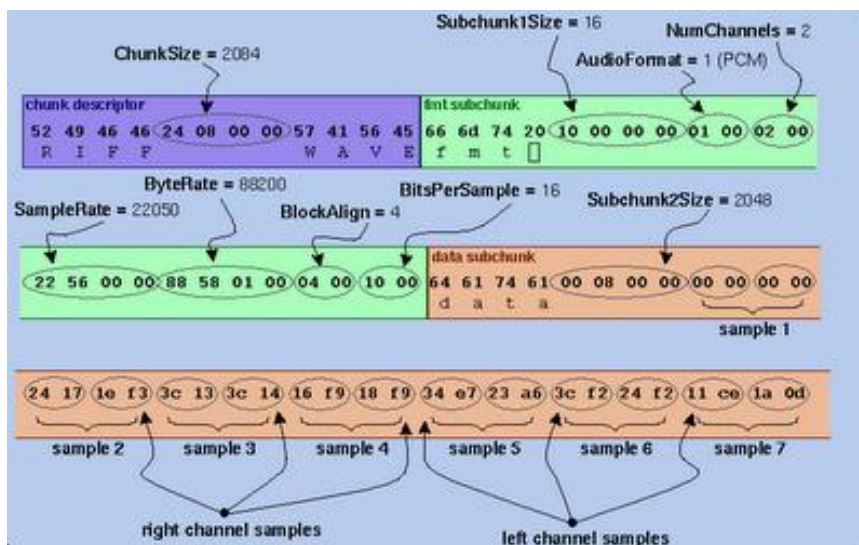
end\_head



# WAV file format (Microsoft) RIFF

## A collection of data chunks.

- Each chunk has a 32-bit Id
- followed by a 32-bit chunk length
- followed by the chunk data.



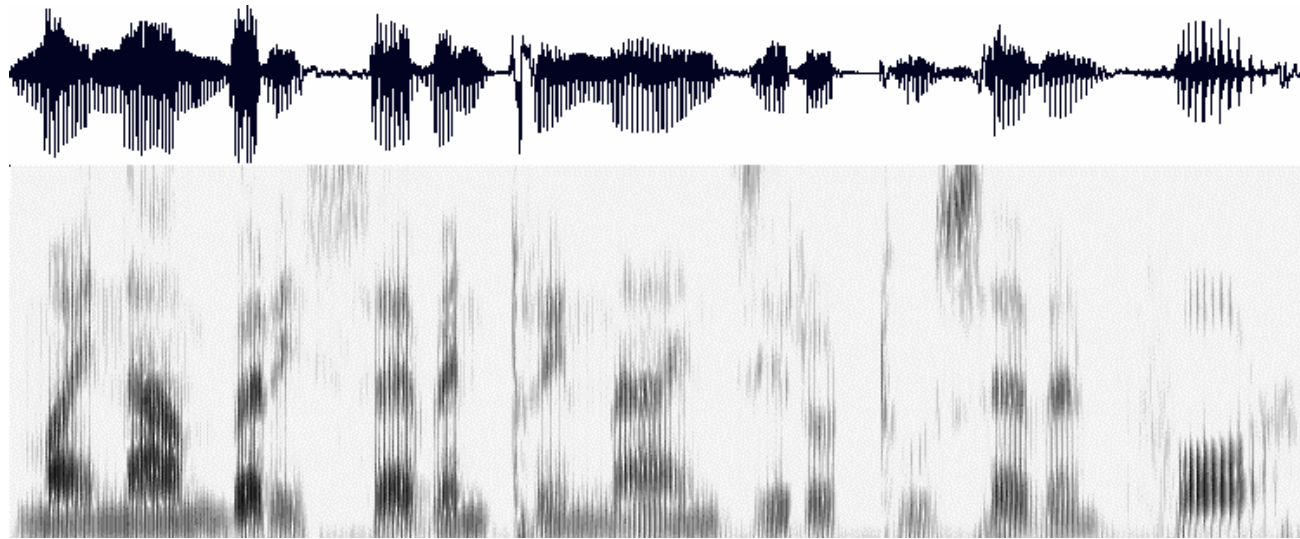
- 0x00 chunk id 'RIFF'
- 0x04 chunk size (32-bits)
- 0x08 wave chunk id 'WAVE'
- 0x0C format chunk id 'fmt '
- 0x10 format chunk size (32-bits)
- 0x14 format tag (currently pcm)
- 0x16 number of channels 1=mono, 2=stereo
- 0x18 sample rate in hz
- 0x1C average bytes per second
- 0x20 number of bytes per sample
  - 1 = 8-bit mono
  - 2 = 8-bit stereo or 16-bit mono
  - 4 = 16-bit stereo
- 0x22 number of bits in a sample
- 0x24 data chunk id 'data'
- 0x28 length of data chunk (32-bits)
- 0x2C Sample data

# Digital Audio Today

- Analog elements in the audio chain are replaced with digital elements.
- 16-bit wordlength, 32/44.1/48 kHz sampling rates.
- Mostly linear signal processing.
- Wide range of digital formats and storage media.
- Rapid development of technology  
=> better SNR, phase and linearity.
- Rapid increase of signal processing power  
=> possibility to implement new, complex features.
- Soon: Digital radio (satellite), HDTV

# Separating speech from background noise

- Noise cancelling microphones
  - Two mics, one facing speaker, the other facing away
  - Ambient noise is roughly same for both mics
- Knowing which bits of the signal relate to speech
  - Spectrograph analysis



## Variability in individuals' speech

- Variation among speakers due to
  - Vocal range (f0, and pitch range - see later)
  - Voice quality (growl, whisper, physiological elements such as nasality, adenoidality, etc)
  - ACCENT !!! (especially vowel systems, but also consonants, allophones, etc.)
- Variation within speakers due to
  - Health, emotional state
  - Ambient conditions
- Speech style: formal read vs spontaneous

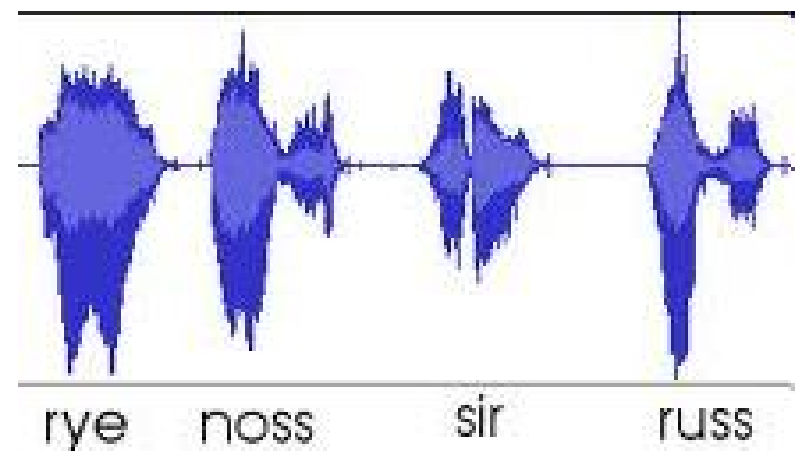
# Speaker-(in)dependent systems

- Speaker-dependent systems
  - Require “training” to “teach” the system your individual idiosyncracies
    - The more the merrier, but typically nowadays 5 or 10 minutes is enough
    - User asked to pronounce some key words which allow computer to infer details of the user’s accent and voice
    - Fortunately, languages are generally systematic
  - More robust
  - But less convenient
  - And obviously less portable
- Speaker-independent systems
  - Language coverage is reduced to compensate need to be flexible in phoneme identification
  - Clever compromise is to learn on the fly



## Identifying phonemes

- Differences between some phonemes are sometimes very small
  - May be reflected in speech signal (eg vowels have more or less distinctive f1 and f2)
  - Often show up in coarticulation effects (transition to next sound)
    - e.g. aspiration of voiceless stops in English
  - Allophonic variation



## Disambiguating homophones

- Mostly differences are recognised by humans by context and need to make sense

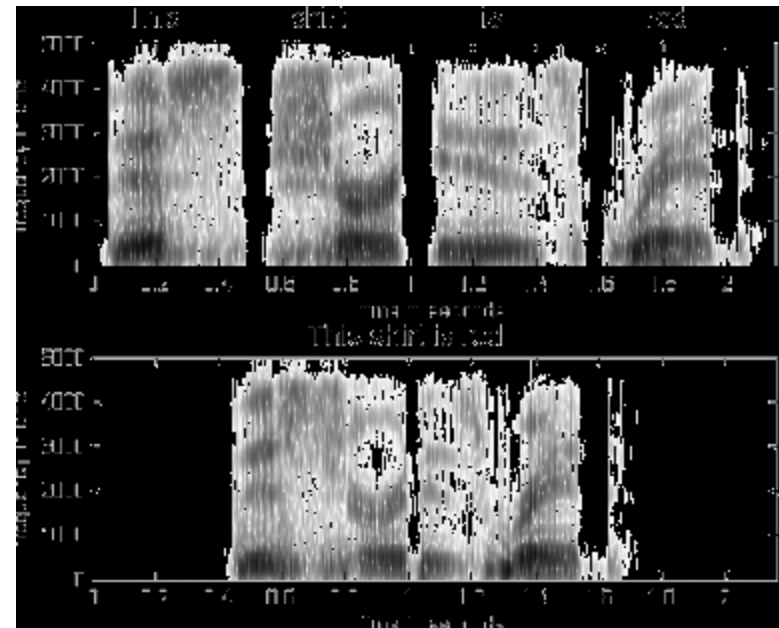
*It's hard to wreck a nice beach*

*What dime's a neck's drain to stop port?*

- Systems can only recognize words that are in their lexicon, so limiting the lexicon is an obvious ploy
- Some ASR systems include a grammar which can help disambiguation

## (Dis)continuous speech

- Discontinuous speech much easier to recognize
  - Single words tend to be pronounced more clearly
- Continuous speech involves contextual coarticulation effects
  - Weak forms
  - Assimilation
  - Contractions

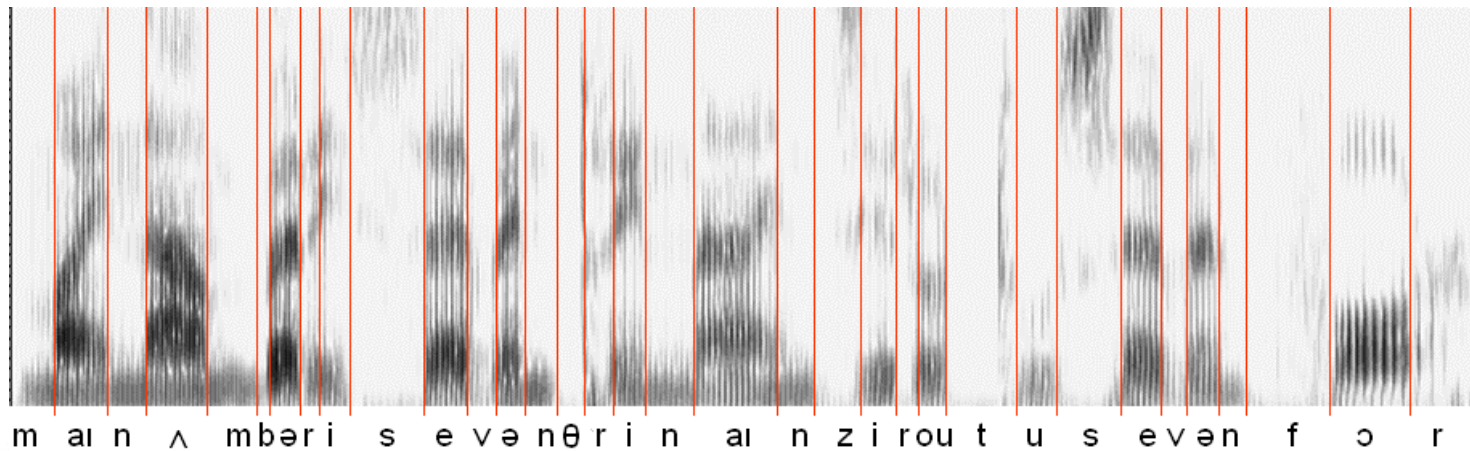


## Interpreting prosodic features

- Pitch, length and loudness are used to indicate “stress”
- All of these are relative
  - On a speaker-by-speaker basis
  - And in relation to context
- Pitch and length are phonemic in some languages

- Pitch contour can be extracted from speech signal
  - But pitch differences are relative
  - One man's high is another (wo)man's low
  - Pitch range is variable
- Pitch contributes to intonation
  - But has other functions in tone languages
- Intonation can convey meaning

- Length is easy to measure but difficult to interpret
- Again, length is relative
- It is phonemic in many languages
- Speech rate is not constant - slows down at the end of a sentence





- Loudness is easy to measure but difficult to interpret
- Again, loudness is relative



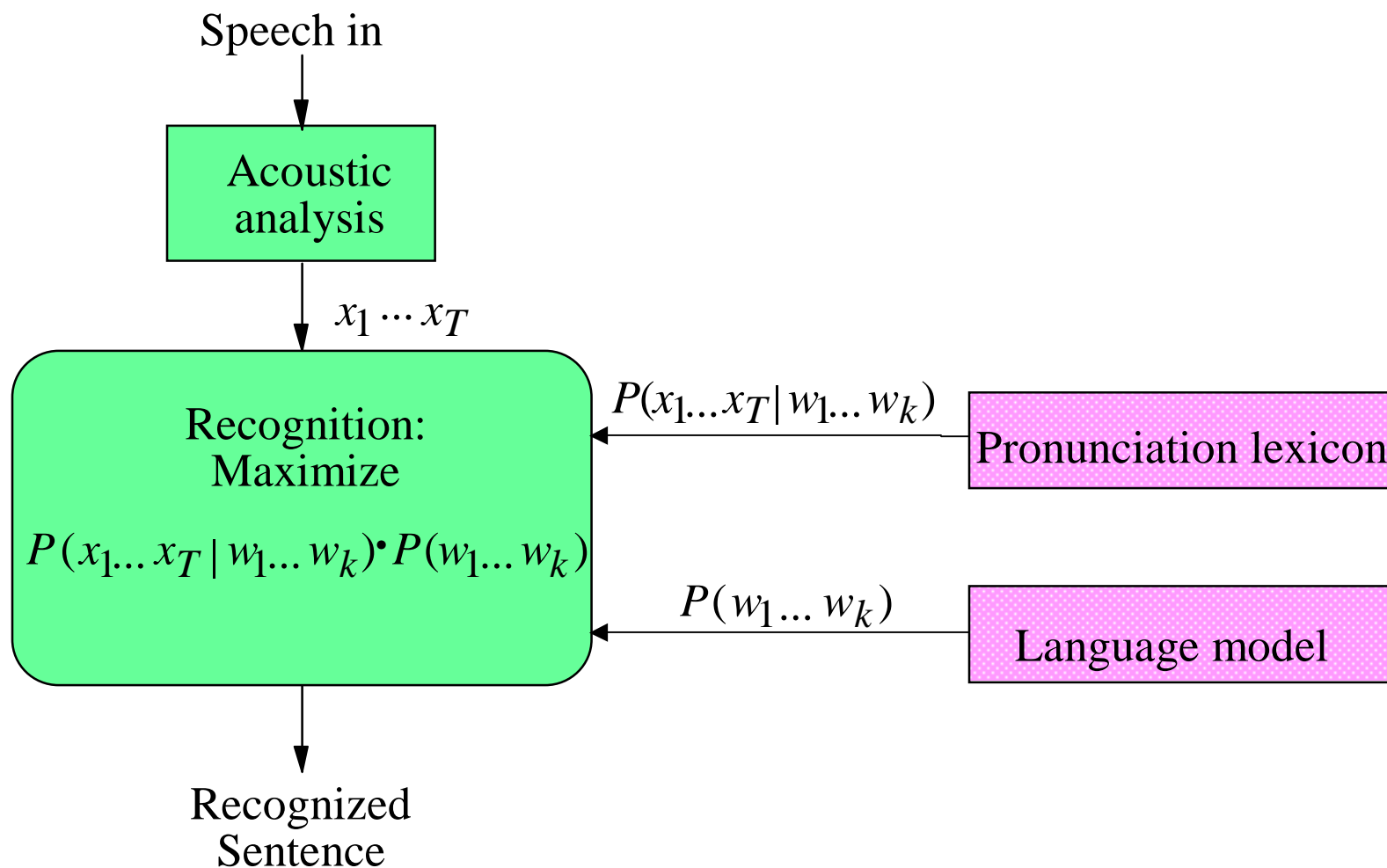
- Performance “errors” include
  - Non-speech sounds
  - Hesitations
  - False starts, repetitions
- Filtering implies handling at syntactic level or above
- Some disfluencies are deliberate and have pragmatic effect – this is not something we can handle in the near future



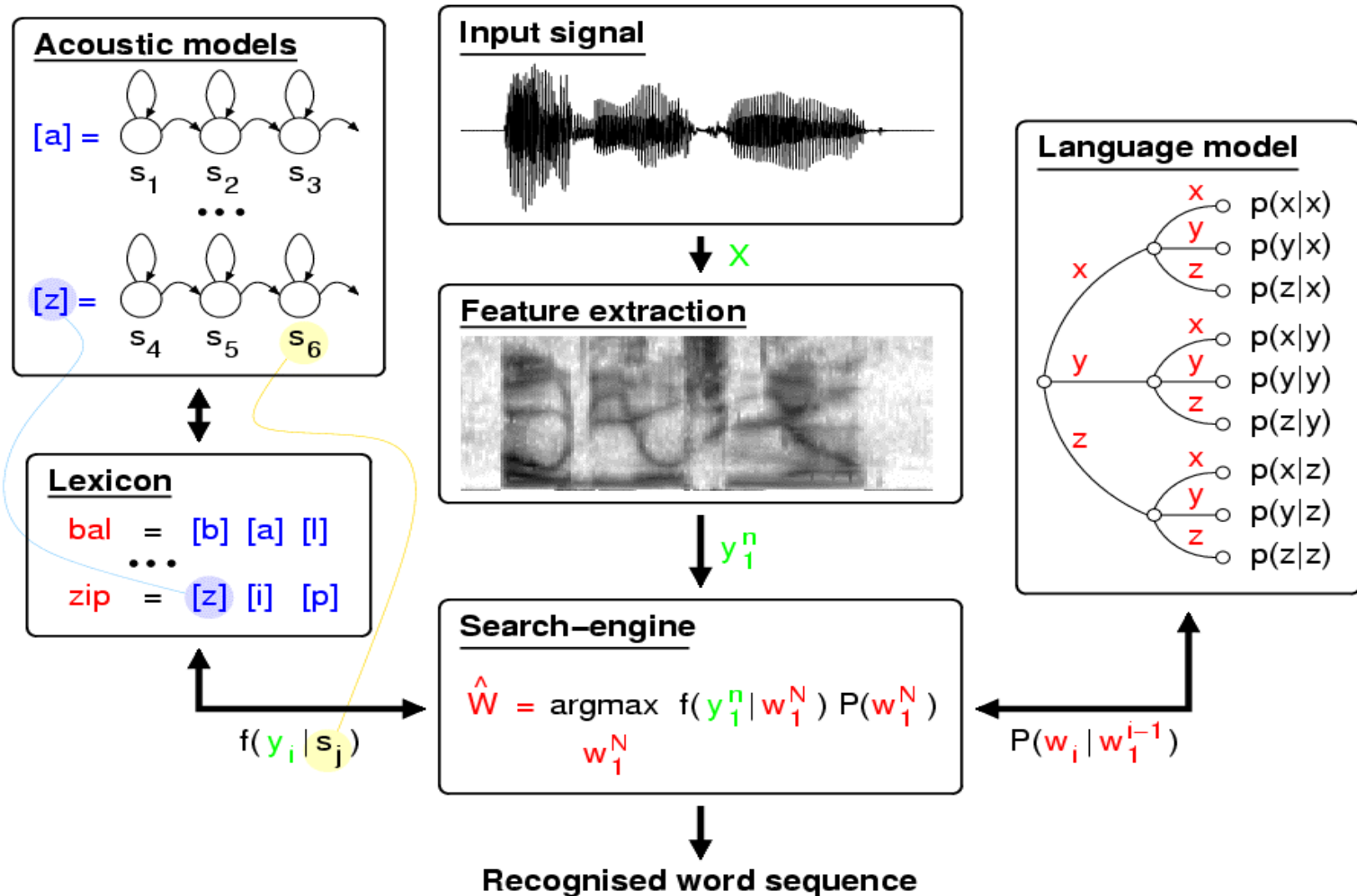
# Speech recognition system

- Acoustic modeling
- Lexicon
- Lenguaje Model
- Speech recognition

# Mechanism of state-of-the-art speech recognizers



*It's not easy to wreck a nice beach.  
It's not easy to recognize speech.*



- Acoustic and Lexical Models
  - Analyse training data in terms of relevant features
  - Learn from large amount of data different possibilities
    - different phone sequences for a given word
    - different combinations of elements of the speech signal for a given phone/phoneme
  - Combine these into a Hidden Markov Model expressing the probabilities





## Approaches to ASR

- Template matching
- Knowledge-based (or rule-based) approach
- Statistical approach:
  - Noisy channel model + machine learning

## Template-based approach

- Store examples of units (words, phonemes), then find the example that most closely fits the input
- Extract features from speech signal, then it's "just" a complex similarity matching problem, using solutions developed for all sorts of applications
- OK for discrete utterances, and a single user

## Template-based approach

- Hard to distinguish very similar templates
- And quickly degrades when input differs from templates
- Therefore needs techniques to mitigate this degradation:
  - More subtle matching techniques
  - Multiple templates which are aggregated
- Taken together, these suggested ...

## Rule-based approach

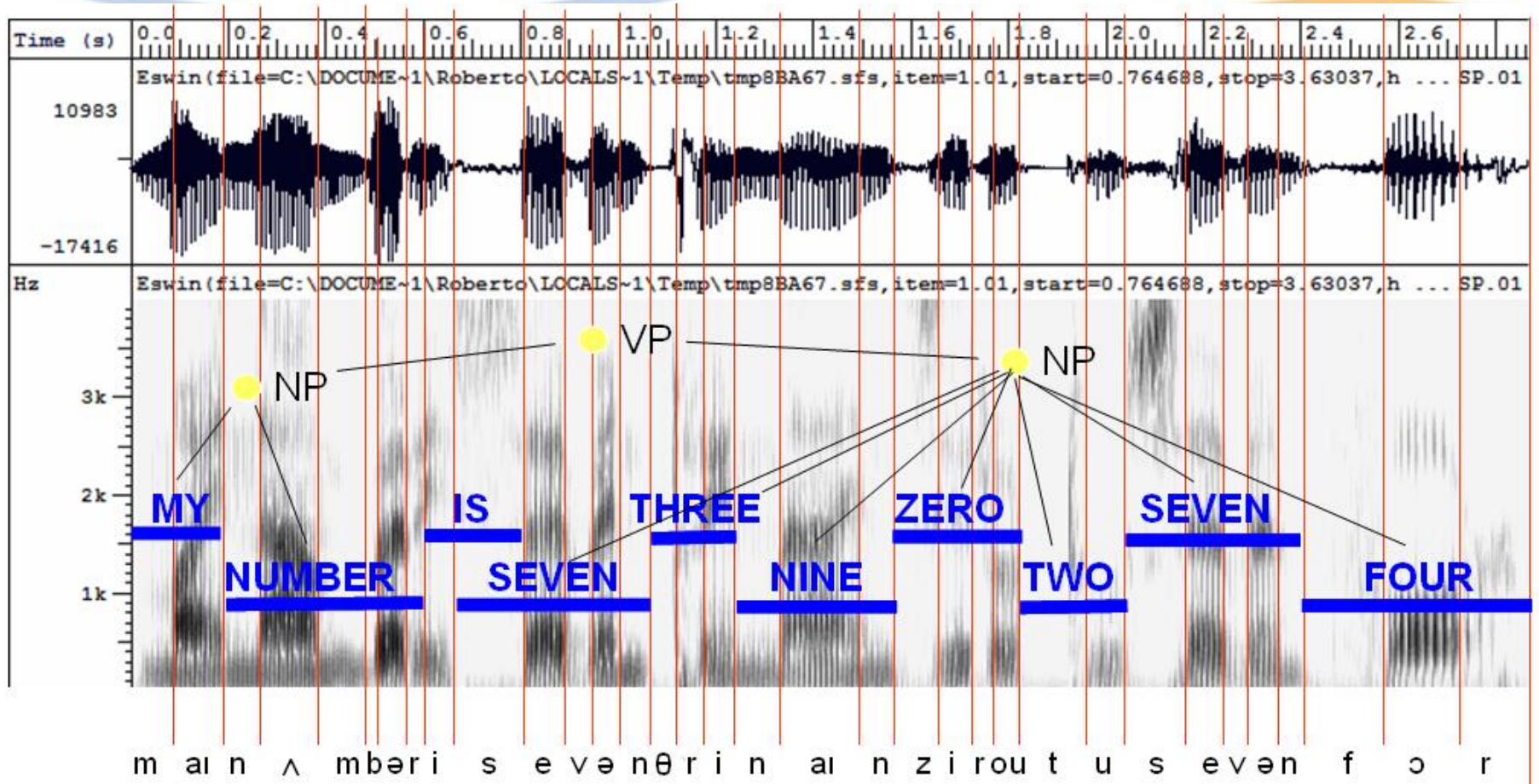
- Use knowledge of phonetics and linguistics to guide search process
- Templates are replaced by rules expressing everything (anything) that might help to decode:
  - Phonetics, phonology, phonotactics
  - Syntax
  - Pragmatics

## Rule-based approach

- Typical approach is based on “blackboard” architecture:
  - At each decision point, lay out the possibilities
  - Apply rules to determine which sequences are permitted
- Poor performance due to
  - Difficulty to express rules
  - Difficulty to make rules interact
  - Difficulty to know how to improve the system

s k i: ∫  
∫ h p iə t f  
t r h  
s

# Rule-based approach



- Identify individual phonemes
- Identify words
- Identify sentence structure and/or meaning
- Interpret prosodic features (pitch, loudness, length)



## Statistics-based approach

- Can be seen as extension of template-based approach, using more powerful mathematical and statistical tools
- Sometimes seen as “anti-linguistic” approach
  - Fred Jelinek (IBM, 1988): “Every time I fire a linguist my system improves”



## Statistics-based approach

- Collect a large corpus of transcribed speech recordings
- Train the computer to learn the correspondences (“machine learning”)
- At run time, apply statistical processes to search through the space of all possible solutions, and pick the statistically most likely one

# The Speech Recognition Problem

- We want to predict a sentence given an acoustic sequence:

$$s^* = \arg \max_s P(s | A)$$

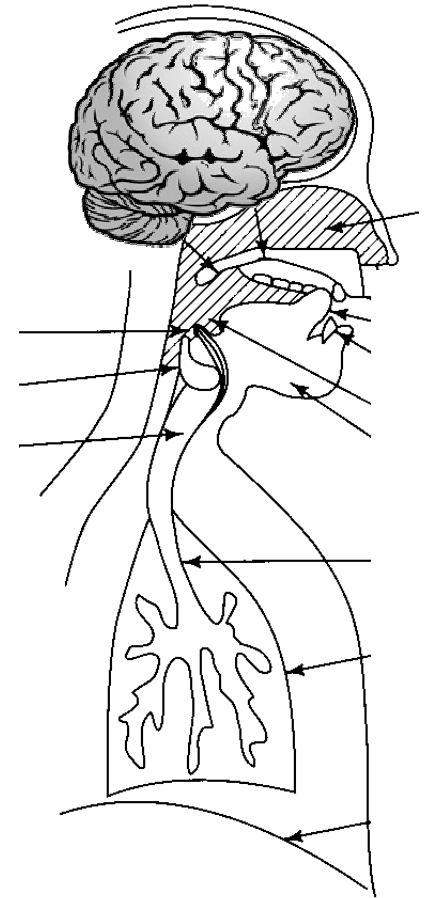
- The noisy channel approach:
  - Build a generative model of production (encoding)

$$P(A, s) = P(s) P(A | s)$$

- To decode, we use Bayes' rule to write

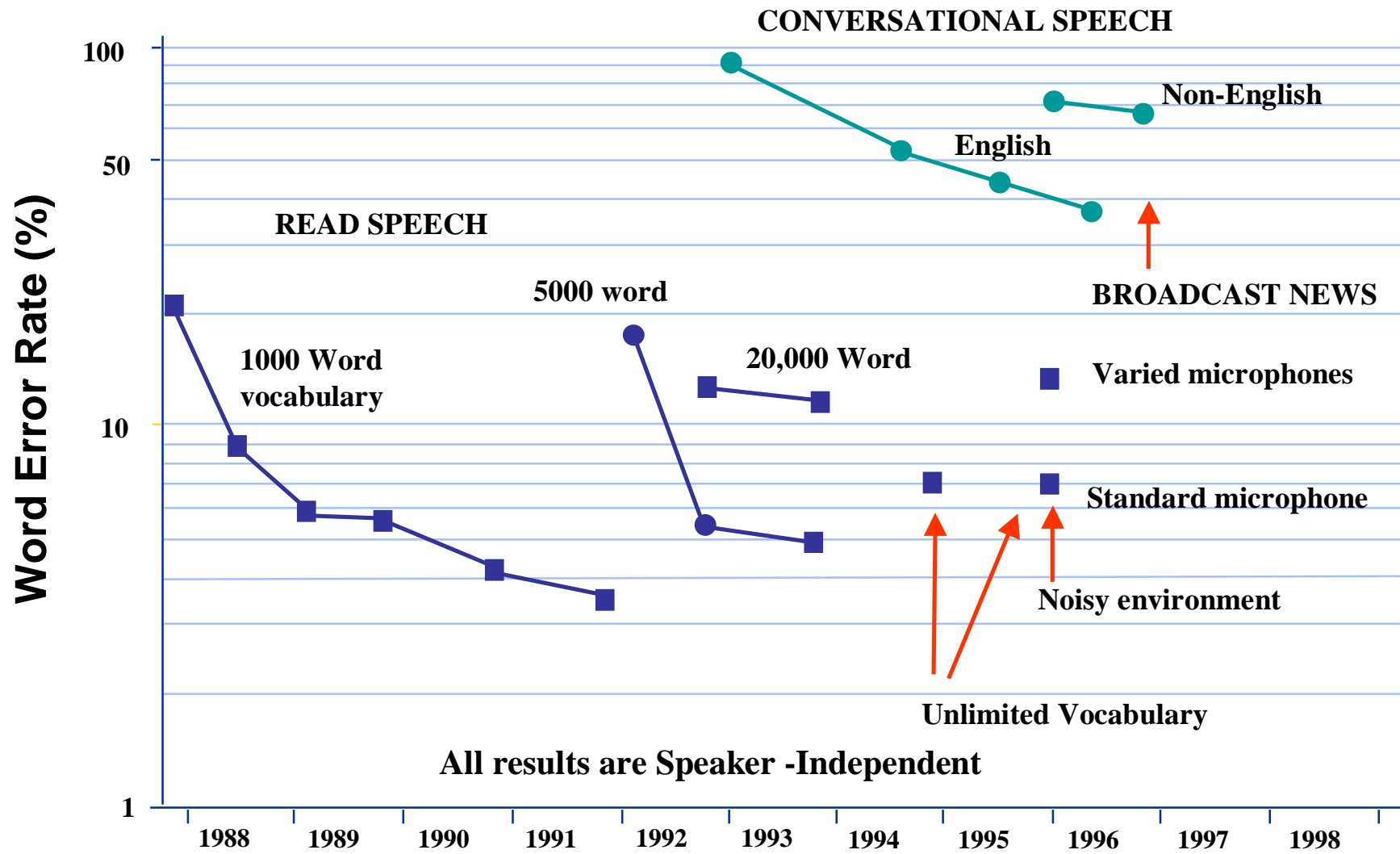
$$\begin{aligned} s^* &= \arg \max_s P(s | A) \\ &= \arg \max_s P(s) P(A | s) / P(A) \\ &= \arg \max_s P(s) P(A | s) \end{aligned}$$

- Now, we have to find a sentence maximizing this product

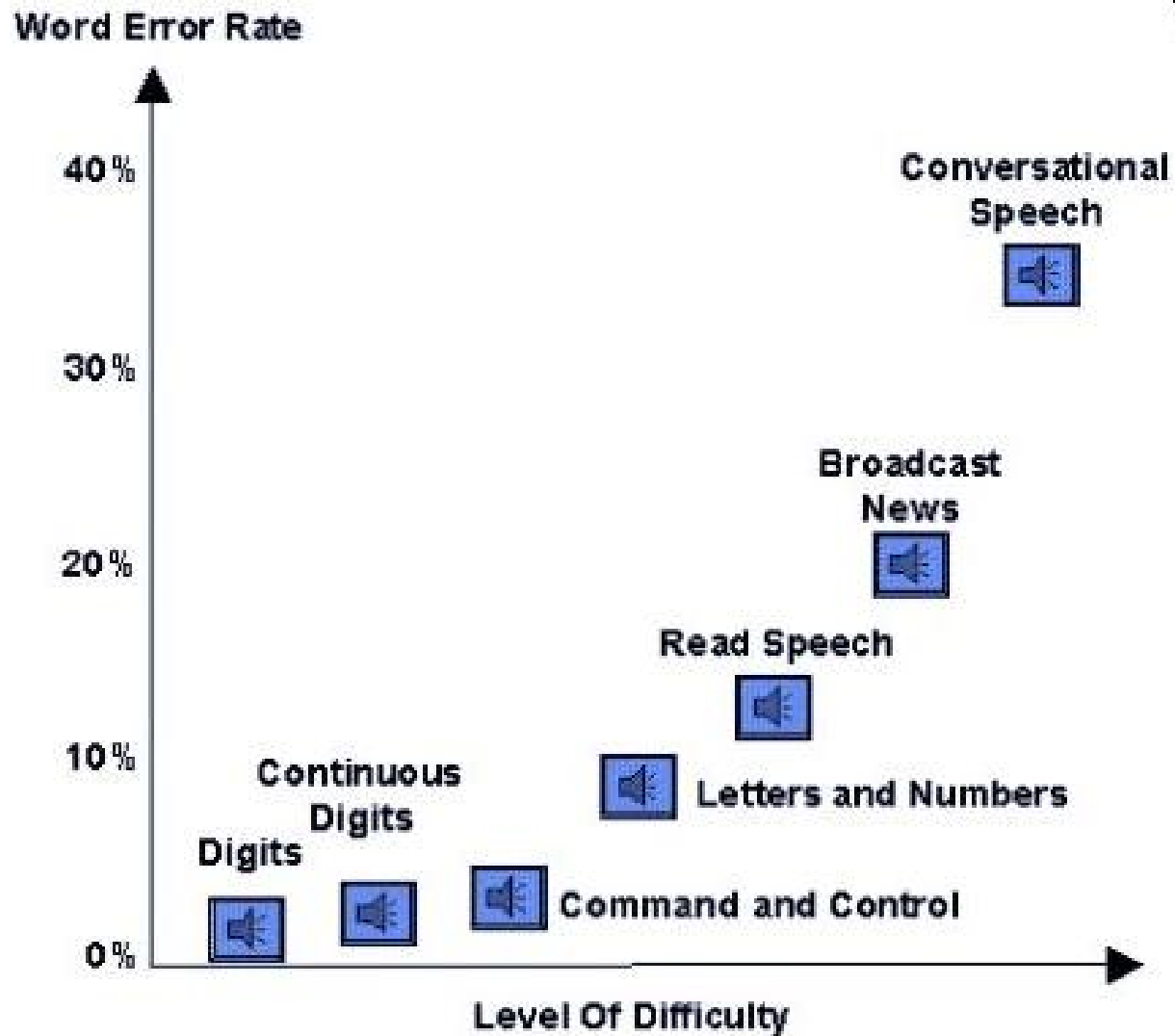


# Continual Progress in Speech Recognition

## Increasingly Difficult Tasks, Steadily Declining Error Rates



# Level of Difficulty

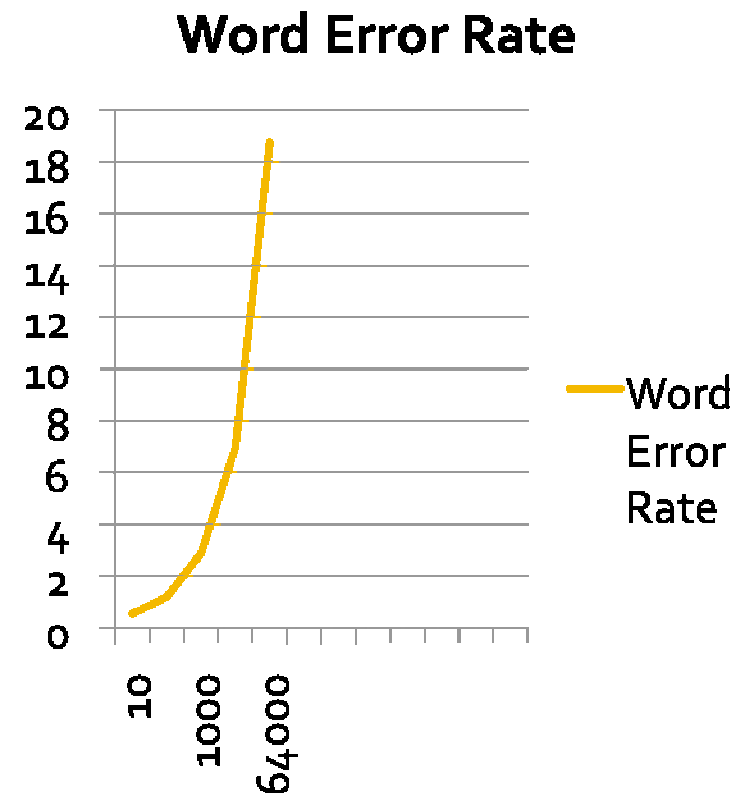


## Where speech recognition works?

- Limited Vocab Multi-Speaker
- Extensive Vocab Single Speaker

Vocabulary	Sphinx4 WER
Digits 0-9	.549%
100 Word	1.192%
1,000 Word	2.88%
5,000 Word	6.97%
64,000 Word	18.756%

*\*If you have noisy audio input multiply expected error rate  $\times 2$*





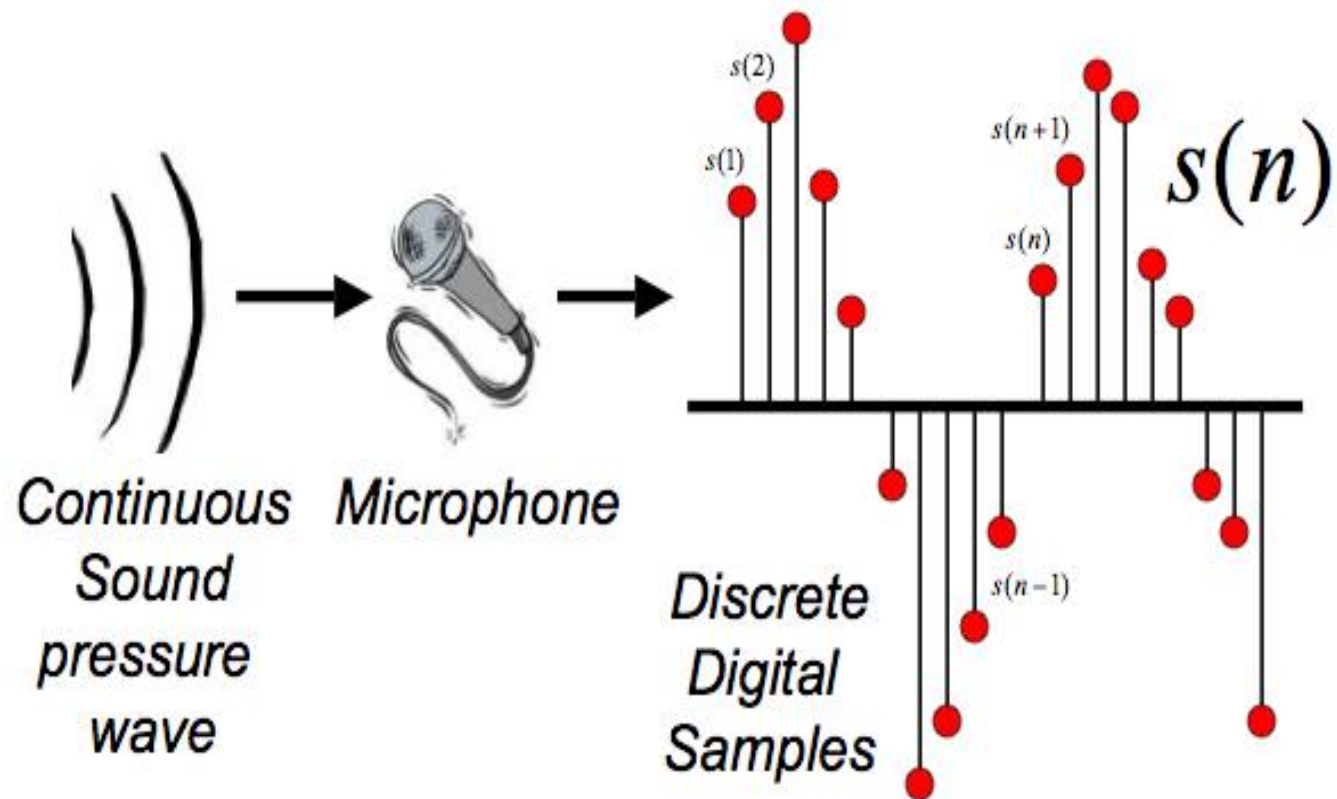
# Speech recognition system

- **Acoustic modeling**
- Lexicon
- Lenguaje Model
- Speech recognition



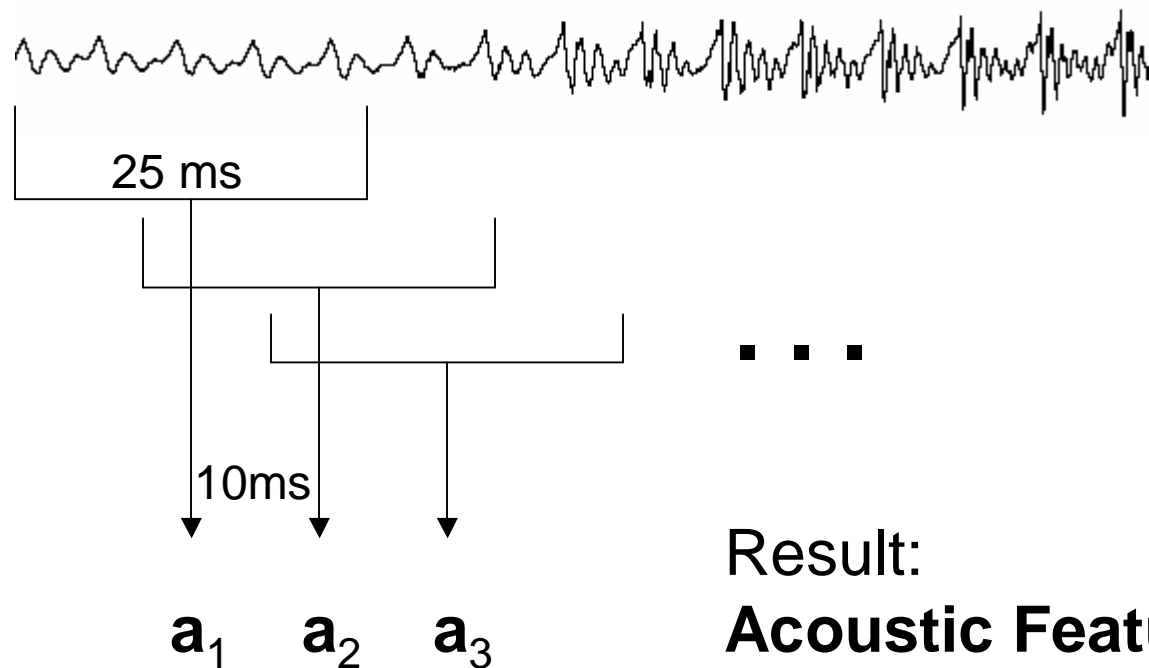
- Constructs the HMMs of phones
- Produces observation likelihoods from audio recordings vs their transcriptions
- Resulting in statistical representations of the sounds that make up each word (through a process called 'training')
- Sampling rate is critical!
- WSJ vs. WSJ\_8k
- TIDIGITS, RM1, AN4, HUB4

# Digitizing Speech



# Acoustic Sampling

- 10 ms frame (ms = millisecond = 1/1000 second)
- ~25 ms window around frame to smooth signal processing



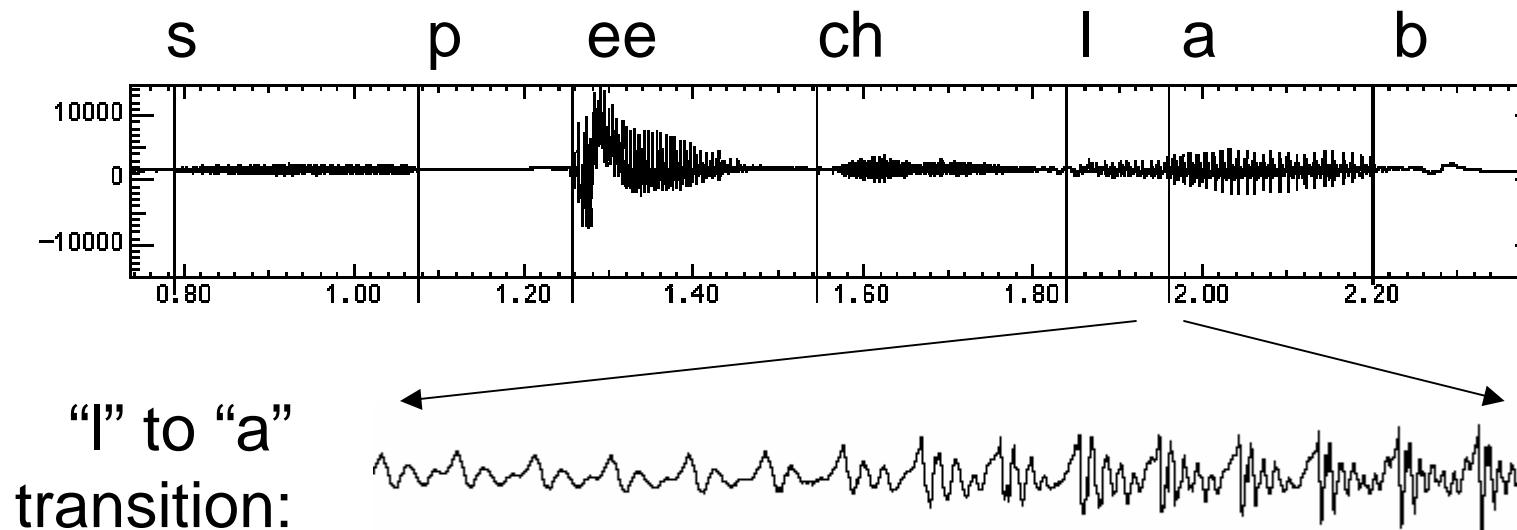
Result:  
**Acoustic Feature Vectors**

## Coding scheme (typical)

- 10 millisecond step size; 25 millisecond window
- ~39 coefficients each step:
  - mel-scale cepstra derived from frequency representation
  - $\Delta$  and  $\Delta \Delta$  coefficients
  - power

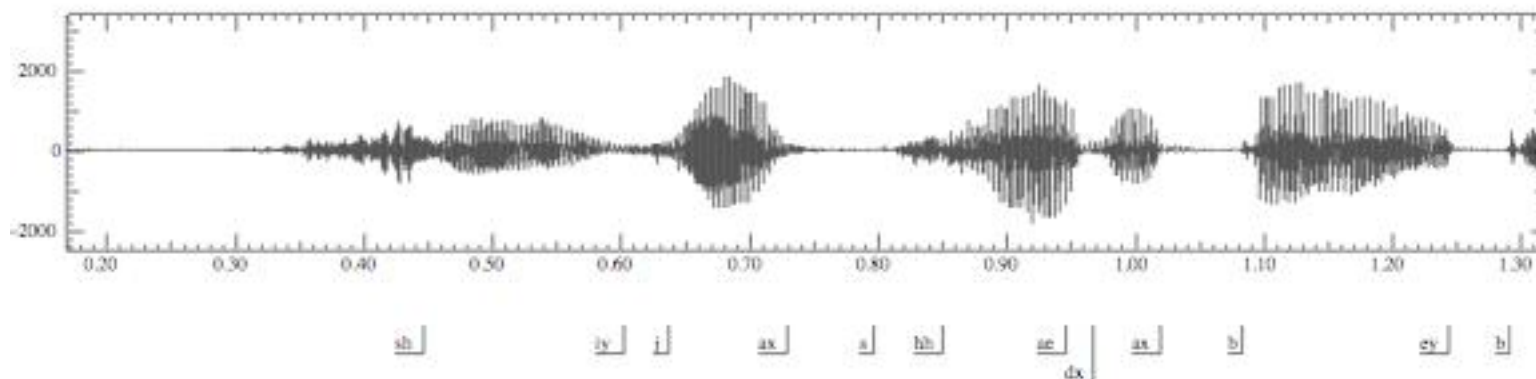
# Speech in an Hour

- Speech input is an acoustic wave form



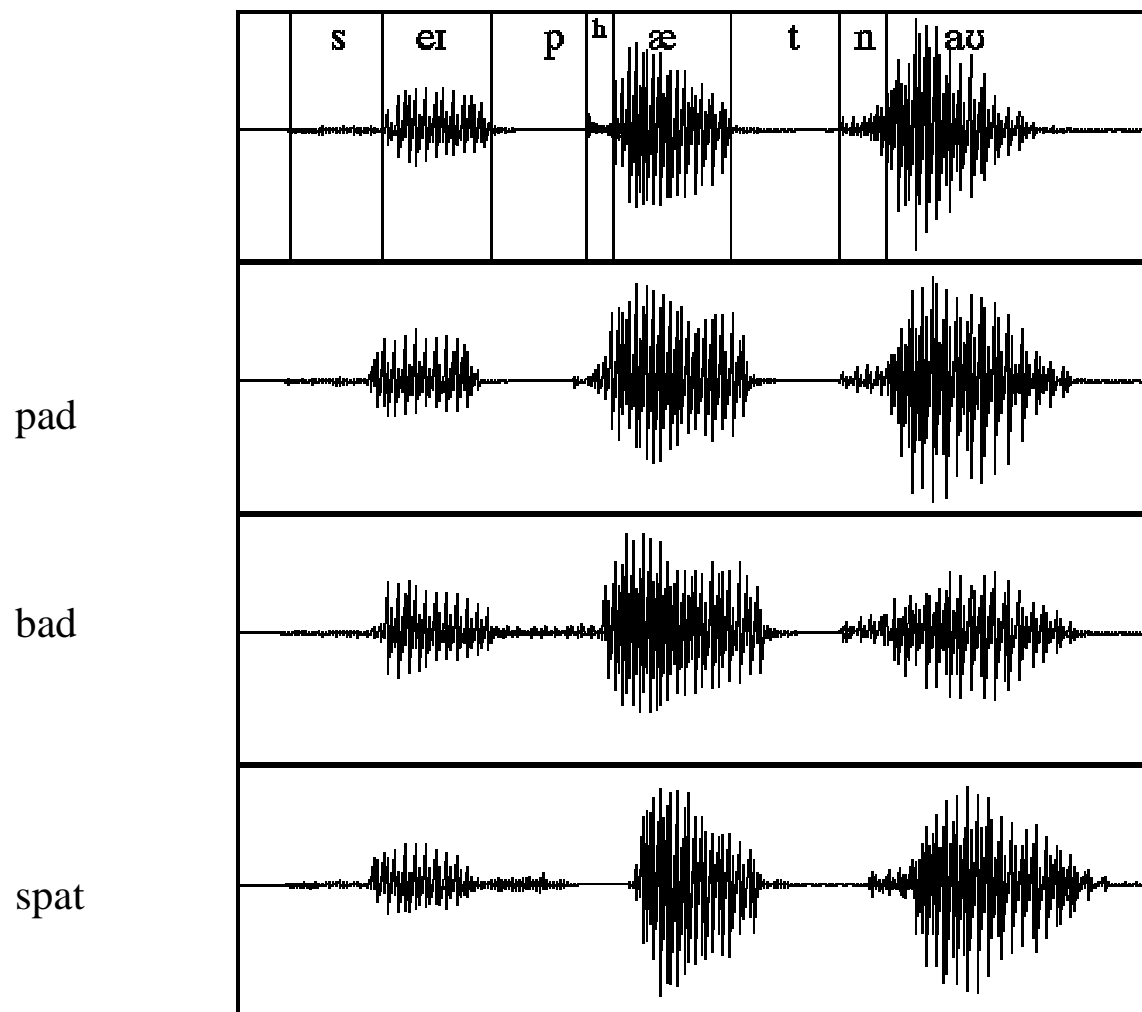
Graphs from Simon Arnfield's web tutorial on speech, Sheffield:  
<http://www.psyc.leeds.ac.uk/research/cogn/speech/tutorial/>

She just had a baby



- What can we learn from a wavefile?
  - Vowels are voiced (vocal cord vibrates), long, loud
  - Length in time = length in space in waveform picture
  - Voicing: regular peaks in amplitude
  - When stops closed: no peaks: silence.
  - Peaks = voicing: .46 to .58 (vowel [i], from second .65 to .74 (vowel [u]) and so on
  - Silence of stop closure (1.06 to 1.08 for first [b], or 1.26 to 1.28 for second [b])
  - Fricatives (f tongue hits upper teeth) like [S] intense irregular pattern; see .33 to .46

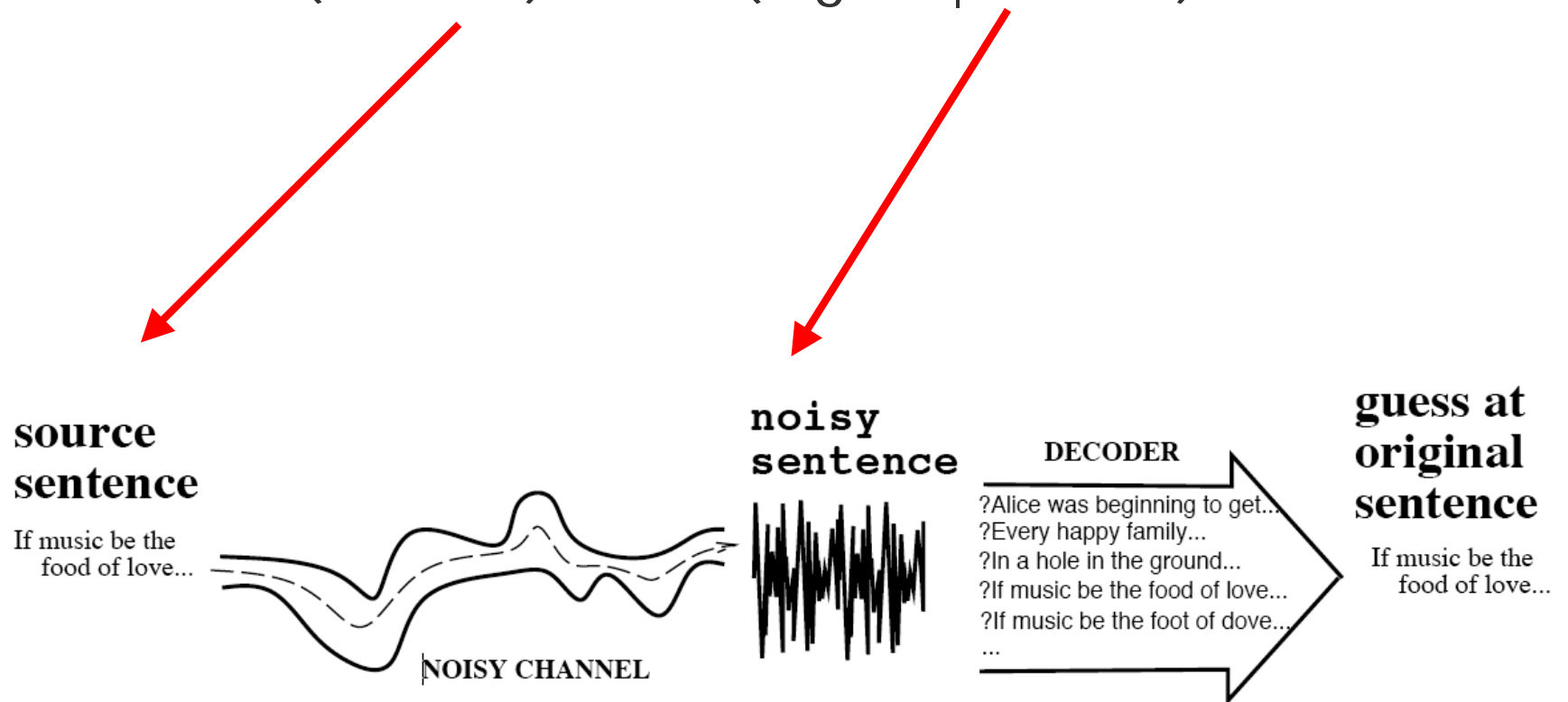
# Examples from Ladefoged



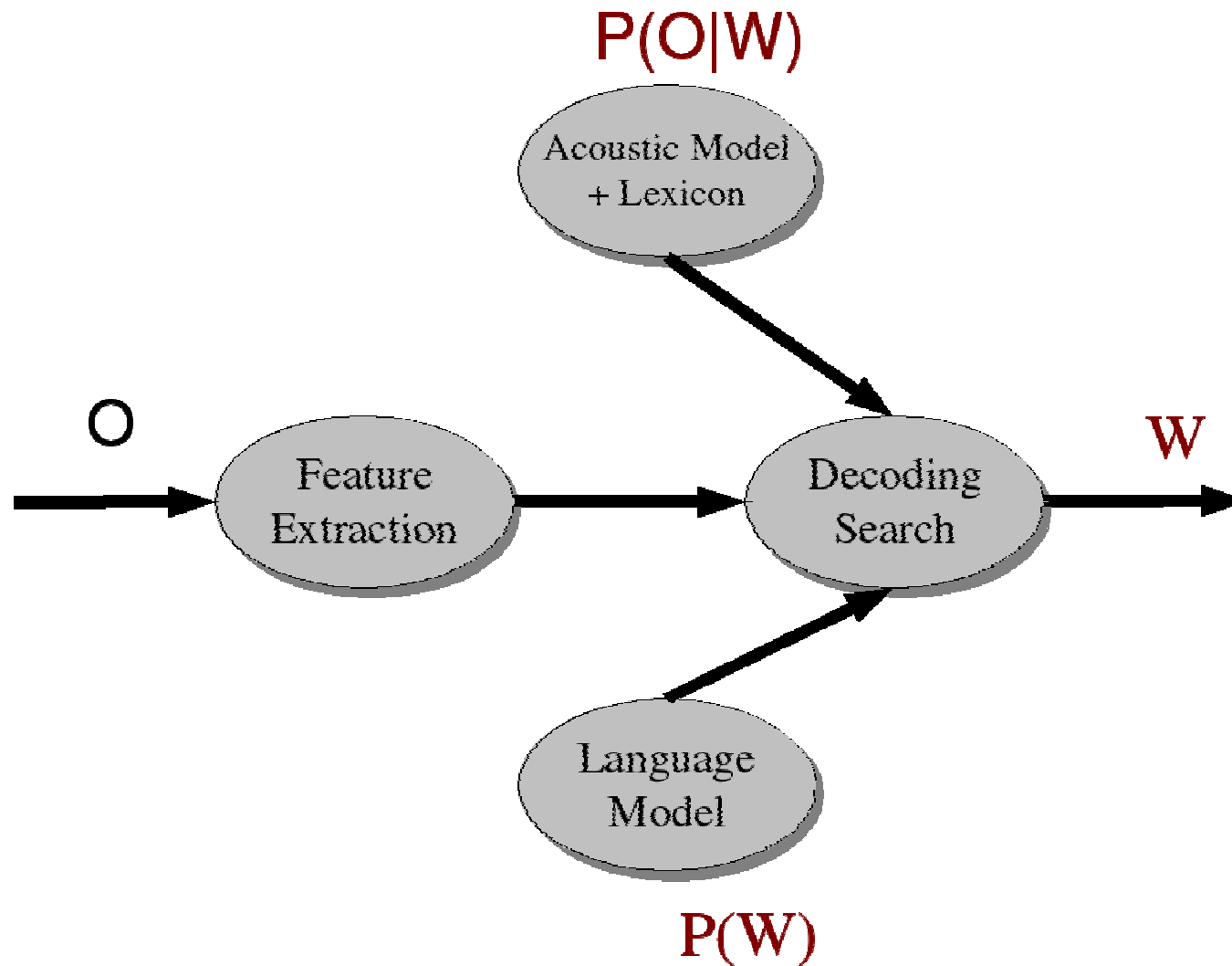


# The noisy channel model

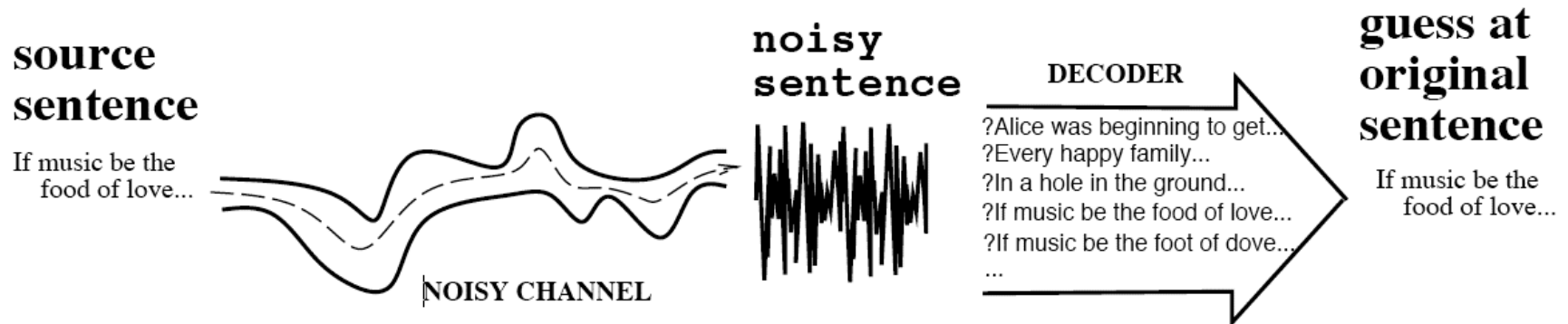
- Ignoring the denominator leaves us with two factors:  $P(\text{Source})$  and  $P(\text{Signal} | \text{Source})$



# Speech Architecture meets Noisy Channel



# The Noisy Channel Model



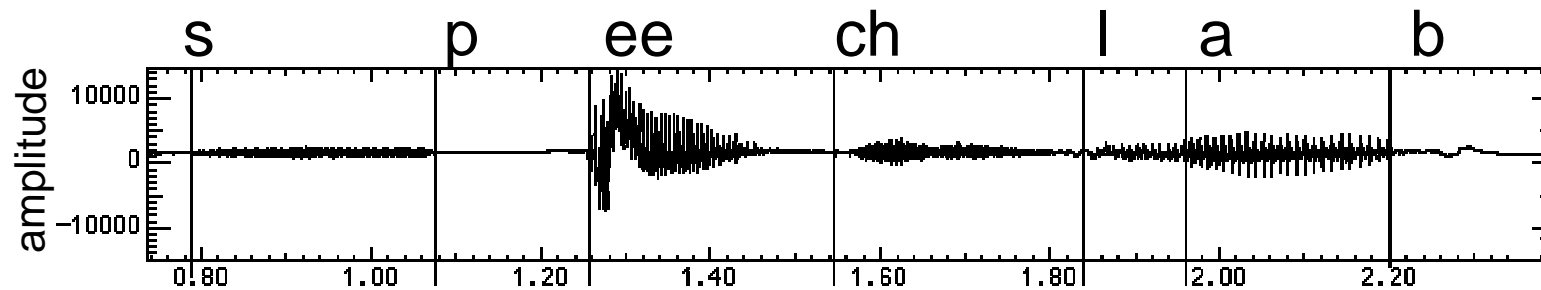
- Search through space of all possible sentences
- Pick the one that is most probable given the waveform

## The Noisy Channel Model

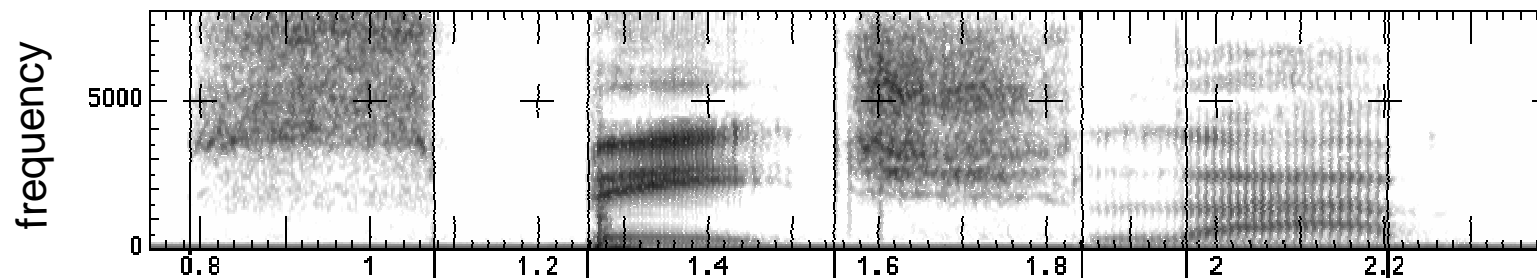
- Use the acoustic model to give a set of likely phone sequences
- Use the lexical and language models to judge which of these are likely to result in probable word sequences
- The trick is having sophisticated algorithms to juggle the statistics
- A bit like the rule-based approach except that it is all learned automatically from data

# Spectral Analysis

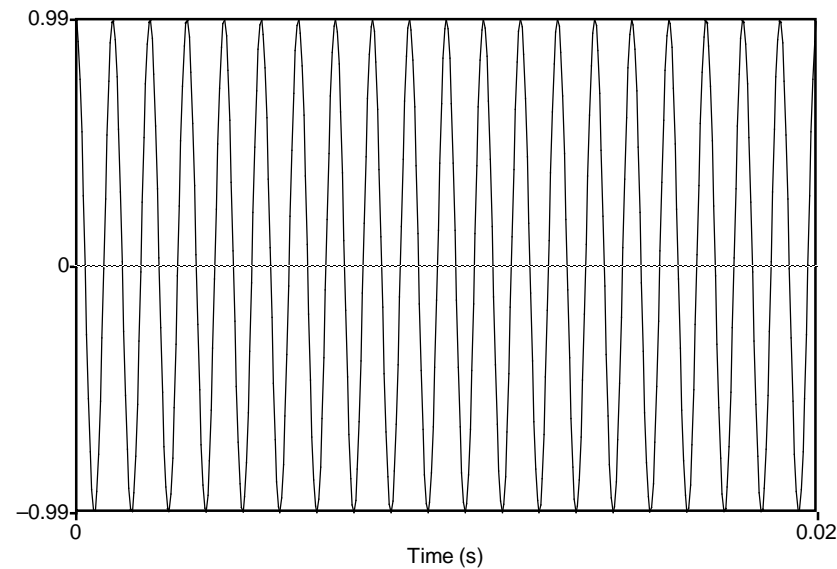
- Frequency gives pitch; amplitude gives volume
  - sampling at ~8 kHz phone, ~16 kHz mic (kHz=1000 cycles/sec)



- Fourier transform of wave displayed as a spectrogram
  - darkness indicates energy at each frequency

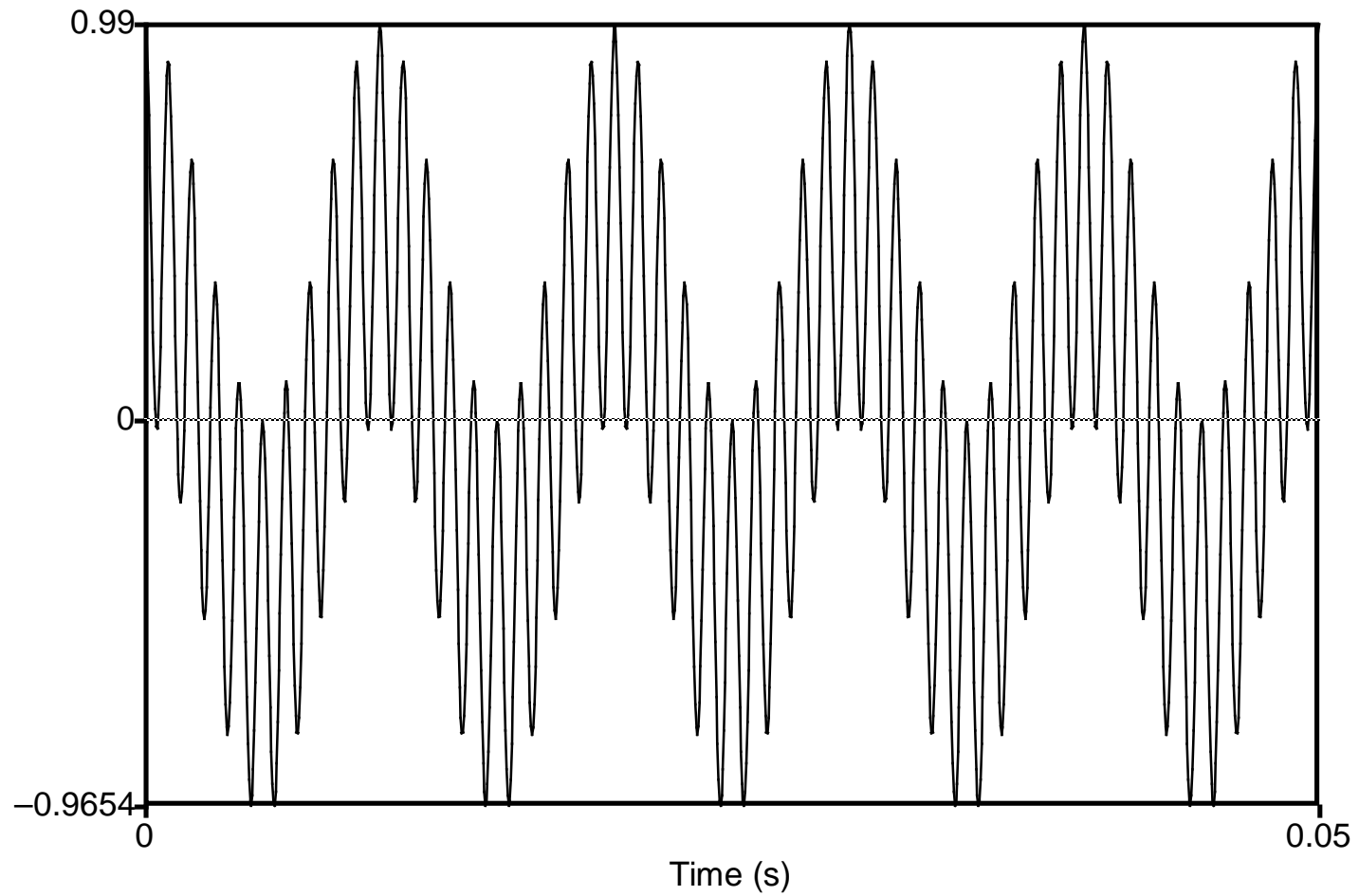


# Simple Periodic Sound Waves



- Y axis: Amplitude = amount of air pressure at that point in time
  - Zero is normal air pressure, negative is rarefaction
- X axis: time. Frequency = number of cycles per second.
  - Frequency =  $1/\text{Period}$
  - 20 cycles in .02 seconds = 1000 cycles/second = 1000 Hz

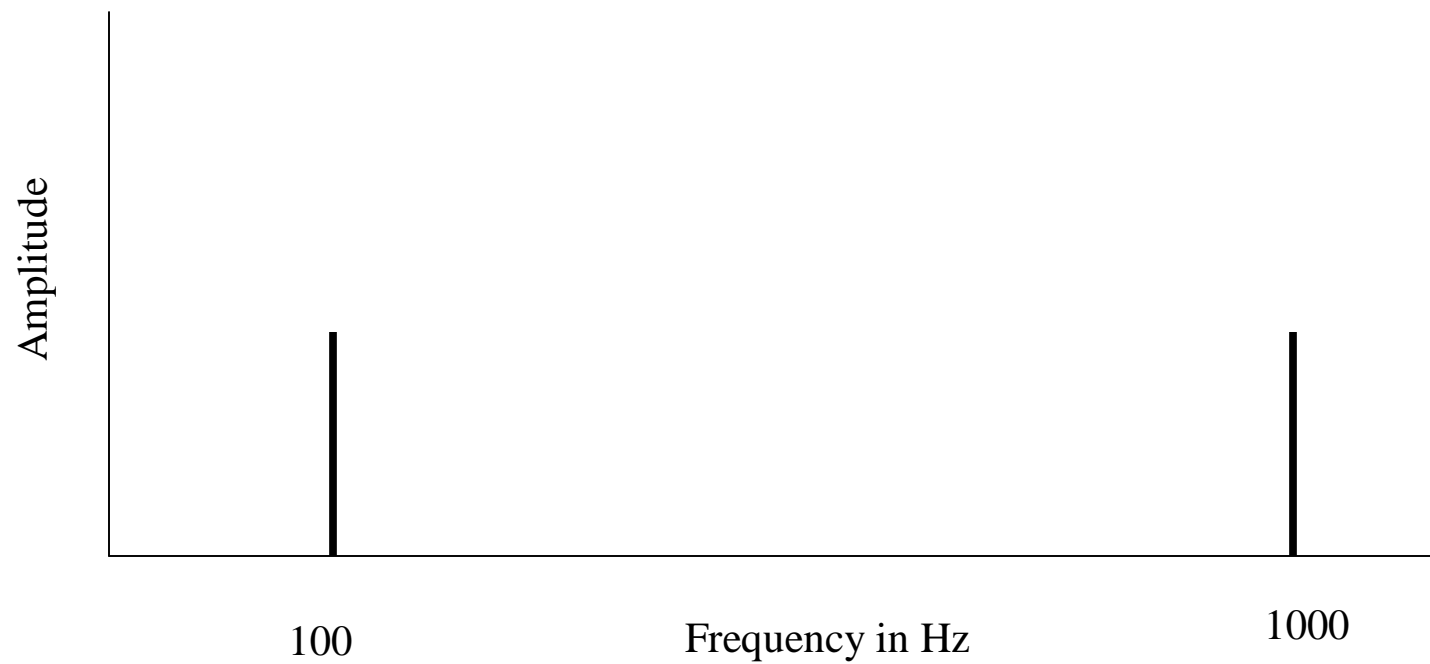
## Adding 100 Hz + 1000 Hz Waves



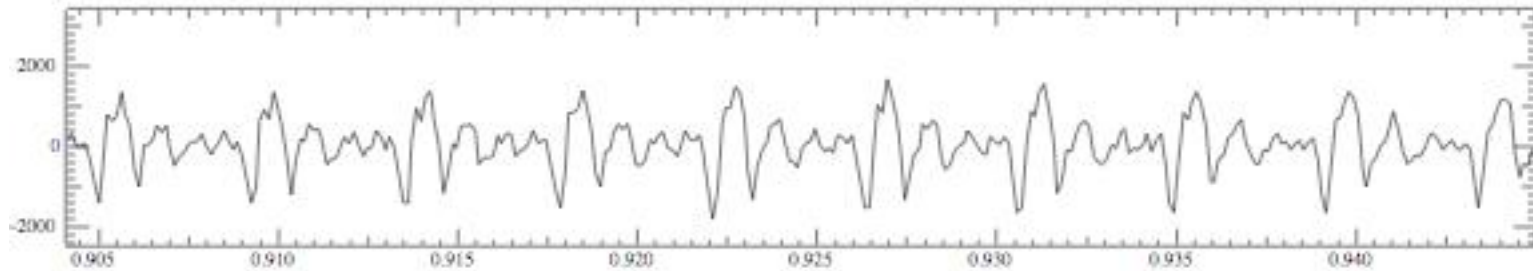


# Spectrum

Frequency components (100 and 1000 Hz) on x-axis



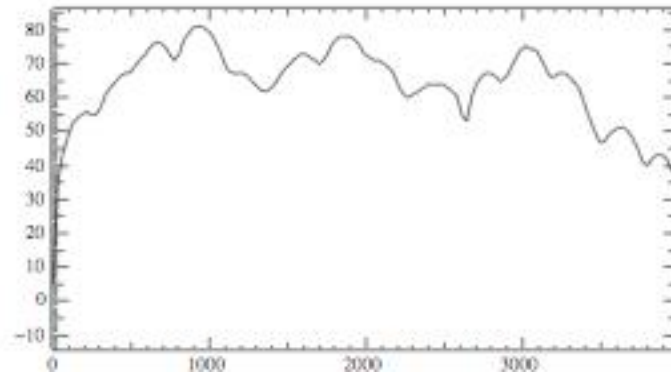
## Part of [ae] from “had”



- Note complex wave repeating nine times in figure
- Plus smaller waves which repeats 4 times for every large pattern
- Large wave has frequency of 250 Hz (9 times in .036 seconds)
- Small wave roughly 4 times this, or roughly 1000 Hz
- Two little tiny waves on top of peak of 1000 Hz waves

## Back to Spectra

- Spectrum represents these freq components
- Computed by Fourier transform, algorithm which separates out each frequency component of wave.

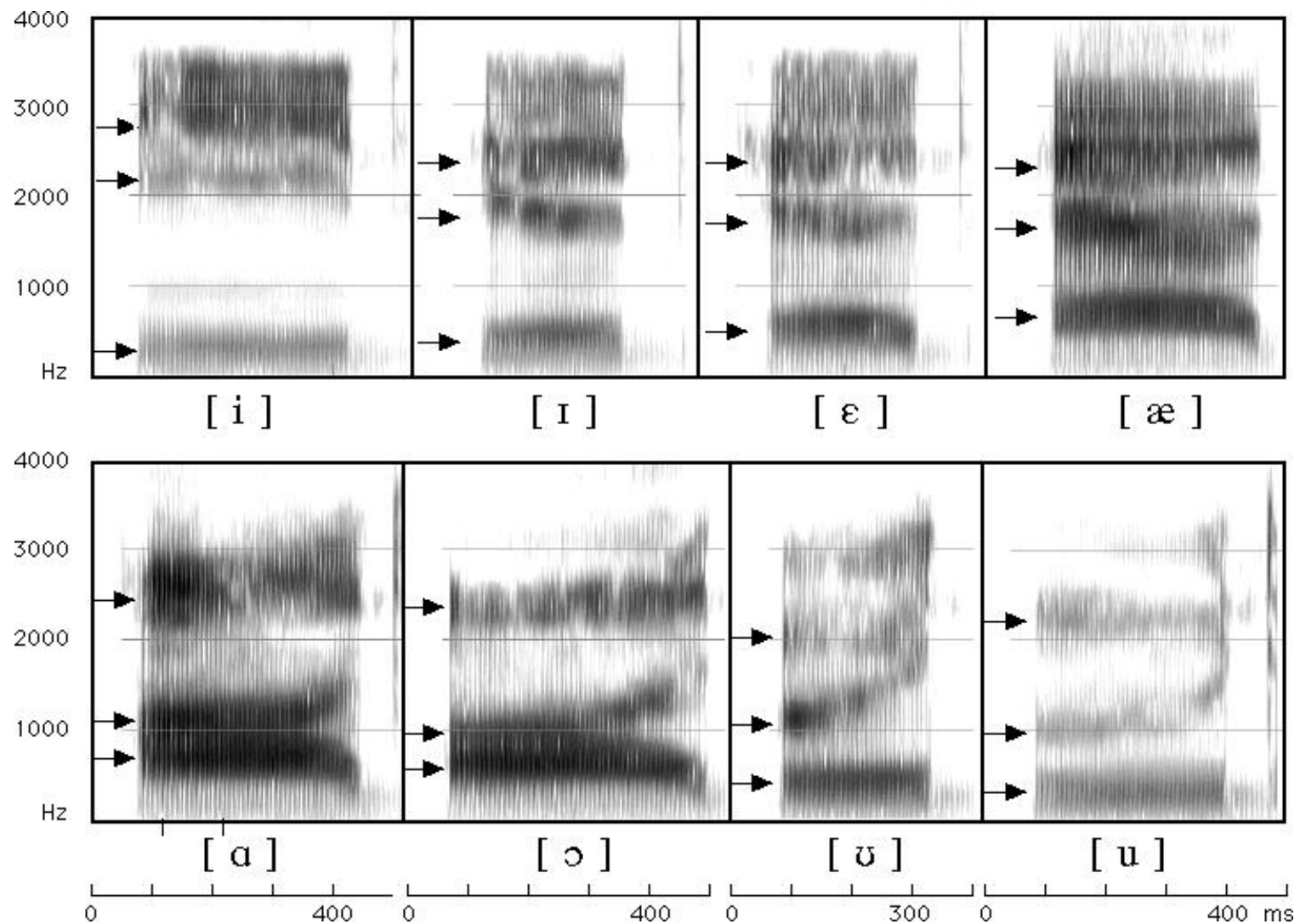


- x-axis shows frequency, y-axis shows magnitude (in decibels, a log measure of amplitude)
- Peaks at 930 Hz, 1860 Hz, and 3020 Hz.

# HMMs for Continuous Observations?

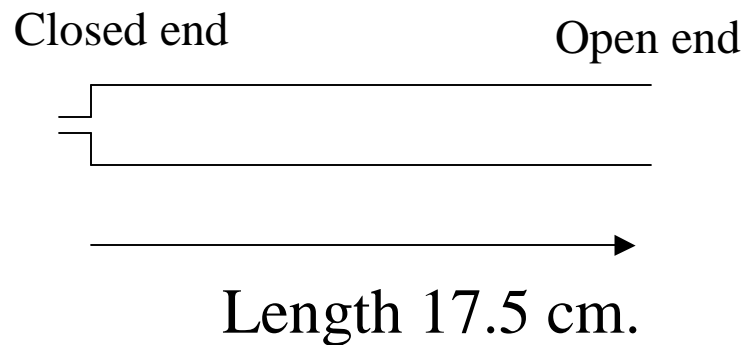
- Before: discrete, finite set of observations
- Now: spectral feature vectors are real-valued!
- Solution 1: discretization
- Solution 2: continuous emissions models
  - Gaussians
  - Multivariate Gaussians
  - Mixtures of Multivariate Gaussians
- A state is progressively:
  - Context independent subphone (~3 per phone)
  - Context dependent phone (=triphones)
  - State-tying of CD phone

# Vowel Formants



# Resonances of the vocal tract

- The human vocal tract as an open tube



- Air in a tube of a given length will tend to vibrate at resonance frequency of tube.
- Constraint: Pressure differential should be maximal at (closed) glottal end and minimal at (open) lip end.

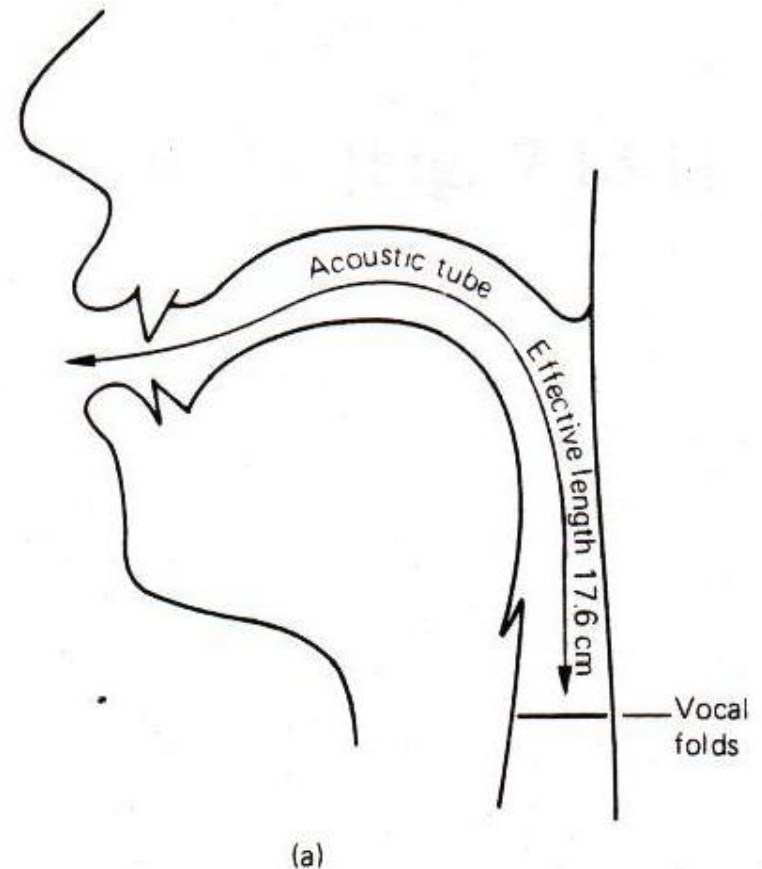
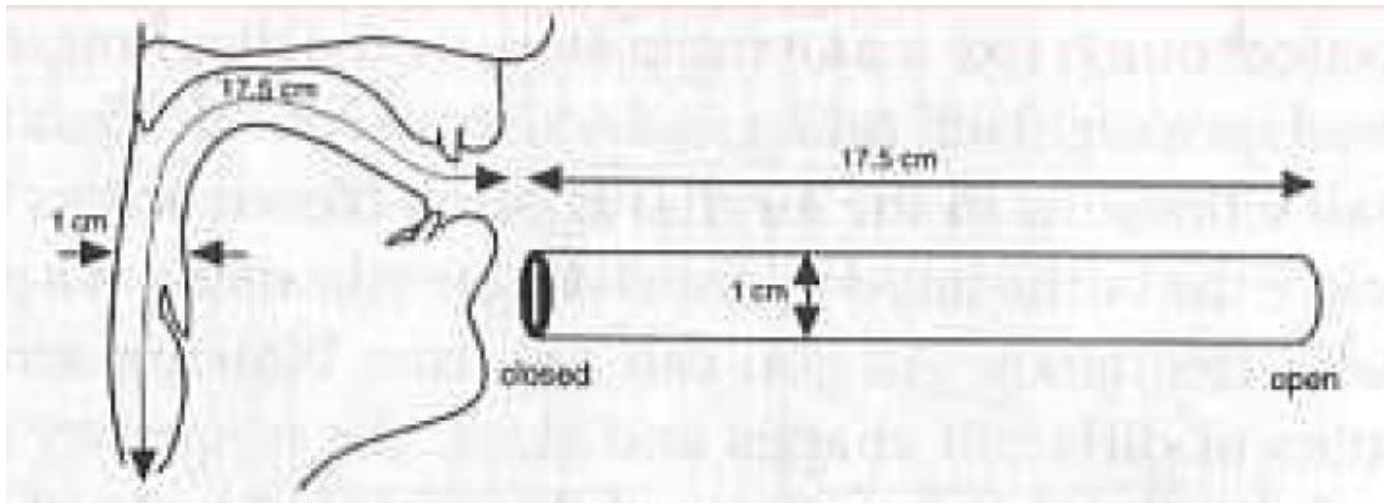


Figure from W. Barry Speech Science slides

# Deriving Schwa

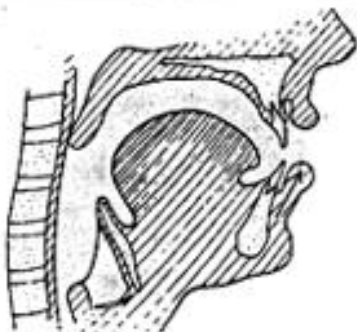
- Reminder of basic facts about sound waves
  - $f = c/\lambda$
  - $c$  = speed of sound (approx 35,000 cm/sec)
  - A sound with  $\lambda=10$  meters:  $f = 35$  Hz (35,000/1000)
  - A sound with  $\lambda=2$  centimeters:  $f = 17,500$  Hz (35,000/2)



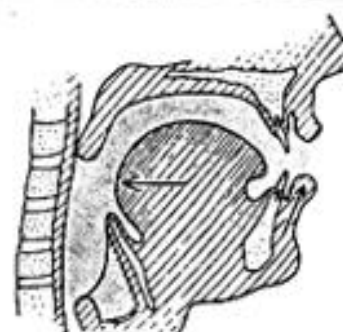
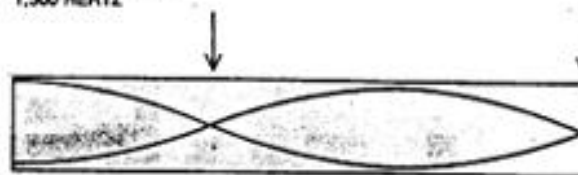


# Resonances

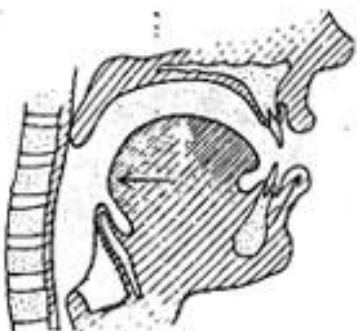
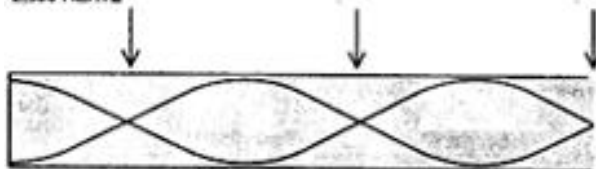
FIRST FORMANT  
1/4 WAVELENGTH  
500 HERTZ



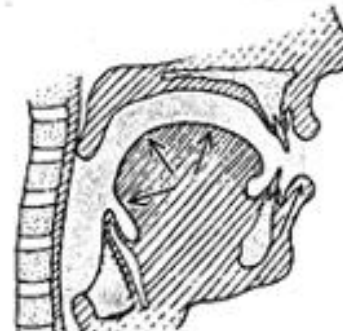
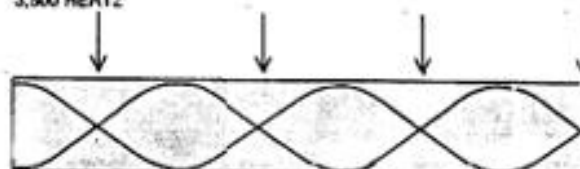
SECOND FORMANT  
3/4 WAVELENGTH  
1,500 HERTZ



THIRD FORMANT  
5/4 WAVELENGTH  
2,500 HERTZ

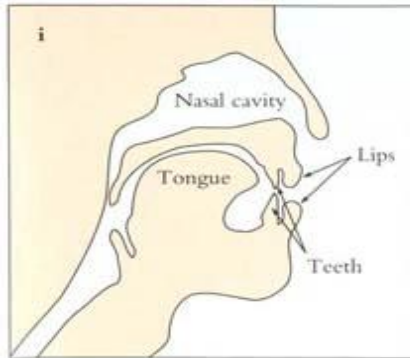


FOURTH FORMANT  
7/4 WAVELENGTH  
3,500 HERTZ

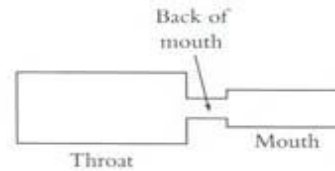


# Resonances

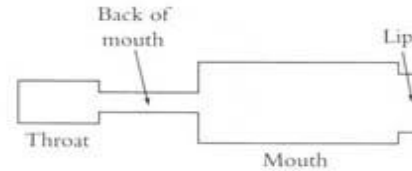
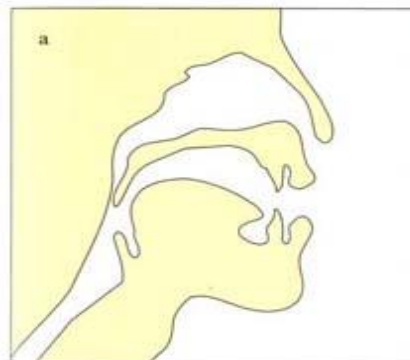
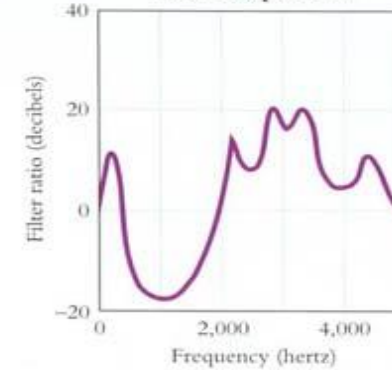
Cross section of vocal tract



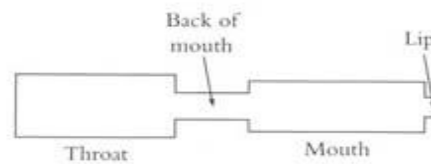
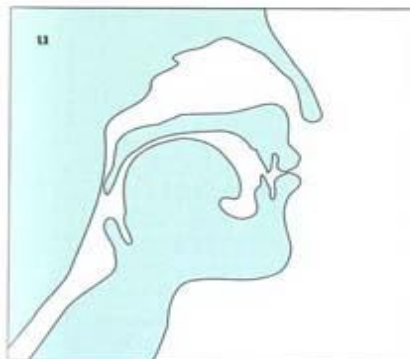
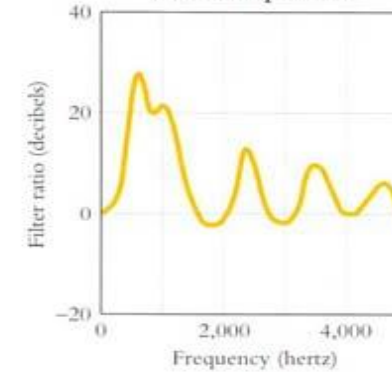
Model of vocal tract



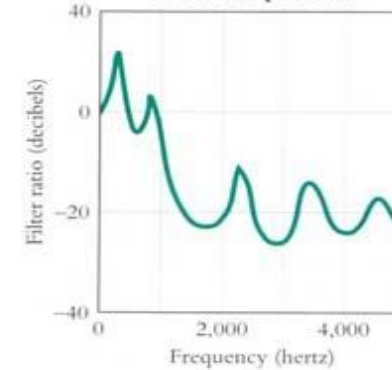
Acoustic spectrum



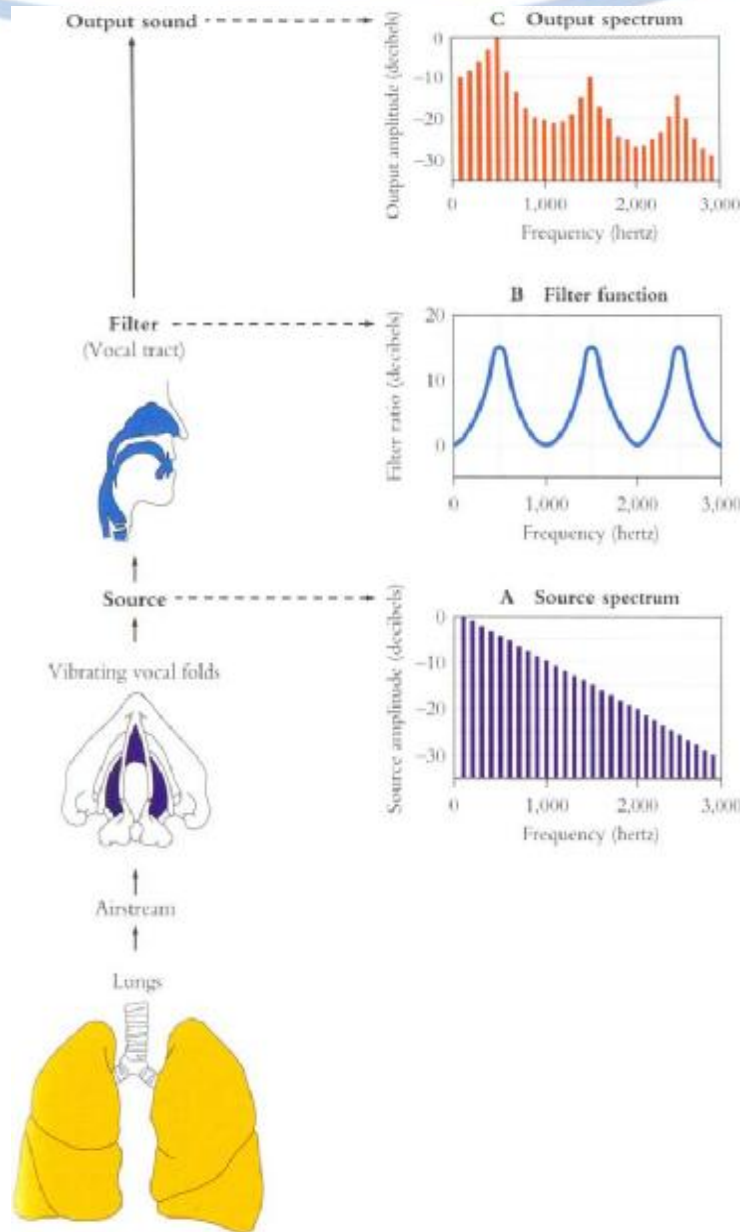
Acoustic spectrum



Acoustic spectrum

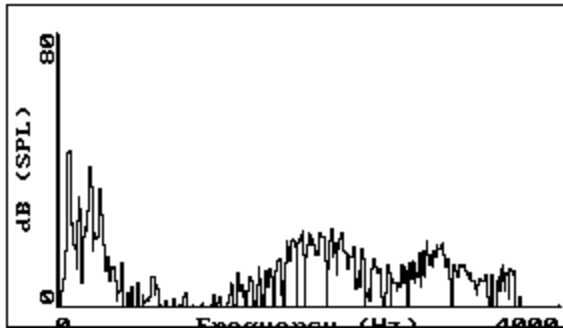


# Articulation Process

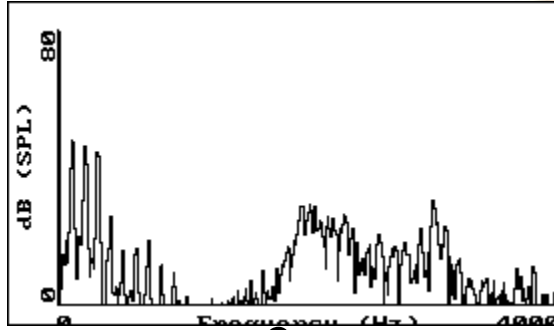


- Articulatory facts:
  - Vocal cord vibrations create harmonics
  - The mouth is a selective amplifier
  - Depending on shape of mouth, some harmonics are amplified more than others

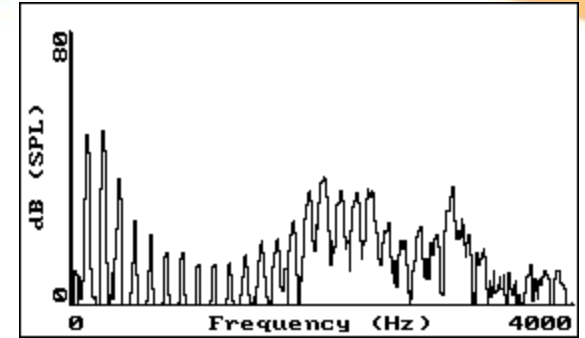
# Vowel [i] sung at successively higher pitch



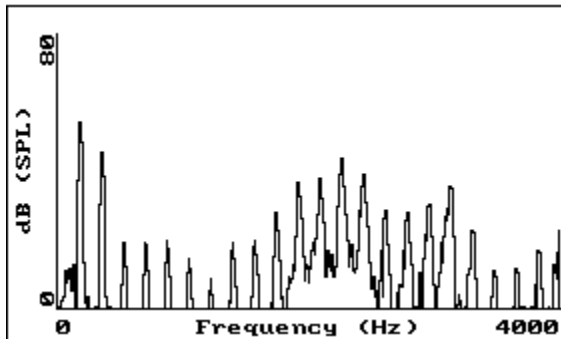
1



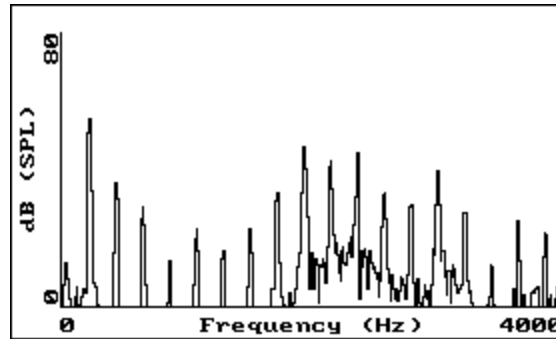
2



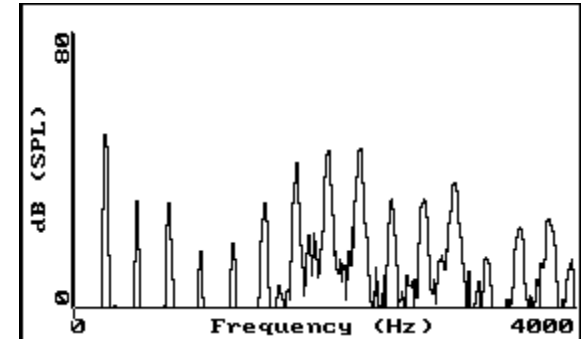
3



4

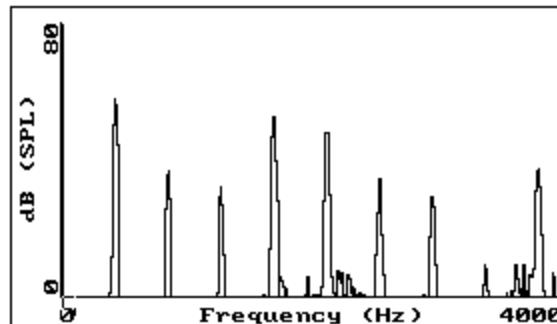


5



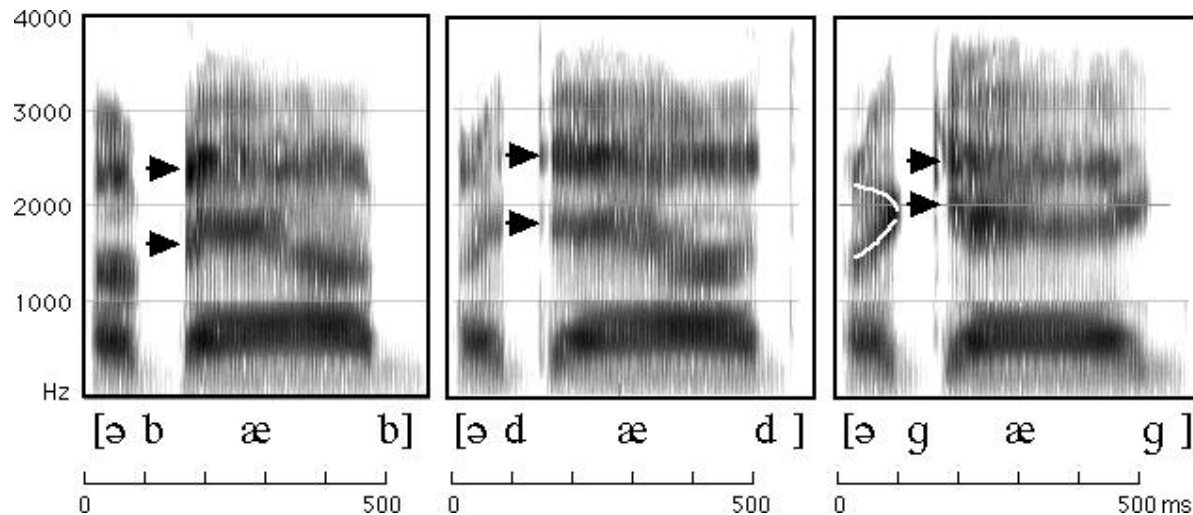
6

7



Figures from Ratre  
Wayland slides from his  
website

## How to read spectrograms



- bab: closure of lips lowers all formants: so rapid increase in all formants at beginning of "bab"
- dad: first formant increases, but F2 and F3 slight fall
- gag: F2 and F3 come together: this is a characteristic of velars. Formant transitions take longer in velars than in alveolars or labials

From Ladefoged "A Course in Phonetics"

## Computing the 3 Formants of Schwa

- Let the length of the tube be L
  - $F_1 = c/\lambda_1 = c/(4L) = 35,000/4*17.5 = 500\text{Hz}$
  - $F_2 = c/\lambda_2 = c/(4/3L) = 3c/4L = 3*35,000/4*17.5 = 1500\text{Hz}$
  - $F_3 = c/\lambda_3 = c/(4/5L) = 5c/4L = 5*35,000/4*17.5 = 2500\text{Hz}$
- So we expect a neutral vowel to have 3 resonances at 500, 1500, and 2500 Hz
- These vowel resonances are called **formants**



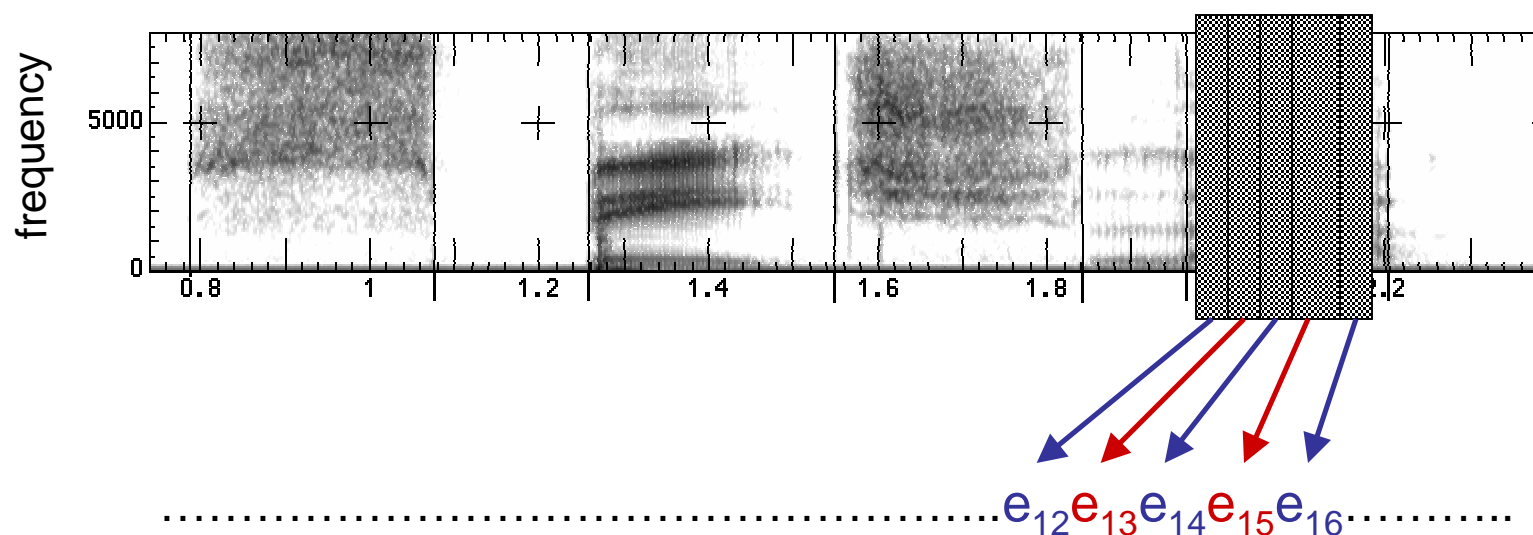
## Final Feature Vector

- 39 (real) features per 10 ms frame:
  - 12 MFCC features
  - 12 Delta MFCC features
  - 12 Delta-Delta MFCC features
  - 1 (log) frame energy
  - 1 Delta (log) frame energy
  - 1 Delta-Delta (log frame energy)
- So each frame is represented by a 39D vector
- (You don't have to know these details!)



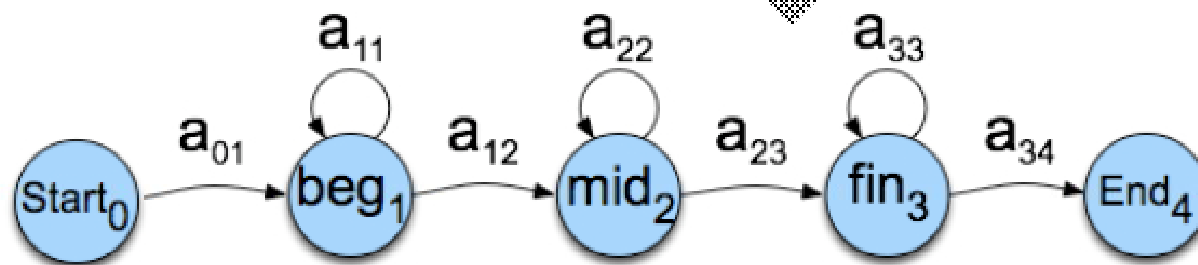
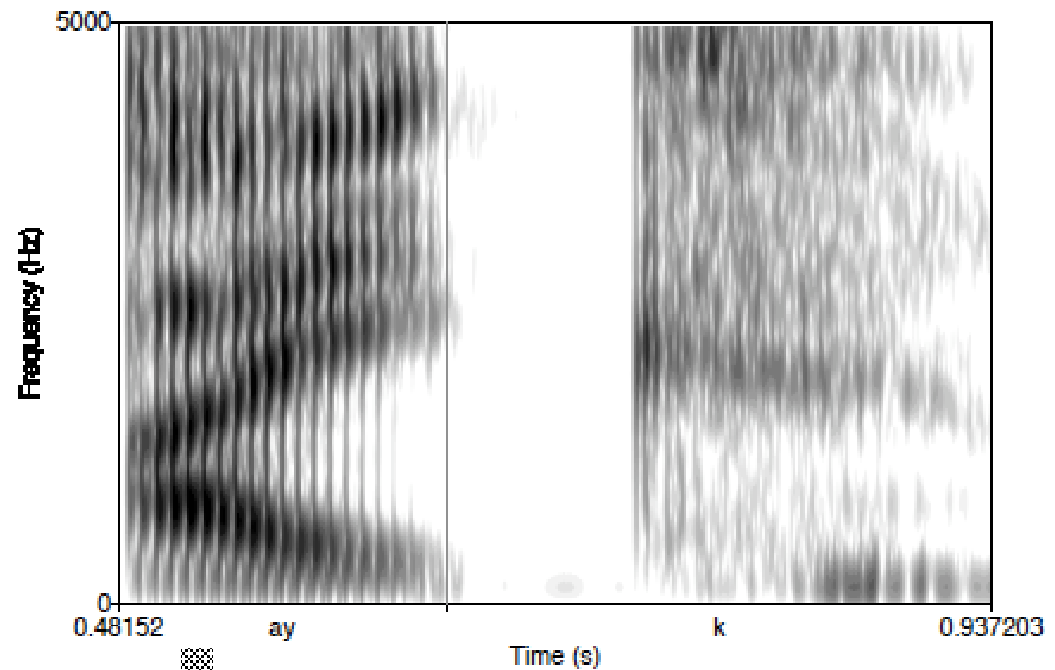
# Acoustic Feature Sequence

- Time slices are translated into acoustic feature vectors (~39 real numbers per slice)



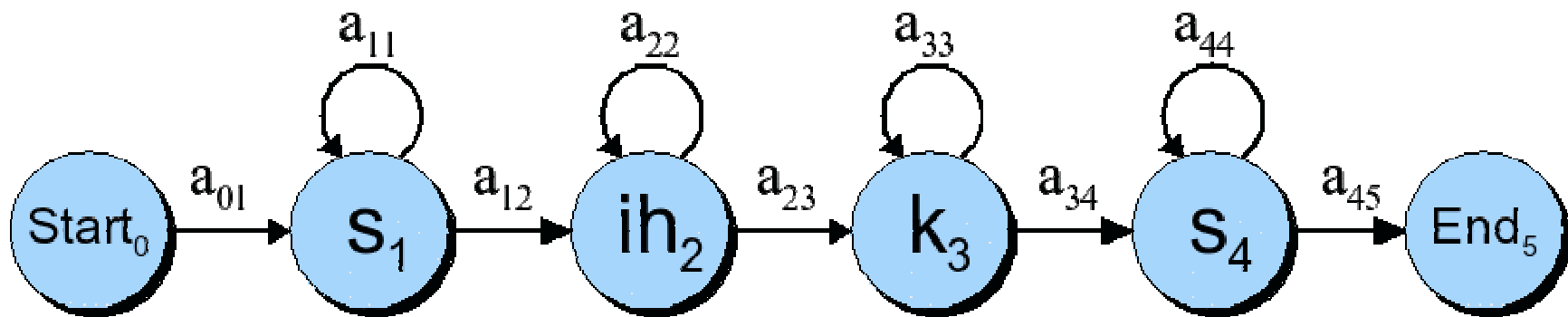
- These are the observations, now we need the hidden states  $X$

# Hidden Markov Models



# Hidden Markov Models (HMMs)

“Six”



# The Markov Assumption

- Only immediately preceding history matters

$$P(X_1, X_2, X_3, \mathbf{K}, X_n) = \prod_{i=1}^n P(X_i | X_1, X_2, X_3, \mathbf{K}, X_{i-1})$$

$$P(X_i | X_1, X_2, X_3, \mathbf{K}, X_n) = P(X_i | X_{i-1})$$

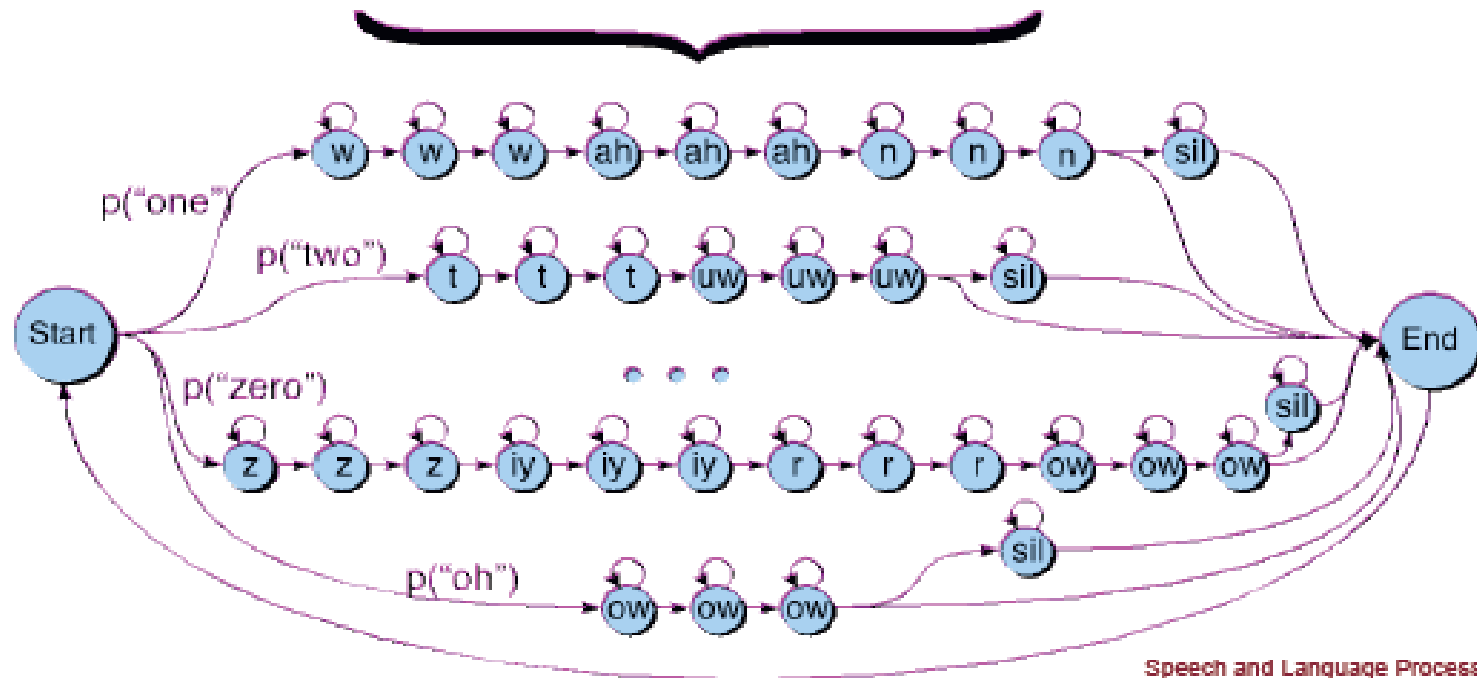
$$P(X_1, X_2, X_3, \mathbf{K}, X_n) = \prod_{i=1}^n P(X_i | X_{i-1})$$

# Search graph

## Lexicon

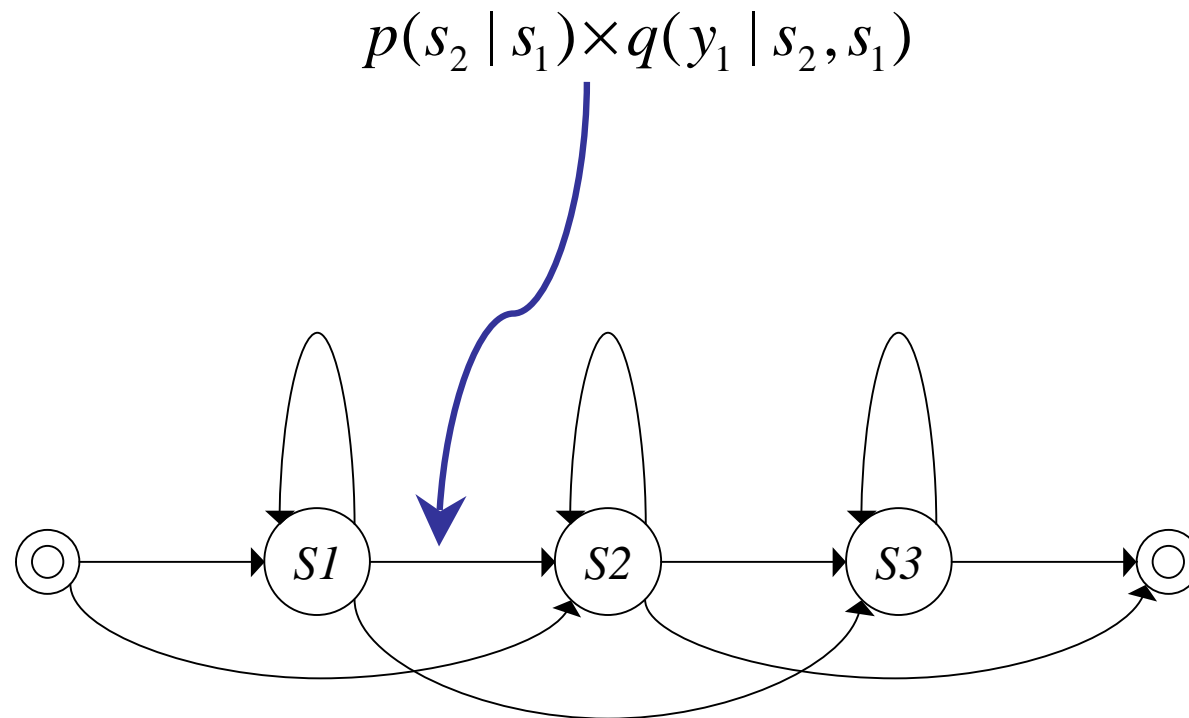
one	w ah n
two	t uw
three	th r iy
four	f ao r
five	f ay v
six	s ih k s
seven	s eh v ax n
eight	ey t
nine	n ay n
zero	z iy r ow n
oh	ow

## Phone HMM



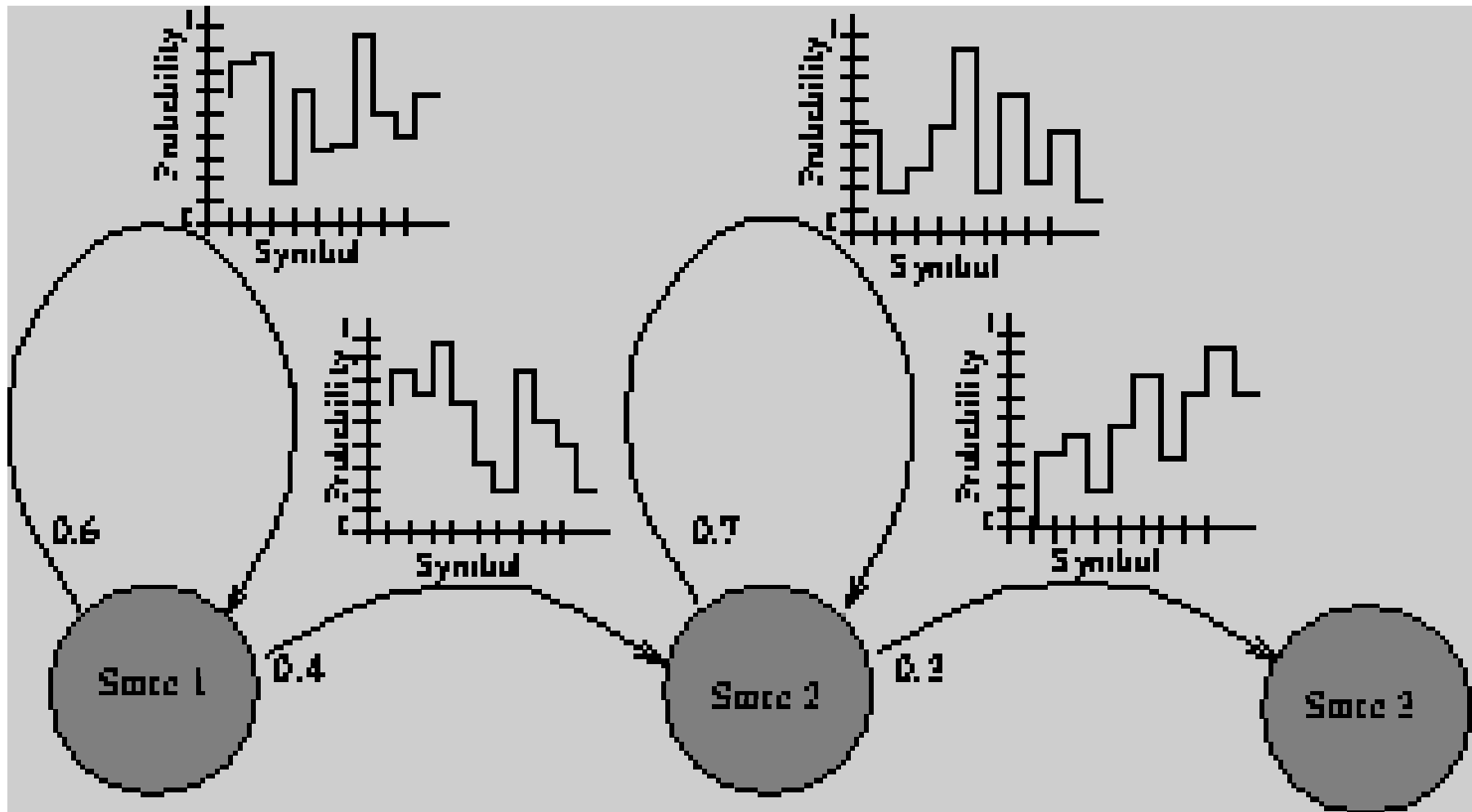
- Searches the graph for the “best fit”
  - $P(\text{sequence of feature vectors} \mid \text{word/phone})$
  - aka.  $P(O \mid W)$
- > “how likely is the input to have been generated by the word”

- In speech recognition the number of states is very large; we can simplify the problem by factoring the problem into two components

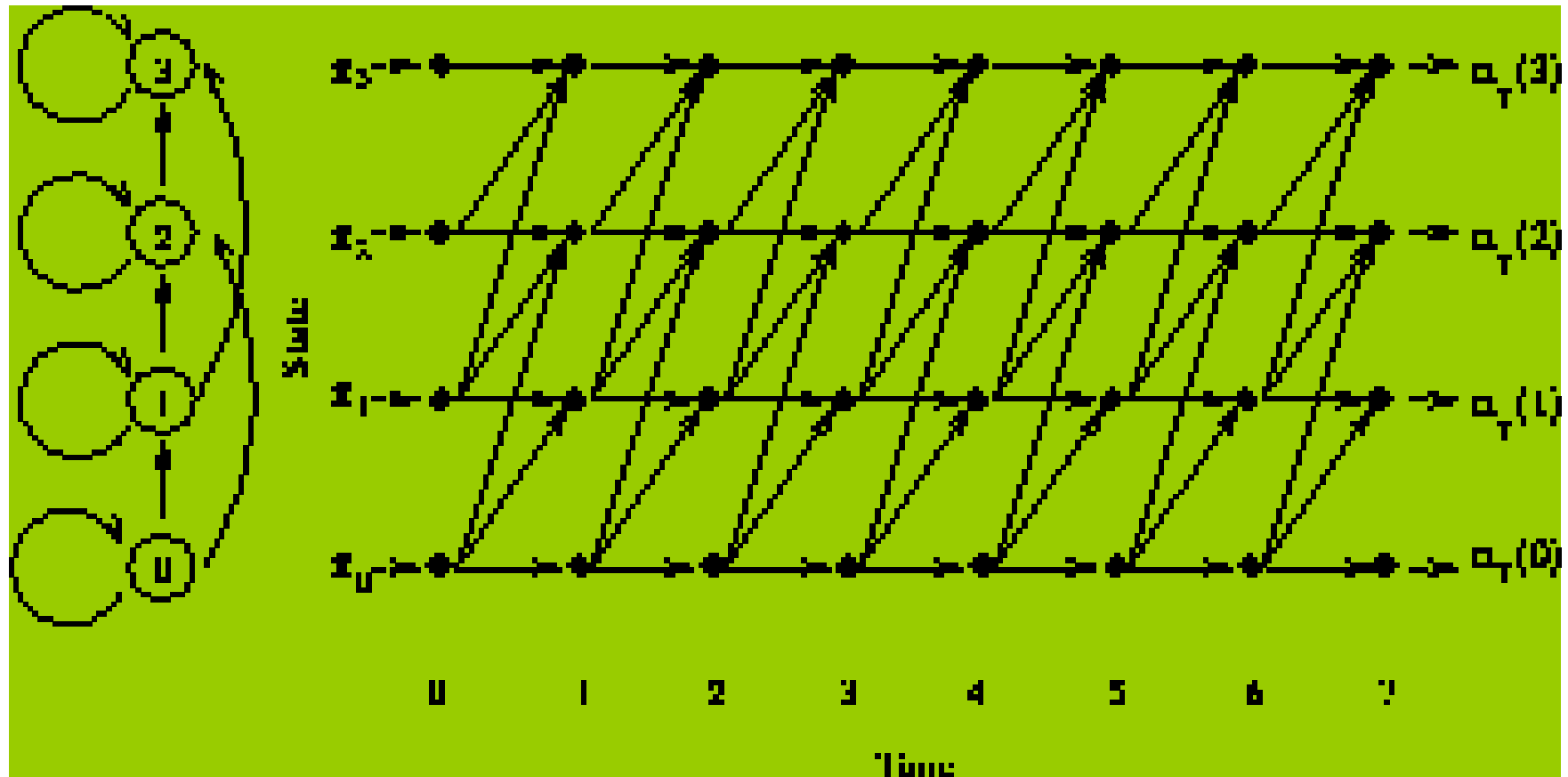




# Hidden Markov Model



# Searching the Speech Signal Trellis



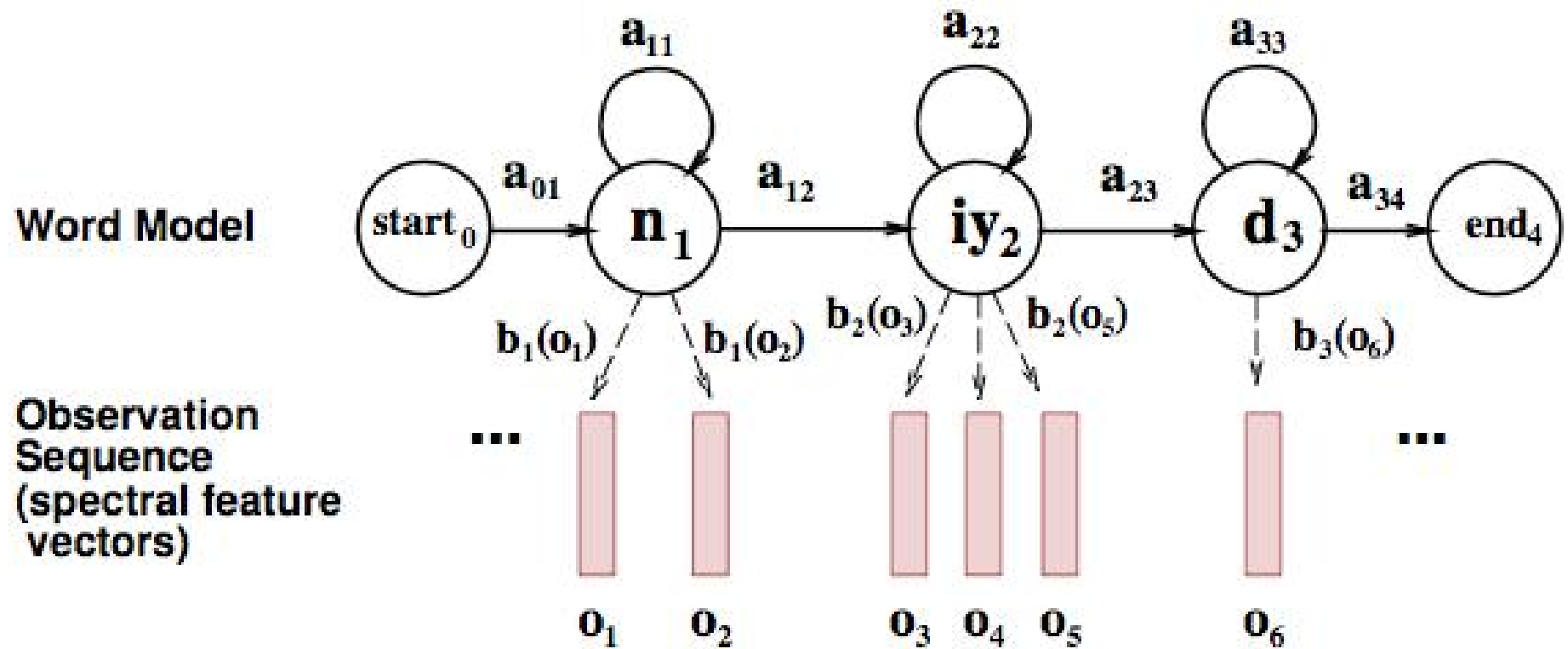


# Speech recognition system

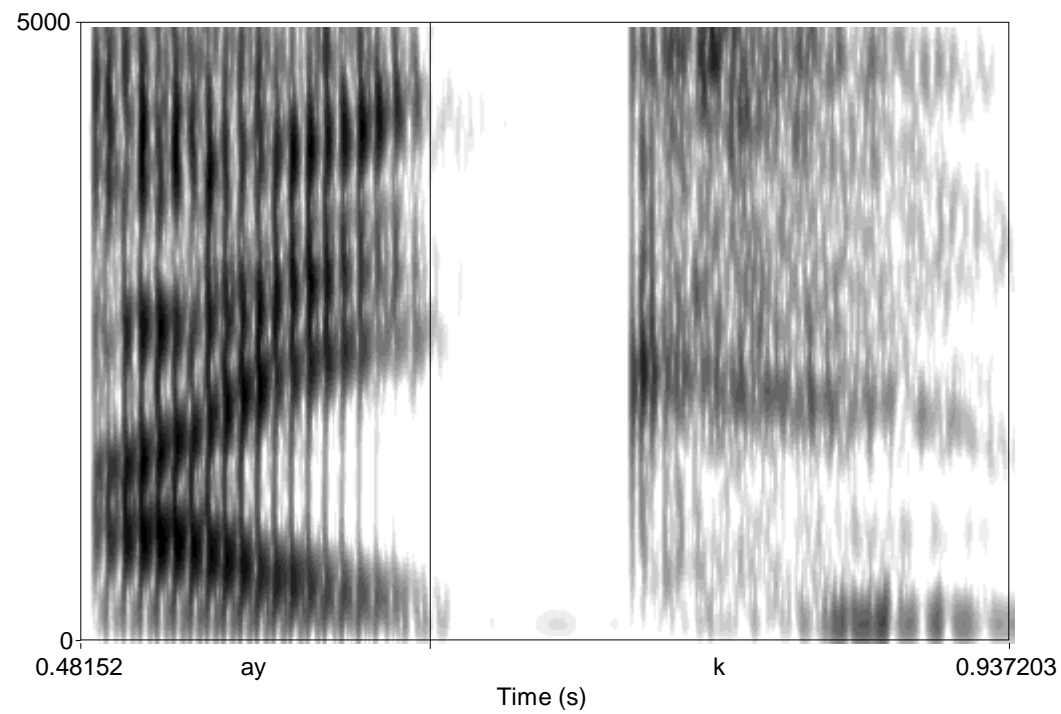
- Acoustic modeling
- **Lexicon**
- Lenguaje Model
- Speech recognition

- $P(E | X)$  encodes which acoustic vectors are appropriate for each phoneme (each kind of sound)
- $P(X | X')$  encodes how sounds can be strung together
- We will have one state for each sound in each word
- From some state  $x$ , can only:
  - Stay in the same state (e.g. speaking slowly)
  - Move to the next position in the word
  - At the end of the word, move to the start of the next word
- We build a little state graph for each word and chain them together to form our state space  $X$

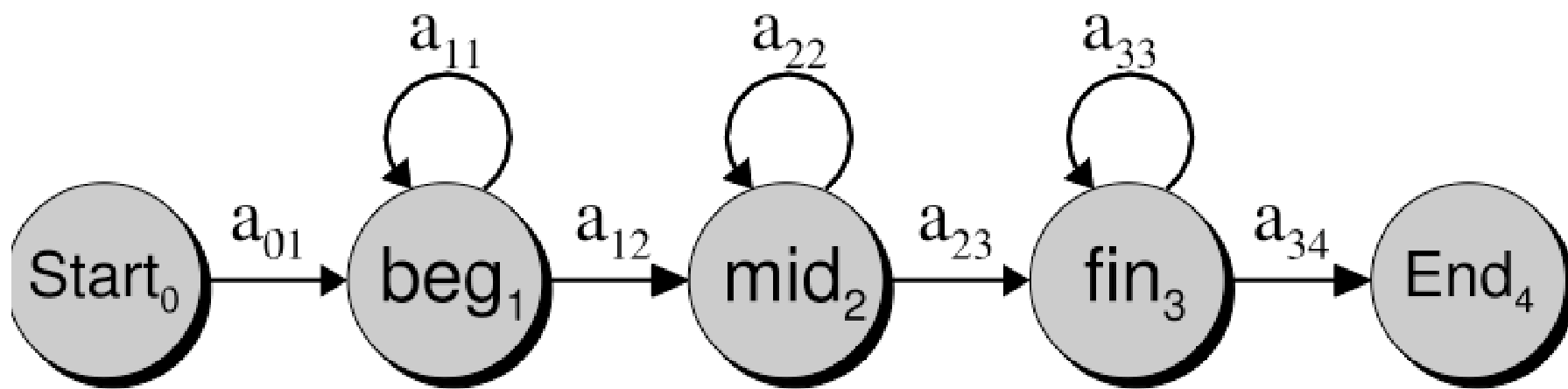
# HMMs for Speech



# Phones are not homogeneous!

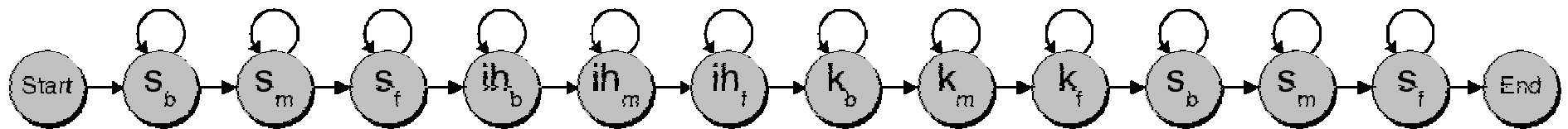


Each phone has 3 subphones

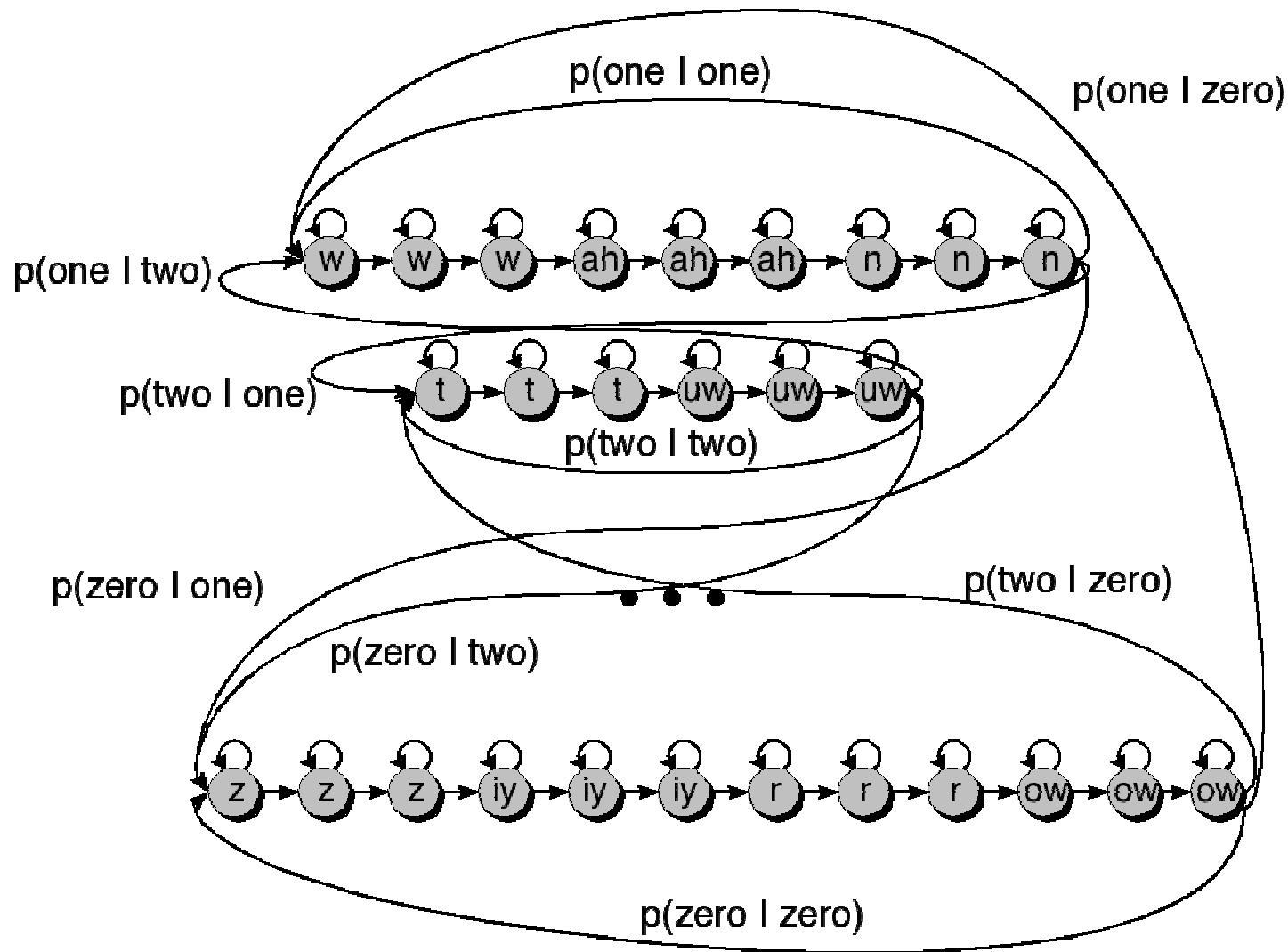




## Resulting HMM word model for "six"



# Search space with bigrams



# Markov Process with Bigrams

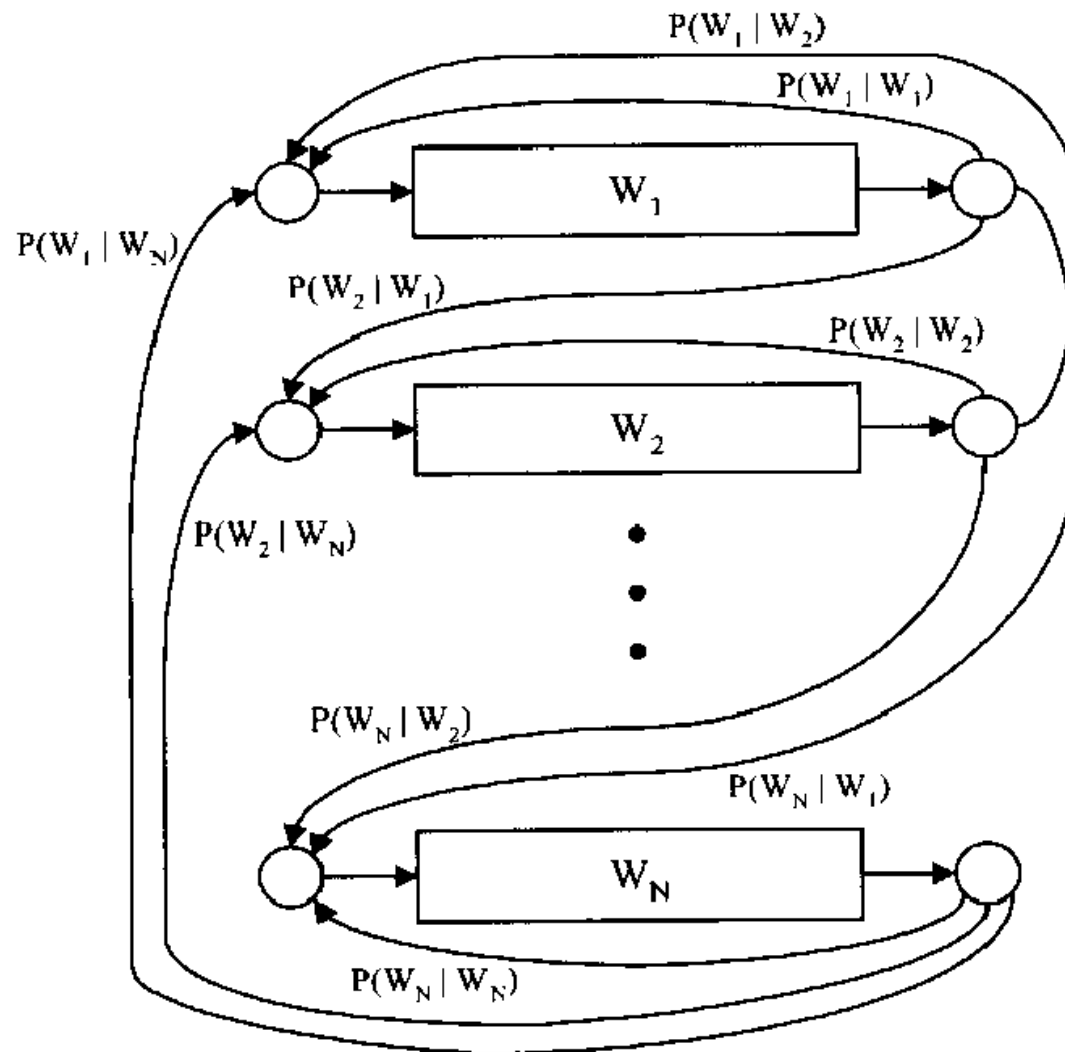
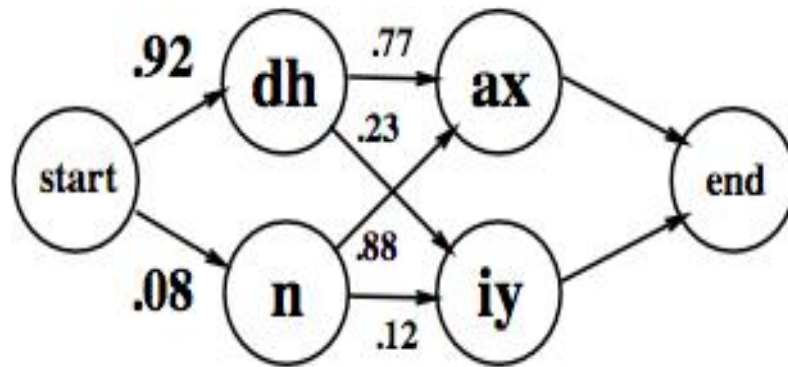
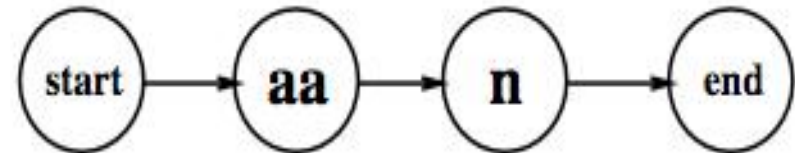


Figure from Huang et al page 618

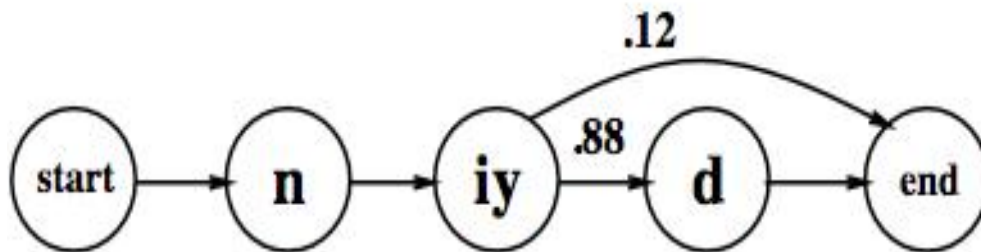
# ASR Lexicon: Markov Models



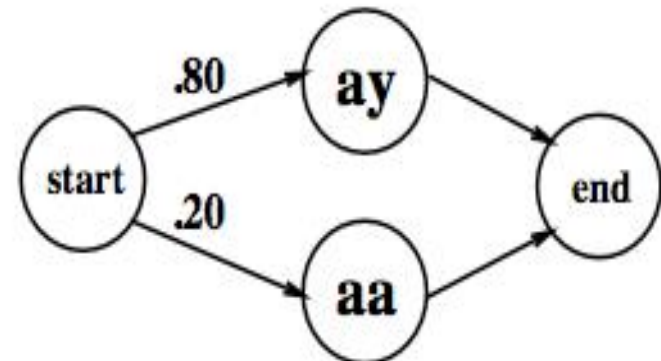
Word model for "the"



Word model for "on"



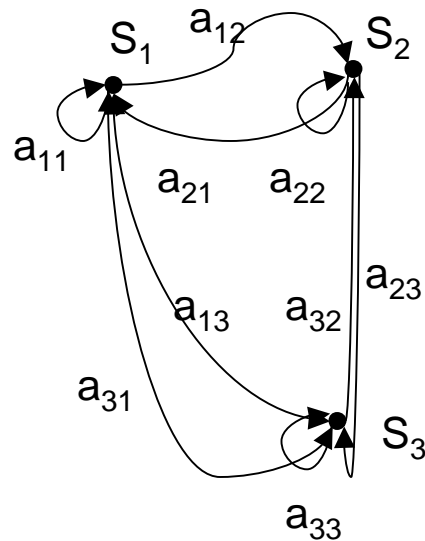
Word model for "need"



Word model for "I"

- Discrete Markov process
- Hidden Markov Model
  - Vitterbi algorithm
  - Forward algorithm
  - Parameter estimation
  - Type of them
- HMMs in ASR Systems
- Popular models
- State of the art and limitation

# Discrete Markov Processes



States= $\{S_1, S_2, S_3\}$   $P(q_t=S_i|q_{t-1}=S_j)=a_{ij}$   
 $\text{Sum}_{(j=1..N)}(a_{ij})=1$

Consider the following model of weather:

- S1: rain or snow
- S2: cloudy
- S3: sunny

$$A = \{a_{ij}\} \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

What is the probability that the weather for the next 3 days will be “sun,sun,rain”.

$O=\{S3,S3,S1\}$  – observation sequence

$$P(O|Model)=P(S3,S3,S1|Model)=P(S3)*P(S3|S3)*P(S3|S1)=1*0.8*0.3=0.24$$

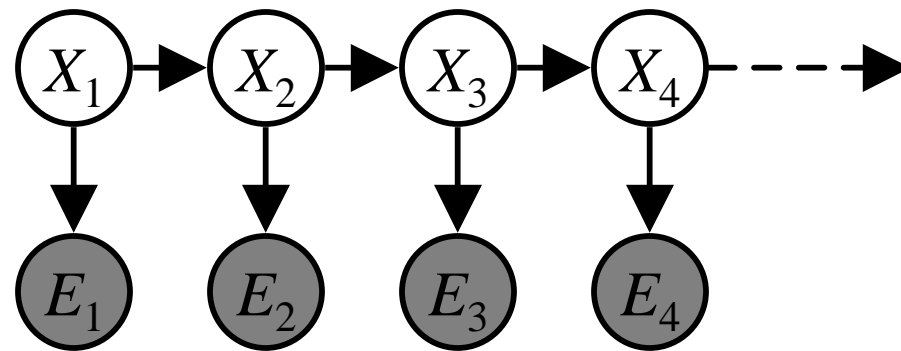
- While there are some practical issues, finding the words given the acoustics is an HMM inference problem
  - Here the state sequence is the sequence of phones
  - Observations are the acoustic vectors
- We want to know which state sequence  $x_{1:T}$  is most likely given the evidence  $e_{1:T}$ :

$$\begin{aligned}x_{1:T}^* &= \arg \max_{x_{1:T}} P(x_{1:T}|e_{1:T}) \\ &= \arg \max_{x_{1:T}} P(x_{1:T}, e_{1:T})\end{aligned}$$



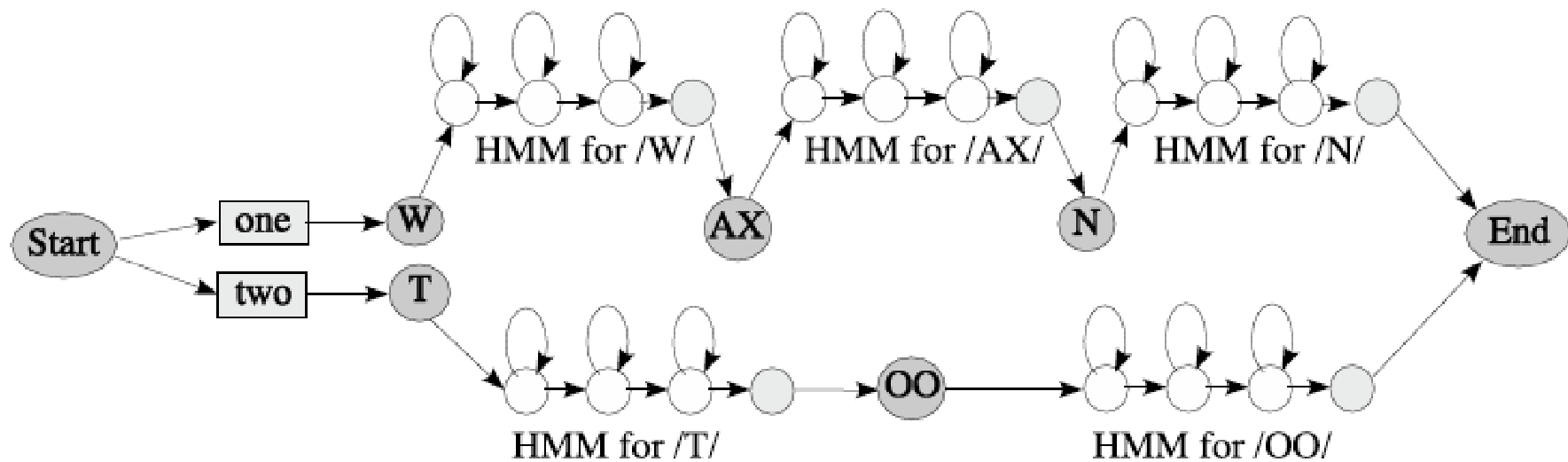
# Hidden Markov Models

- Hidden Markov models (HMMs)
  - Underlying Markov chain over states  $X$
  - You observe outputs (effects)  $E$  at each time step
  - As a Bayes' net:



- Several questions you can answer for HMMs:
  - Last time: filtering to track belief about current  $X$  given evidence
  - Last time: Viterbi estimation to compute most likely sequence

# Search graph



Can be statically or dynamically constructed

# Hidden Markov Model

The observation is a probabilistic function of the state.

## Teacher-mood-model

Situation:

Your school teacher gave three different types of daily homework assignments:

- A: took about 5 minutes to complete
- B: took about 1 hour to complete
- C: took about 3 hours to complete

Your teacher did not reveal openly his mood to you daily, but you knew that your teacher was either in a bad, neutral, or a good mood for a whole day.

Mood changes occurred only overnight.

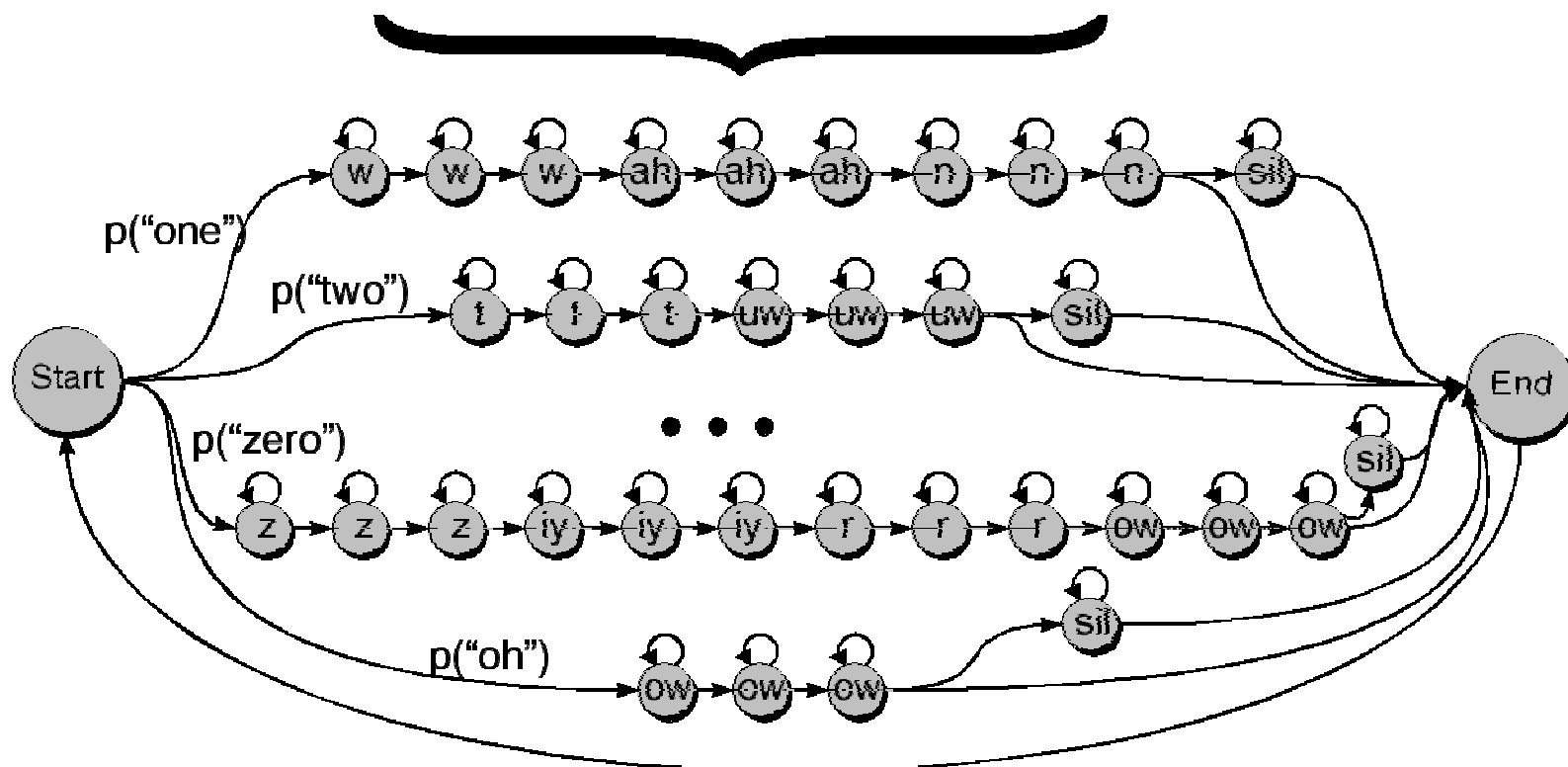
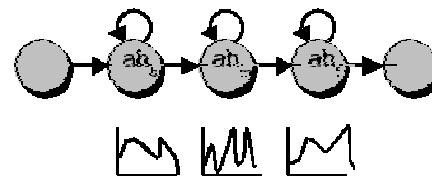
Question: How were his moods related to the homework type assigned that day?

## Lexicon

one	w ah n
two	t uw
three	th r iy
four	f ao r
five	f ay v
six	s ih k s
seven	s eh v ax n
eight	ey t
nine	n ay n
zero	z iy r ow
oh	ow

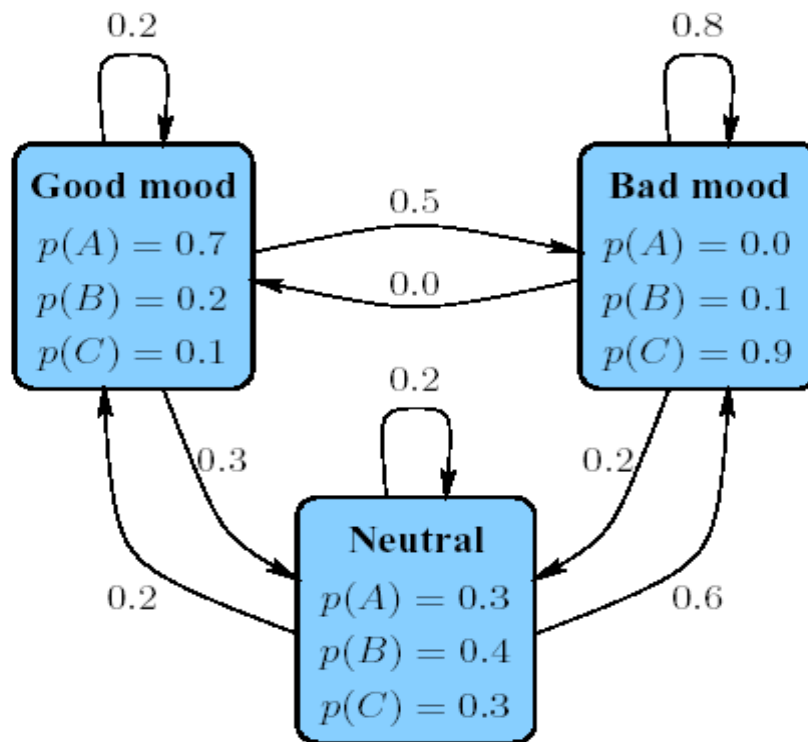
## Hidden Markov Model

### Phone HMM



# Hidden Markov Model

## Model parameters:



$S$  - states {good, neutral, bad}

$a_{kl}$  - probability that state  $l$  is followed by  $k$

$\Sigma$  - alphabet {A,B,C}

$e_k(x)$  - probability that state  $k$  emit symbol  $x \in \Sigma$

# Hidden Markov Model

One week, your teacher gave the following homework assignments:

- Monday: A
- Tuesday: C
- Wednesday: B
- Thursday: A
- Friday: C

Questions:

- What did his mood curve look like most likely that week?  
(Searching for the most probable path - Viterbi algorithm)
- What is the probability that he would assign this order of homework assignments? (Probability of a sequence - Forward algorithm)
- How do we adjust the model parameters  $\lambda(S, a_{ij}, e_i(x))$  to maximize  $P(O | \lambda)$   
(create a HMM for a given sequence set)

## Real HMM Examples

- Speech recognition HMMs:
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)
- Machine translation HMMs:
  - Observations are words (tens of thousands)
  - States are translation positions (dozens)
- Robot tracking:
  - Observations are range readings (continuous)
  - States are positions on a map (continuous)

## HMM - Viterbi algorithm

- Question: what is the most likely state sequence given the observations?
  - Slow answer: enumerate all possibilities
  - Better answer: cached incremental version

$$x_{1:T}^* = \arg \max_{x_{1:T}} P(x_{1:T} | e_{1:T})$$

$$\begin{aligned} m_t[x_t] &= \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, e_{1:t}) \\ &= \max_{x_{1:t-1}} P(x_{1:t-1}, e_{1:t-1}) P(x_t | x_{t-1}) P(e_t | x_t) \\ &= P(e_t | x_t) \max_{x_{t-1}} P(x_t | x_{t-1}) \max_{x_{1:t-2}} P(x_{1:t-1}, e_{1:t-1}) \\ &= P(e_t | x_t) \max_{x_{t-1}} P(x_t | x_{t-1}) m_{t-1}[x_{t-1}] \end{aligned}$$



## Viterbi with 2 Words + Unif. LM

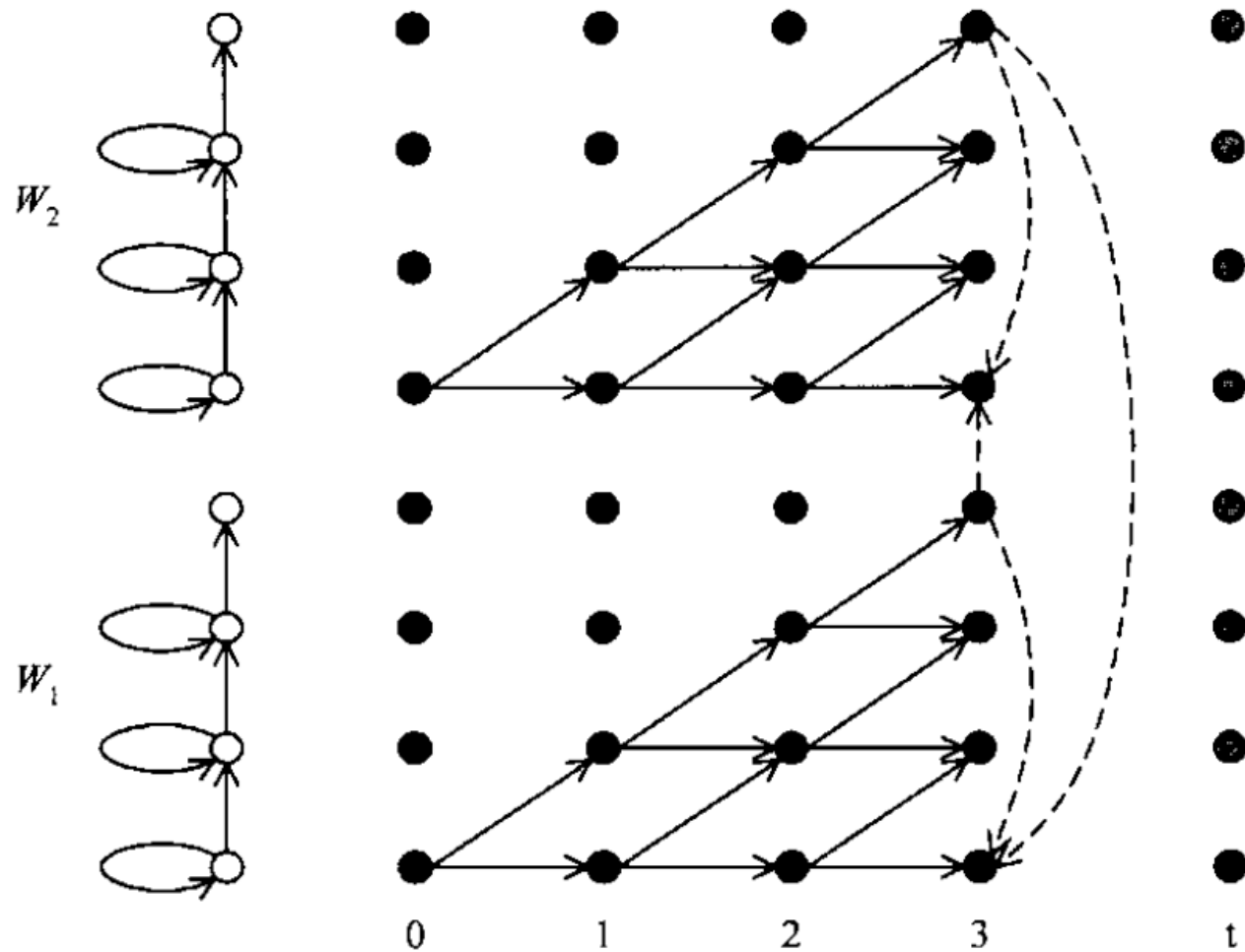
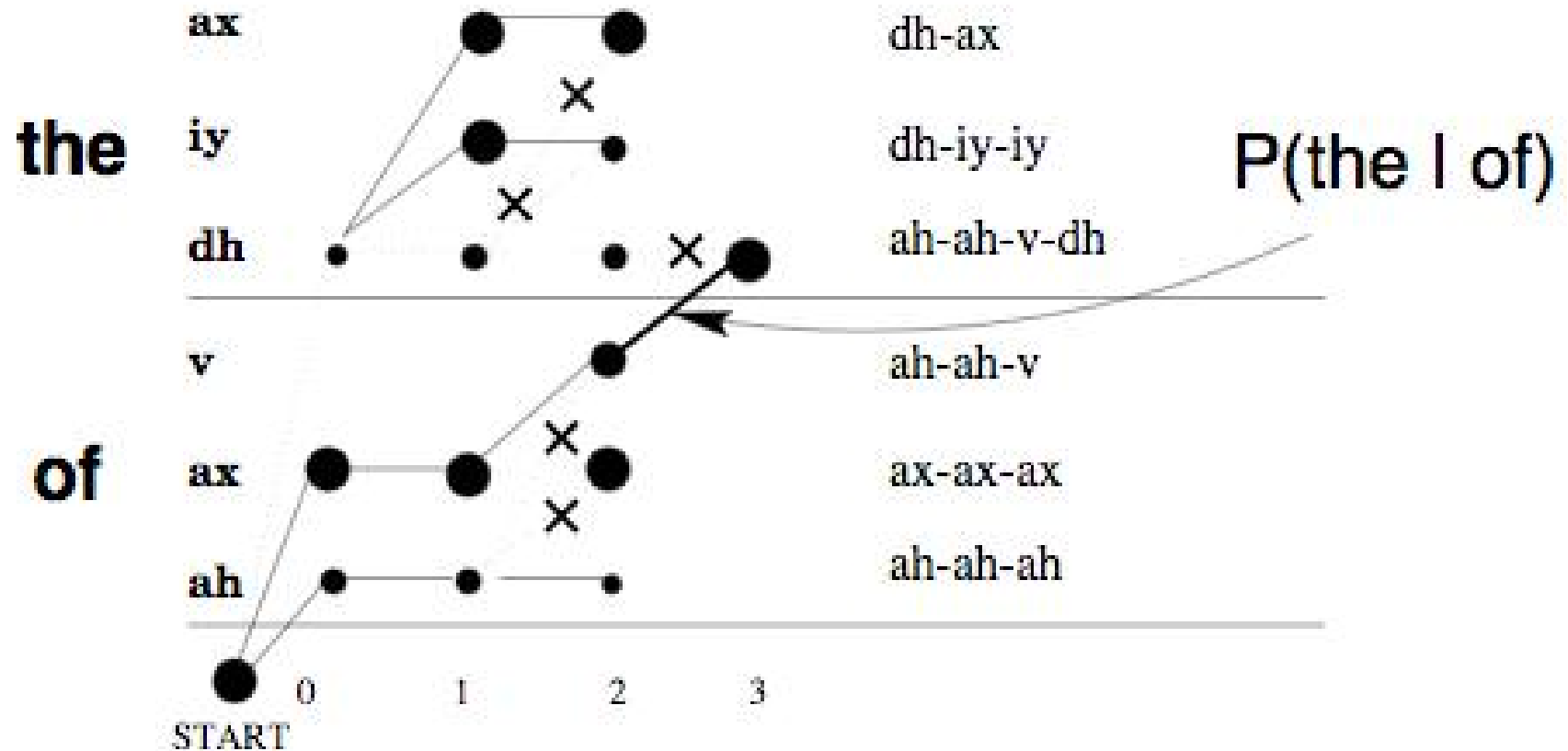


Figure from Huang et al page 612

# Viterbi Decoding



## HMM - Viterbi algorithm

- Empty table

	A	C	B	A	C
good					
neutral					
bad					

## HMM - Viterbi algorithm


Initialization:  $v_l(1) = e_l(A) / \#states$

	<b>A</b>	<b>C</b>	<b>B</b>	<b>A</b>	<b>C</b>
<b>good</b>	0.23				
<b>neutral</b>	0.1				
<b>bad</b>	0.0				

## HMM - Viterbi algorithm

$$v_l(2) = e_l(C) \max_k (v_k(1) a_{kl})$$

	A	C	B	A	C
good	0.23	0.0046			
neutral	0.1	0.0207			
bad	0.0	0.1035			



## HMM - Viterbi algorithm

$$v_l(3) = e_l(B) \max_k (v_k(2) a_{kl})$$

	A	C	B	A	C
good	0.23	0.0046	.000828		
neutral	0.1	0.0207	.00828		
bad	0.0	0.1035	.00828		

## HMM - Viterbi algorithm

Maximum entry in last column

	A	C	B	A	C
good	0.23	0.0046	.000828	.0011592	.0000231
neutral	0.1	0.0207	.00828	.006624	.0001043
bad	0.0	0.1035	.00828	0	.0005216

## HMM - Viterbi algorithm

Reconstruct path along pointers

	A	C	B	A	C
good	0.23	0.0046	.000828	.0011592	.0000231
neutral	0.1	0.0207	.00828	.006624	.0001043
bad	0.0	0.1035	.00828	0	.0005216

The diagram illustrates the reconstruction of the Viterbi path through a sequence of states (A, C, B, A, C) across three classes (good, neutral, bad). Red arrows indicate the path chosen at each step: from 'good' to 'neutral' to 'bad' in the first three steps, and from 'neutral' to 'bad' in the last two steps. The final path is 'good' -> 'neutral' -> 'bad' -> 'neutral' -> 'bad'.



Question:

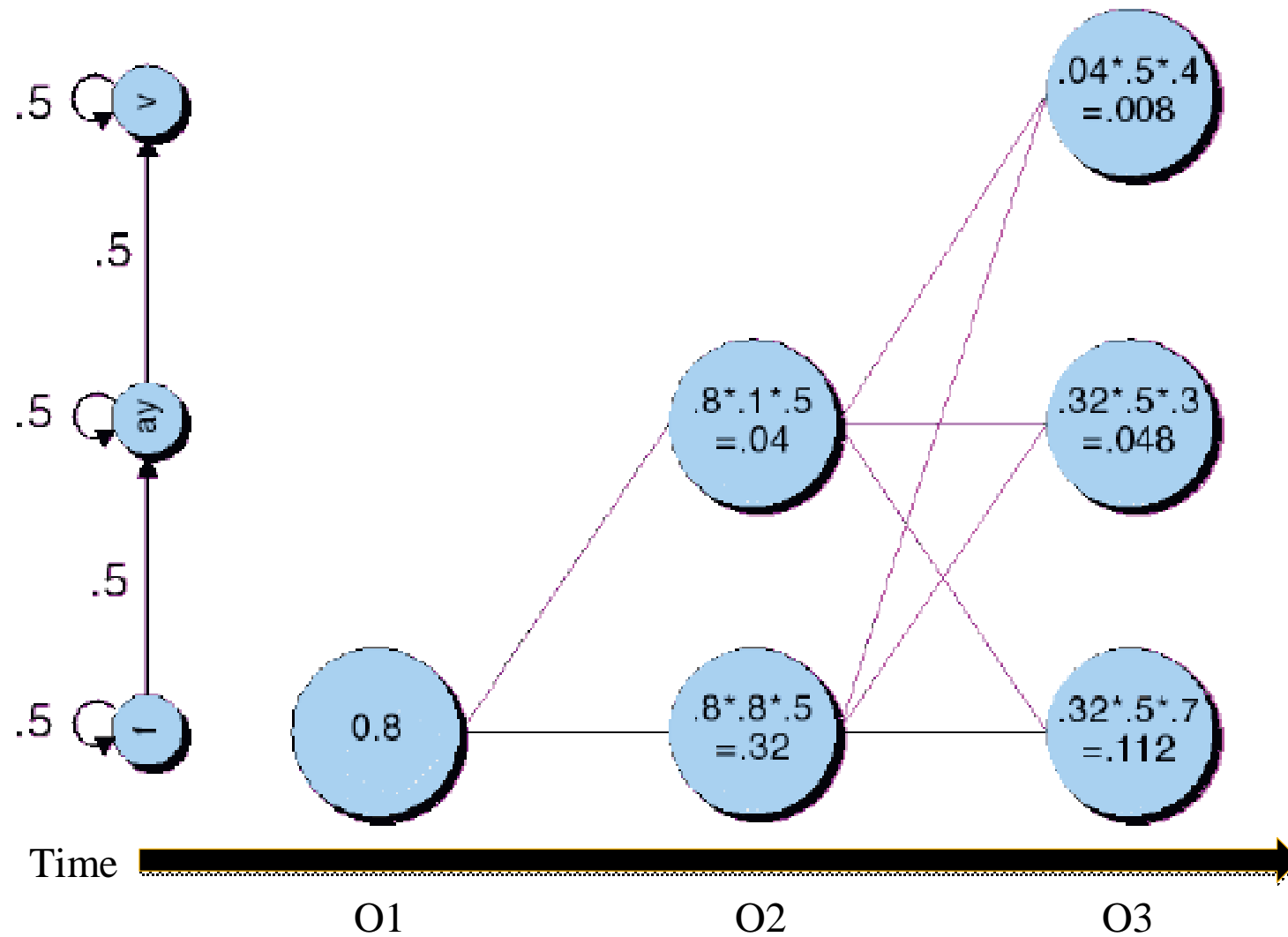
What did his mood curve look like most likely that week?

Answer:

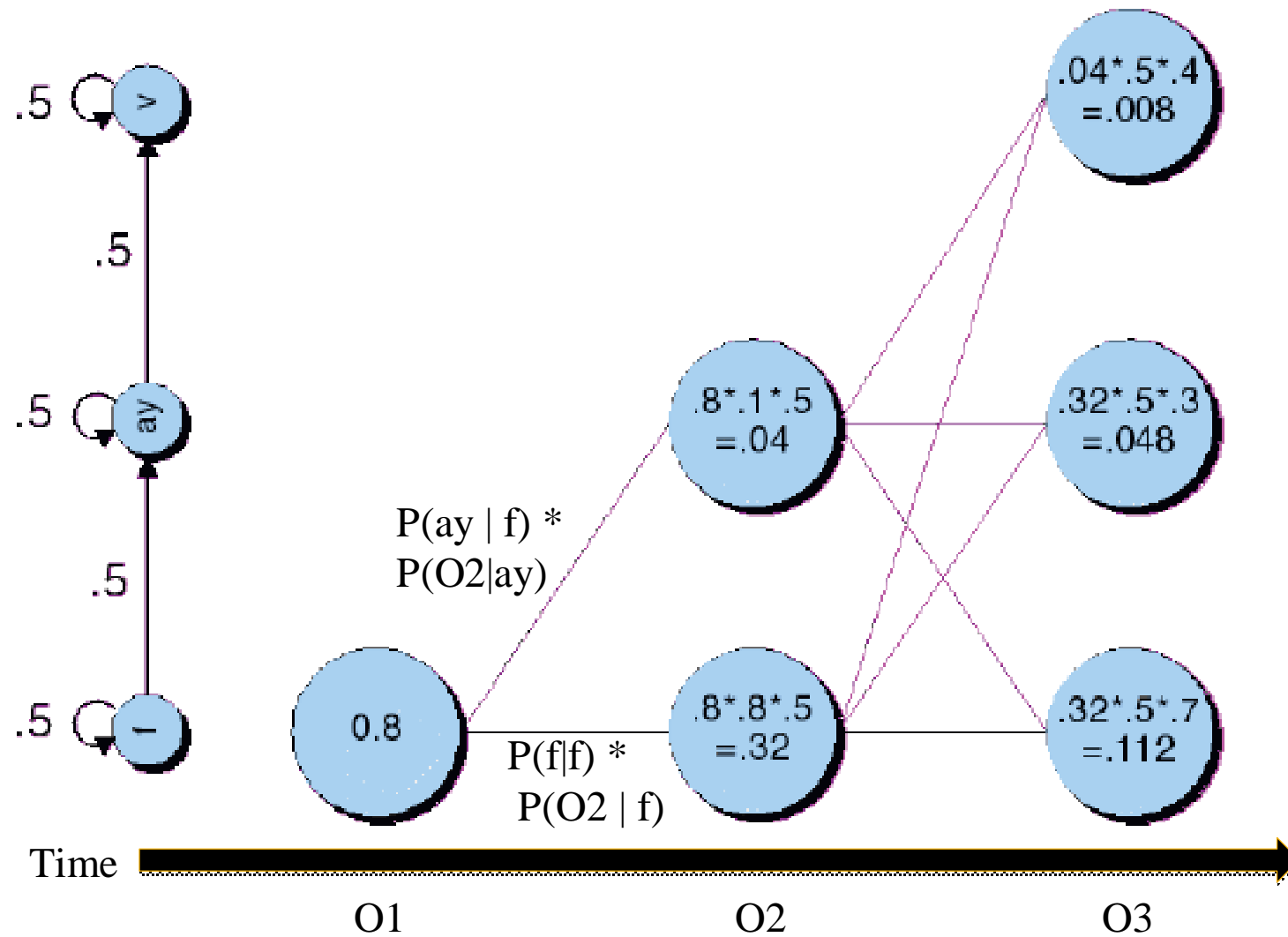
Most probable mood curve:

Day:	Mon	Tue	Wed	Thu	Fri
Assignment:	A	C	B	A	C
Mood:	good	bad	neutral	good	bad

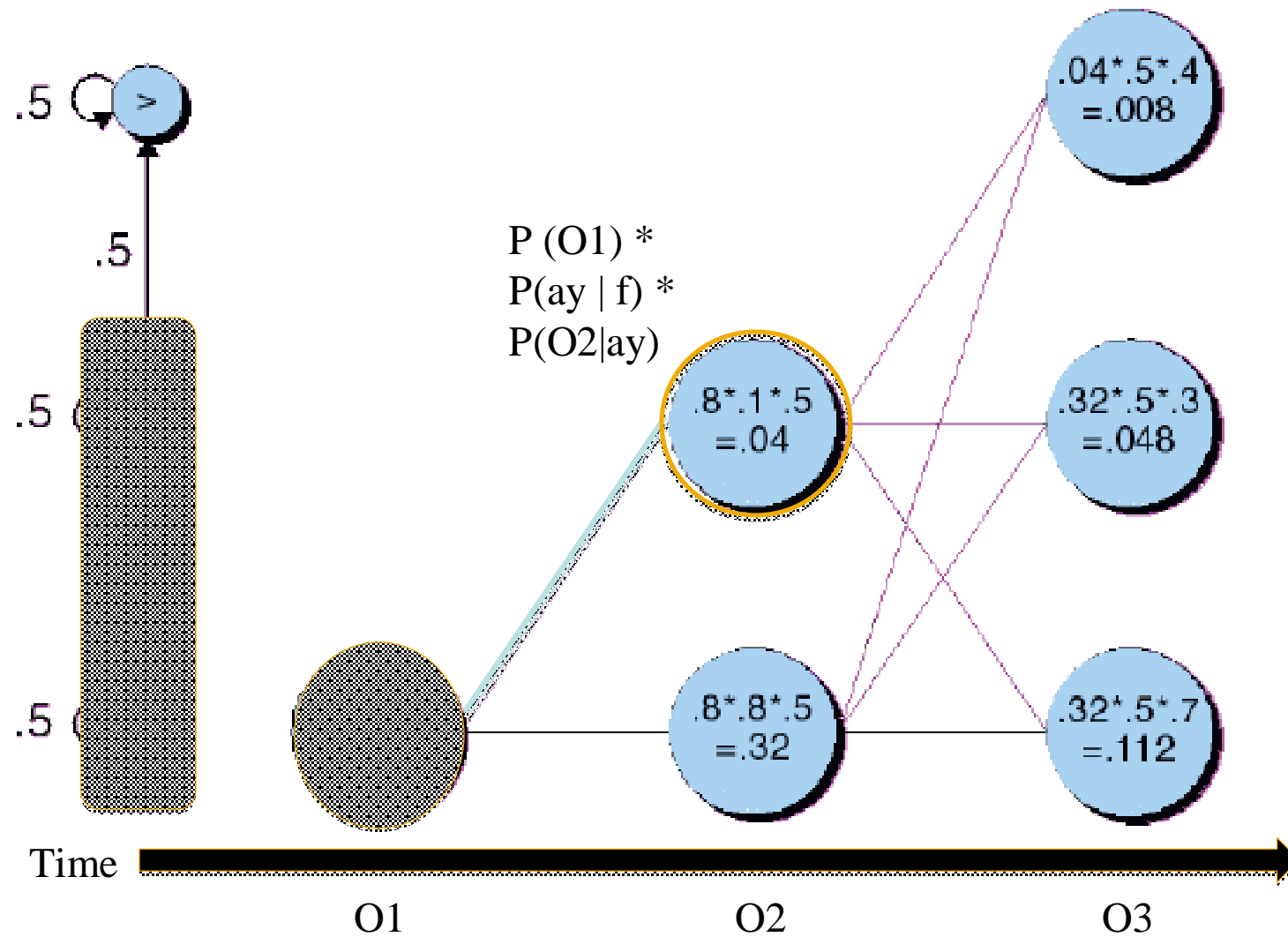
# Viterbi Algorithm



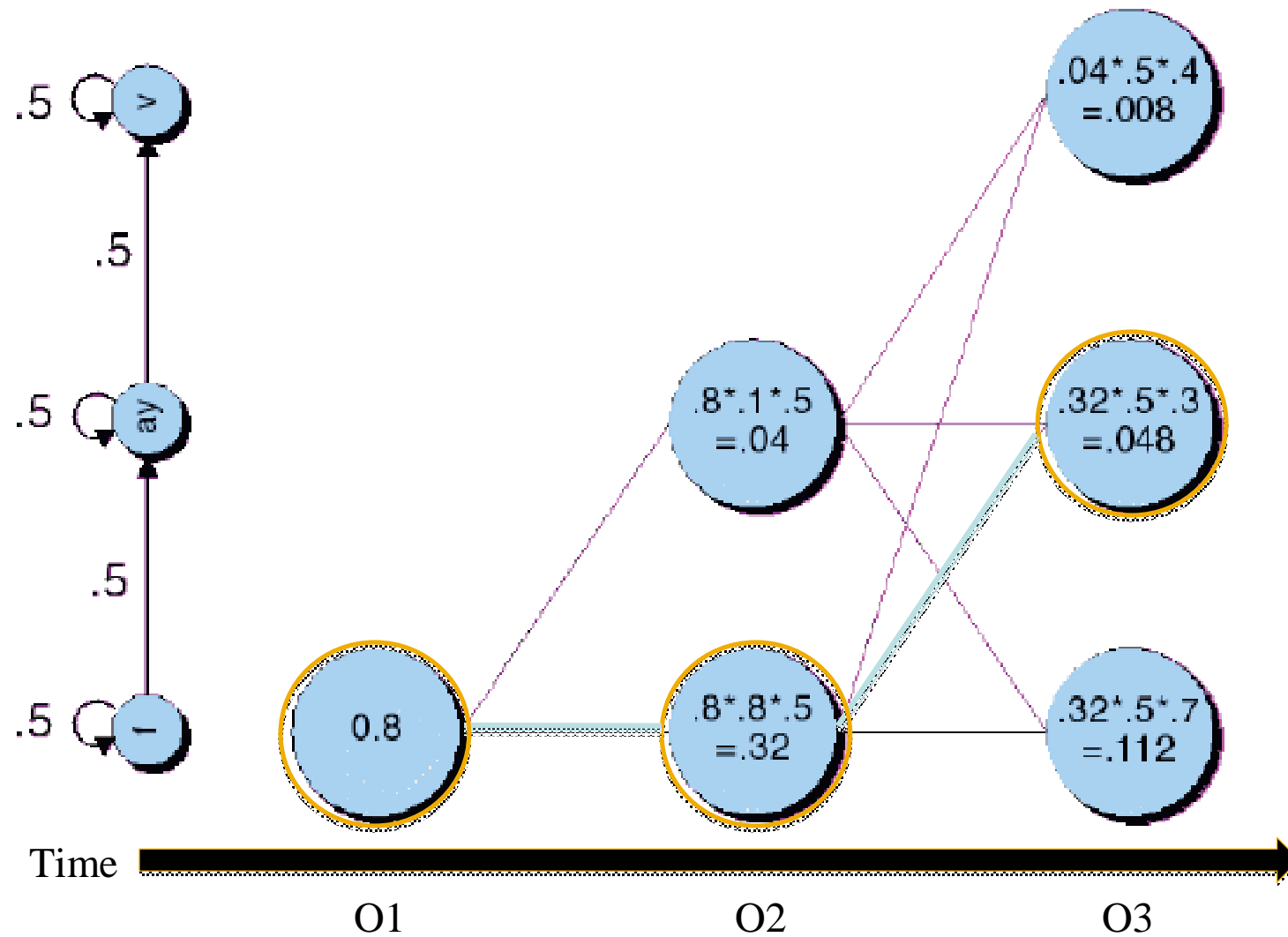
# Viterbi Algorithm



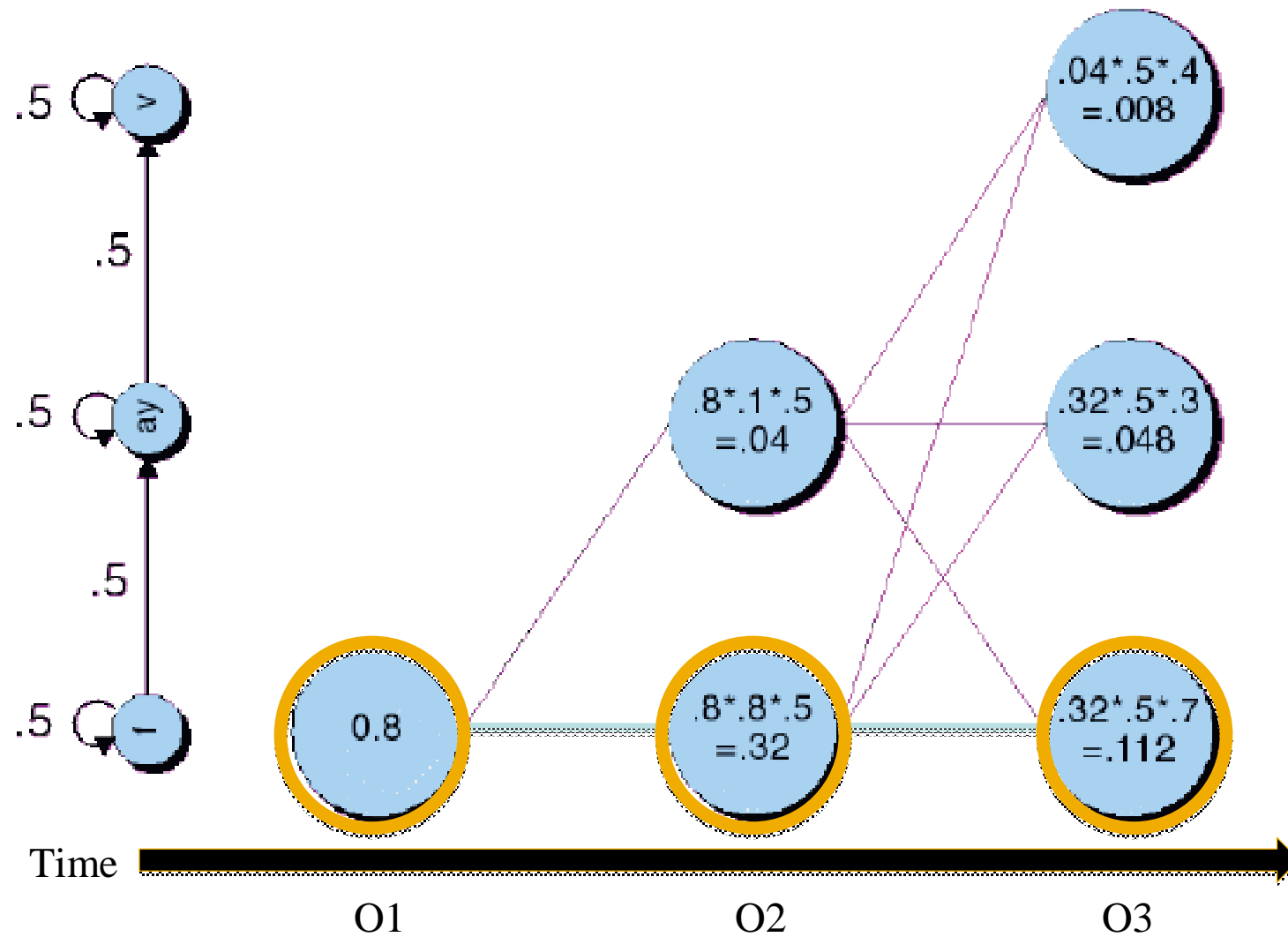
# Viterbi Algorithm



# Viterbi Algorithm



# Viterbi Algorithm



# HMM - Forward algorithm

Used to test the probability of a sequence.

Given:

- Hidden Markov model:  $S, a_{kl}, e_l(x)$ .
- Observed symbol sequence  $E = x_1; : : : ; x_n$ .

What is the probability of  $E$ .

Let  $f_k(i)$  be the probability of the symbol sequence  $x_1, x_2, \dots, x_i$  ending in state  $k$ . Then:

$$\begin{aligned} f_l(i+1) &= \sum_k (e_l(x_{i+1}) * f_k(i) * a_{kl}) \\ &= e_l(x_{i+1}) * \sum_k (f_k(i) * a_{kl}) \end{aligned}$$

## HMM - Forward algorithm

Matrix  $f_k(i)$ , where  $k \in S$  and  $1 \leq i \leq n$ .

- Initialization:  $f_k(1) = e_k(x_1)$  for all states  $k \in S$ .
- $f_k(i) = e_k(x_i) \sum_{l \in S} f_l(i-1) a_{lk}$  for all states  $k \in S$ ,  $i \leq 2$ .

### Algorithm

- Iteratively build up matrix  $f_k(i)$ .
- Probability of symbol sequence is sum of entries in last column.



## HMM - Forward algorithm

Initialization:  $f_l(1) = e_l(A) / \#states$

	<b>A</b>	<b>C</b>	<b>B</b>	<b>A</b>	<b>C</b>
<b>good</b>	0.23				
<b>neutral</b>	0.1				
<b>bad</b>	0.0				

## HMM – Forward algorithm

$$f_l(i+1) = e_l(x_{i+1}) \sum_k (f_k(i) a_{kl})$$

	A	C	B	A	C
good	0.23	0.0066	.001332	.0023604	.0000857
neutral	0.1	0.0267	.015528	.0019235	.0003278
bad	0.0	0.1575	.014532	0	.0021009

## HMM – Forward algorithm

Probability of symbol sequence is sum of entries in last column.

	<b>A</b>	<b>C</b>	<b>B</b>	<b>A</b>	<b>C</b>
<b>good</b>	0.23	0.0066	.001332	.0023604	.0000857
<b>neutral</b>	0.1	0.0267	.015528	.0019235	.0003278
<b>bad</b>	0.0	0.1575	.014532	0	.0021009

$$\Sigma .0025144$$

Question:

- How do we know the transition probabilities  $a_{kl}$  and the emission probabilities  $e_k(x)$ ?

Answer:

- We use training sequences and construct maximum likelihood estimators.

# HMM - Parameter estimation

Case 1: State sequences of training sequences are known.

We construct maximum likelihood estimators.

Let  $A_{kl}$  be the number of transitions from state  $k$  into state  $l$  in training data (+ constant).

Let  $E_k(x)$  be the number of emissions of symbol  $x$  from state  $k$  in training data (+ constant).

Transition estimator:

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

Emission estimator:

$$e_k(x) = \frac{E_k(x)}{\sum_{x'} E_k(x')}$$

## HMM - Parameter estimation

Case 2: State sequences of training sequences are not known.

*Viterbi training:* We iteratively use the Viterbi algorithm to compute the most probable paths and set the estimators (from case 1) according to this data.

Algorithm sketch:

Initialization: Pick arbitrary model parameters.

REPEAT

Set all  $A_{kl}$  and  $E_k(x)$  to their constant value

FOR each training sequence

Compute most probable state paths (Viterbi)

Add contribution to  $A_{kl}$  and  $E_k(x)$ .

Update  $a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$  and  $e_k(x) = \frac{E_k(x)}{\sum_{x'} E_k(x')}$ .

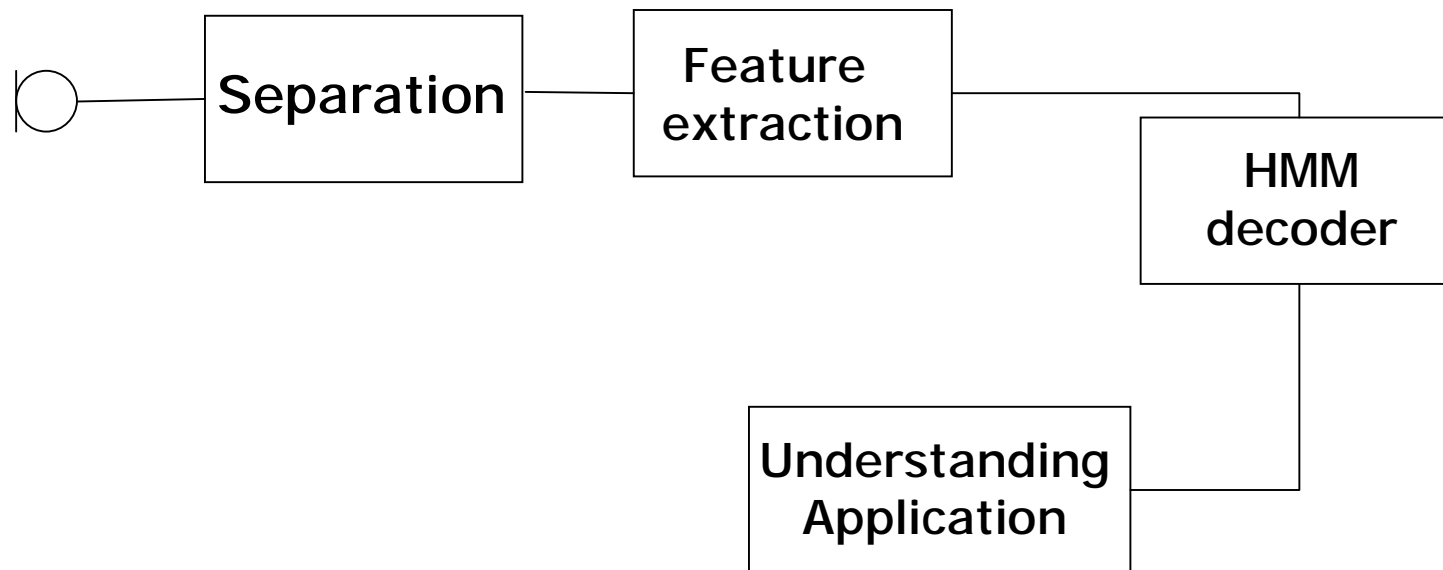
UNTIL stopping criterion is reached.

- Till now I was speaking about HMMs with discrete output (finite  $|\Sigma|$ ).

### Extensions:

- continuous observation probability density function
- mixture of Gaussian pdfs

# HMMs in ASR





How HMM can be used to classify feature sequences to known classes.

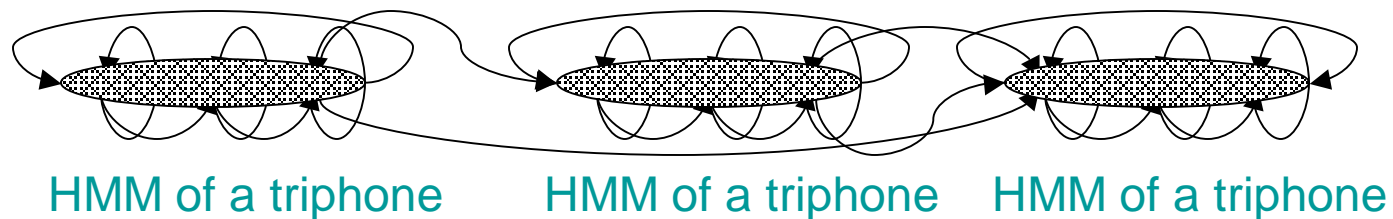
Make an HMM for each class.

By determining the probability of a sequence to the HMMs, we can decide which HMM could most probably generate the sequence.

There are several ideas what to model:

- Isolated word recognition (HMM for each known word)  
Usable just on small dictionaries. (digit recognition etc.)  
Number of states usually  $\geq 4$ . Left-to-right HMM
- Monophone acoustic model (HMM for each phone)  
~50 HMM
- Triphone acoustic model (HMM for each three phone sequence)  
 $50^3 = 125000$  triphones  
each triphone has 3 states

- Hierarchical system of HMMs



Language model

## Typical state-of-the-art large-vocabulary ASR system:

- speaker independent
- 64k word vocabulary
- trigram (2-word context) language model
- multiple pronunciations for each word
- triphone or quinphone HMM-based acoustic model
- 100-300X real-time recognition
- WER 10%-50%

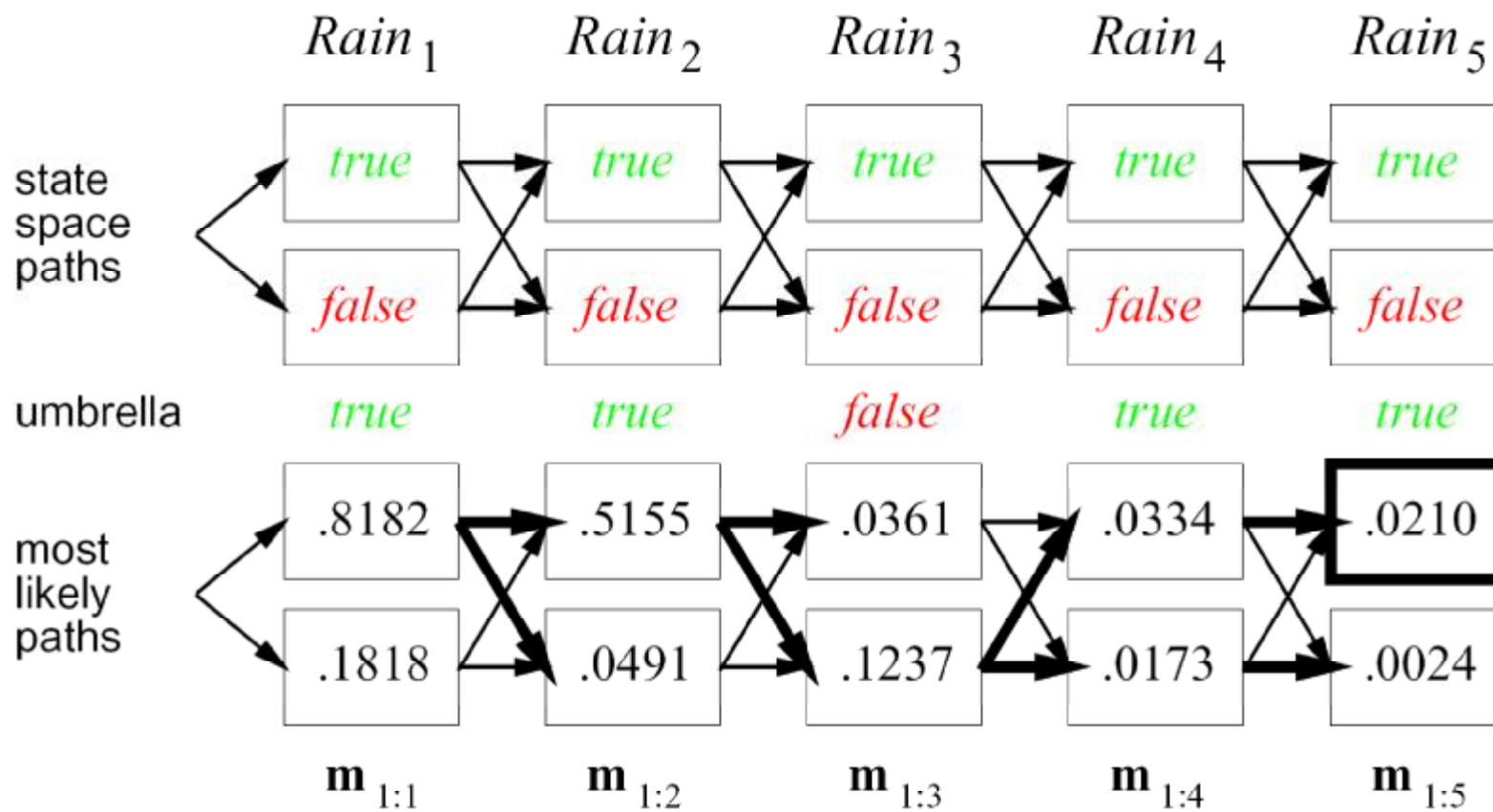
- Data intensive
- Computationally intensive
  - 50 phones = 125000 possible triphones
    - 3 states per triphone
    - 3 Gaussian mixture for each state
  - 64k word vocabulary
    - 262 trillion trigrams
    - 2-20 phonemes per word in 64k vocabulary
  - 39 dimensional feature vector sampled every 10ms
    - 100 frame per second

- Medical transcription mainly in radiology and pathology
- First use of speech recognition in the field of radiology in 1981
- Mean accuracy rate of reading pathology reports, using IBM Via Voice Pro software - 93.6% compared to human transcription at 99.6%

## Current error rates

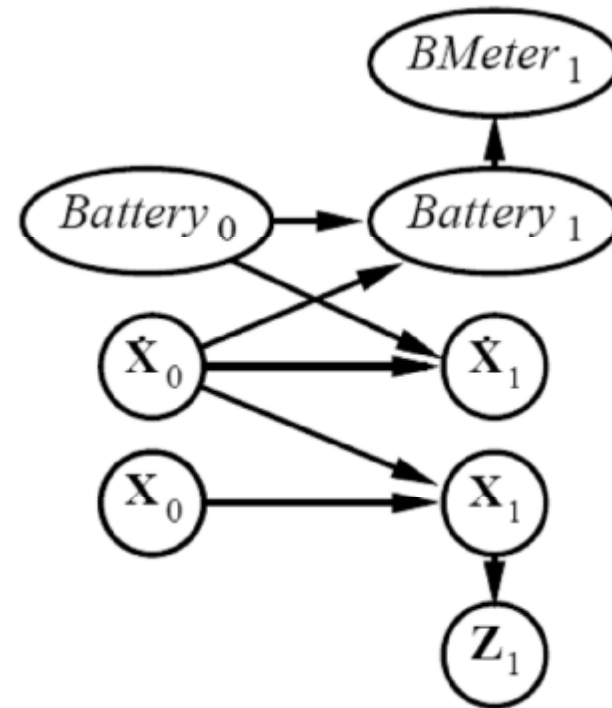
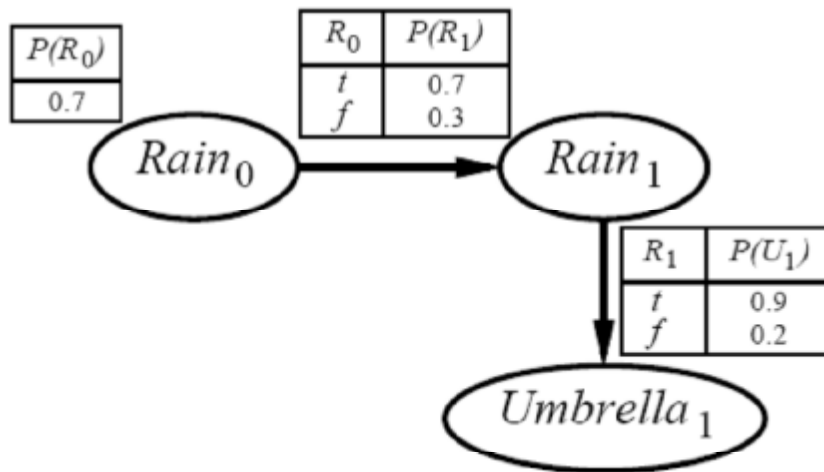
<b>Task</b>	<b>Vocabulary</b>	<b>Error Rate%</b>
<b>Digits</b>	<b>11</b>	<b>0.55</b>
<b>WSJ read speech</b>	<b>5K</b>	<b>3.0</b>
<b>WSJ read speech</b>	<b>20K</b>	<b>&lt;6.6</b>
<b>Broadcast news</b>	<b>64,000+</b>	<b>9.9</b>
<b>Conversational Telephone</b>	<b>64,000+</b>	<b>20.7</b>

# Example



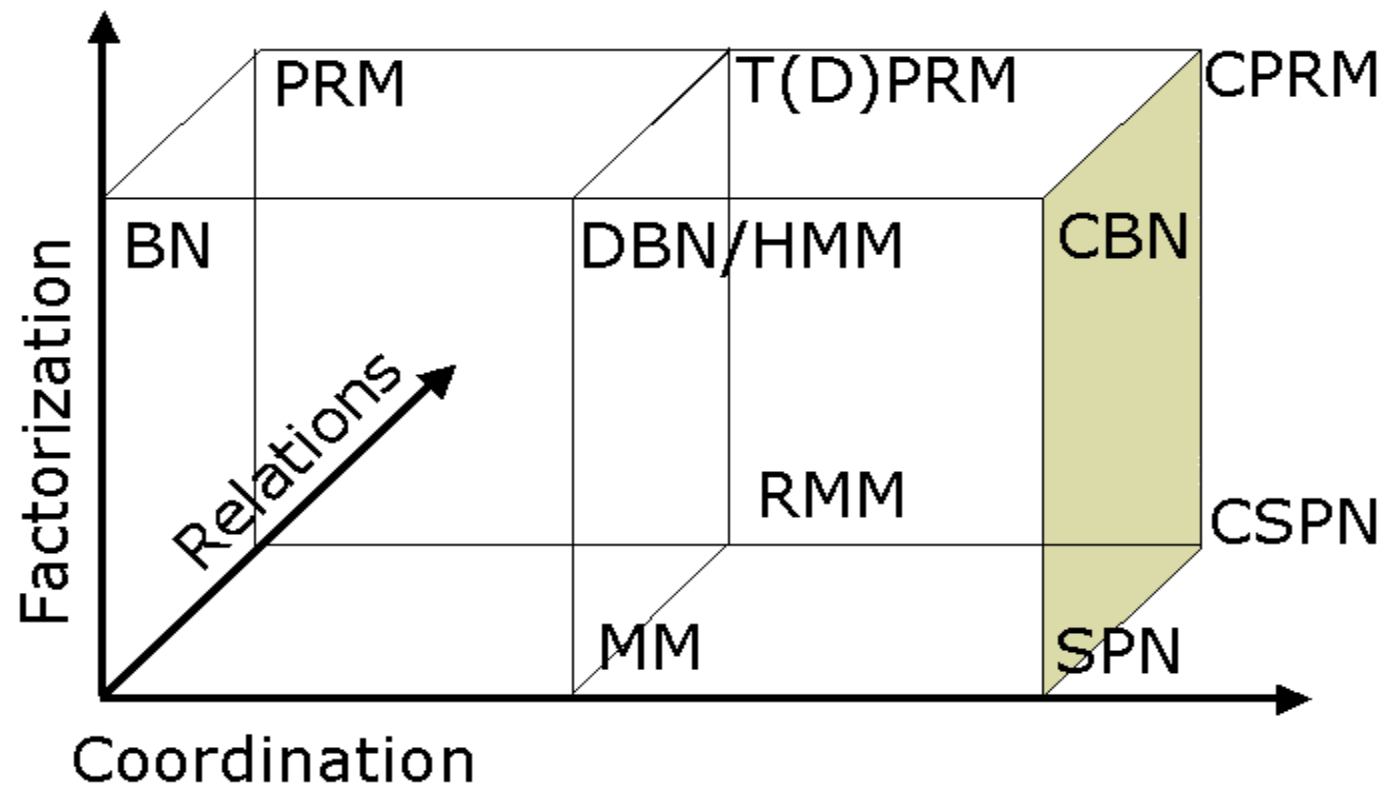
# Dynamic Bayes Nets

DBN = Multiple Hidden State Variables. Each State is a BN





# Structured: Probabilistic Inference





# Speech recognition system

- Acoustic modeling
- Lexicon
- **Lenguaje Model**
- Speech recognition algorithms

- A statistical language model assigns a probability to a sequence of  $m$  words  $P(w_1, \dots, w_n)$  by means of a probability distribution.
- In speech recognition and in data compression, such a model tries to capture the properties of a language, and to predict the next word in a speech sequence.

A language model is a probability distribution over word sequences

$$p(W) = p(w_1, \dots, w_n) = \prod_{i=1}^n p(w_i \mid w_0, \dots, w_{i-1})$$

- $n = 3, 4, 5$  [lose the rest of the context]
- Hard to estimate large contexts: consider  $64,000^3$  words

Need large collections of text

Smoothing  $P(w_i \mid w_{i-2}, w_{i-1})$  is necessary

- Models likelihood of word given previous word(s)
- n-gram models:
  - Build the model by calculating bigram or trigram probabilities from text training corpus
  - Smoothing issues

public <basicCmd> = <startPolite> <command> <endPolite>;

public <startPolite> = (please | kindly | could you ) \*;

public <endPolite> = [ please | thanks | thank you ];

<command> = <action> <object>;

<action> = (open | close | delete | move);

<object> = [the | a] (window | file | menu);

- Constructs the search graph of HMMs from:
  - Acoustic model
  - Statistical Language model ~or~
  - Grammar
  - Dictionary
- Linguist types
  - FlatLinguist
  - DynamicFlatLinguist
  - LexTreeLinguist

- Example from cmudict.06d

POULTICE	P OW L T AH S
POULTICES	P OW L T AH S IH Z
POULTON	P AW L T AH N
POULTRY	P OW L T R IY
POUNCE	P AW N S
POUNCED	P AW N S T
POUNCEY	P AW N S IY
POUNCING	P AW N S IH NG
POUNCY	P UW NG K IY



# Lexicon - links words to phones in acoustic model

Aaron EH R AX N

Aaron(2) AE R AX N

abandon AX B AE N D AX N

abandoned AX B AE N D AX N DD

abandoning AX B AE N D AX N IX NG

abandonment AX B AE N D AX N M AX N TD

abated AX B EY DX IX DD

abatement AX B EY TD M AX N TD

abbey AE B IY

Abbott AE B AX TD

Abboud AA B UW DD

abby AE B IY

abducted AE BD D AH KD T IX DD

Abdul AE BD D UW L



# Speech recognition system

- Acoustic modeling
- Lexicon
- Lenguaje Model
- **Speech recognition**

# Speech Recognition: The fundamental Equation

O is an acoustical 'Observation'

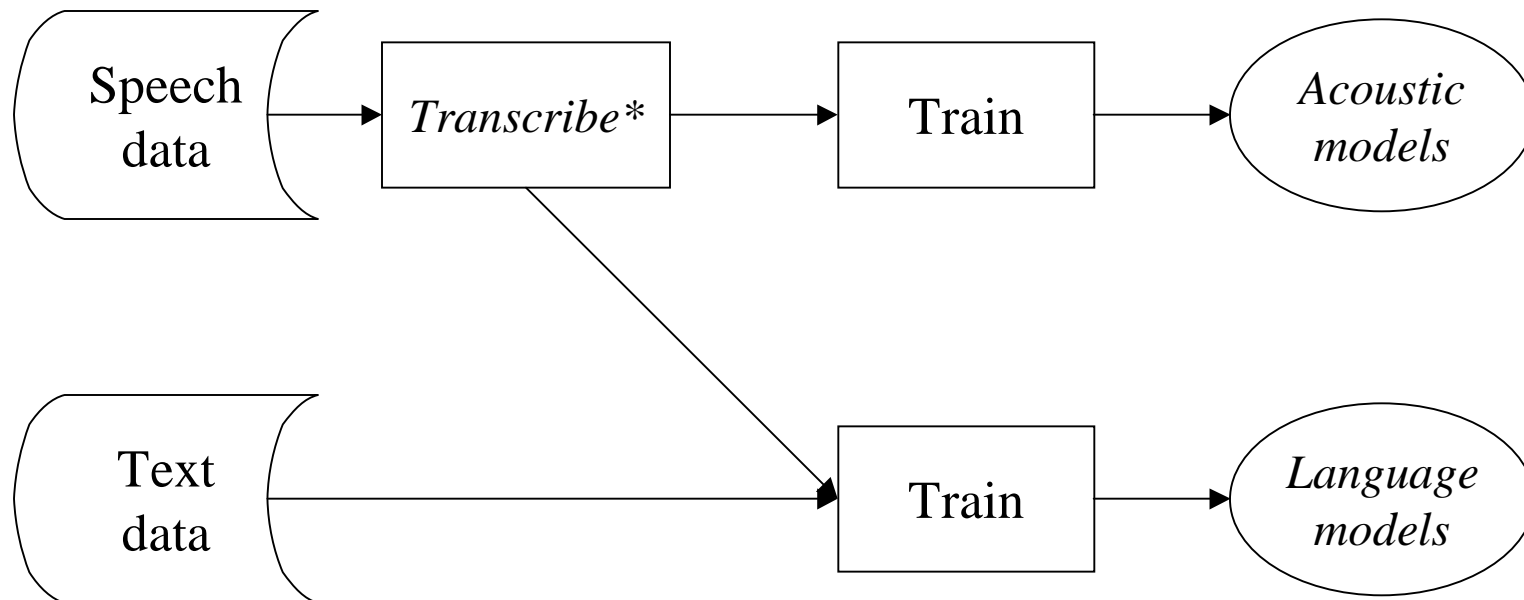
w is a 'word' we are trying to recognize

Maximize  $w = \operatorname{argmax} (P(W) \mid O)$

$P(W \mid O)$  is unknown so by Bayes' rule:

$$P(W \mid O) = \frac{P(O \mid W) P(W)}{P(O)}$$

# Creating models for recognition



- Most common metric
- Measure the # of modifications to transform recognized sentence into reference sentence
- Reference: "This is a reference sentence."
- Result: "This is neuroscience."
- Requires 2 deletions, 1 substitution

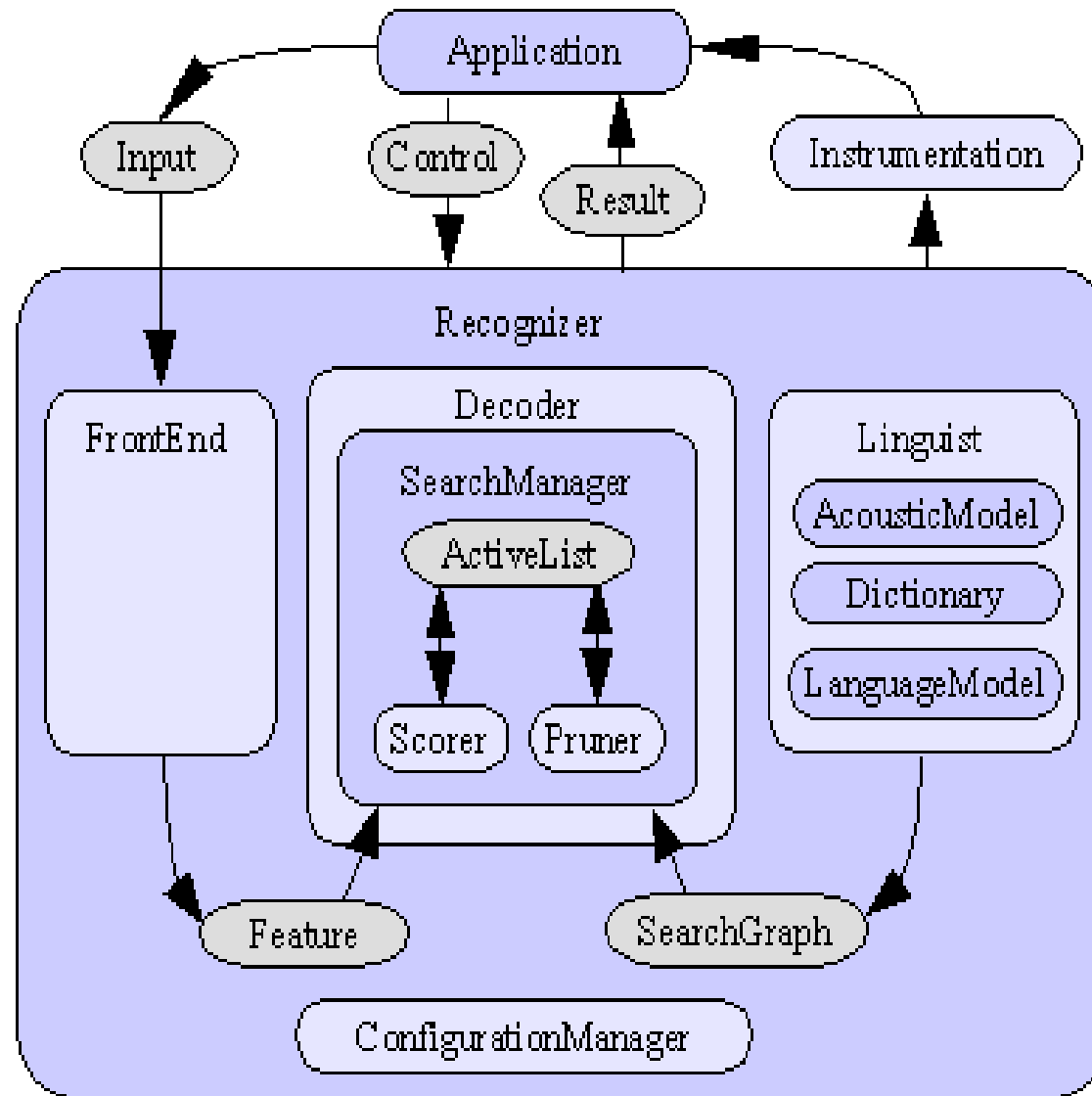
$$WER = 100 \times \frac{\text{deletions} + \text{substitutions} + \text{insertions}}{\text{Length}}$$

$$WER = 100 \times \frac{2 + 1 + 0}{5} = 100 \times \frac{3}{5} = 60\%$$

- Factors include
  - Speaking mode: isolated words vs continuous speech
  - Speaking style: read vs spontaneous
  - “Enrollment”: speaker (in)dependent
  - Vocabulary size (small <20 ... large > 20,000)
  - Equipment: good quality noise-cancelling mic ... telephone
  - Size of training set (if appropriate) or rule set
  - Recognition method

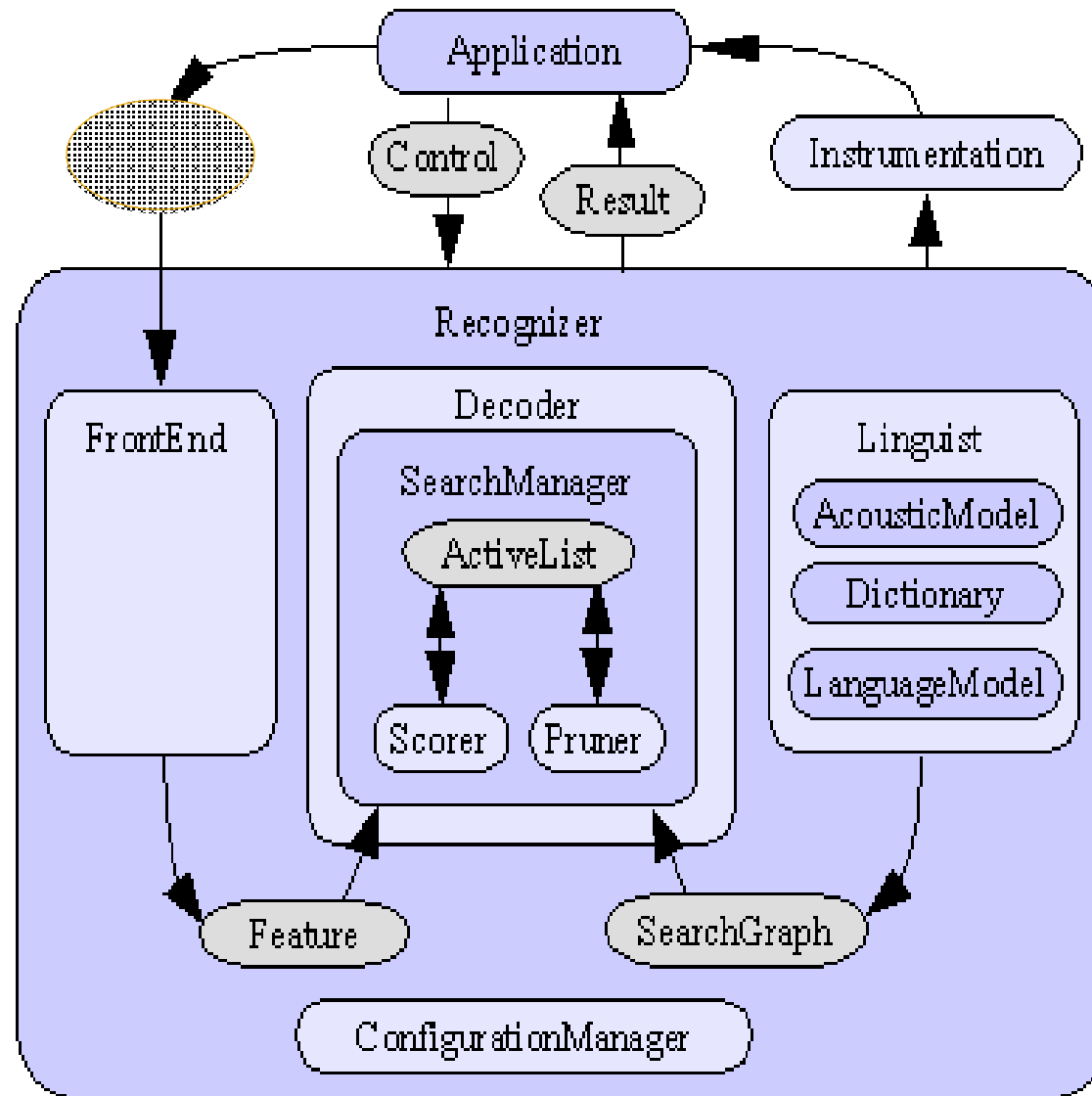
- Funders have been very keen on competitive quantitative evaluation
- Subjective evaluations are informative, but not cost-effective
- For transcription tasks, word-error rate is popular (though can be misleading: all words are not equally important)
- For task-based dialogues, other measures of understanding are needed

# Sphinx4 implementation

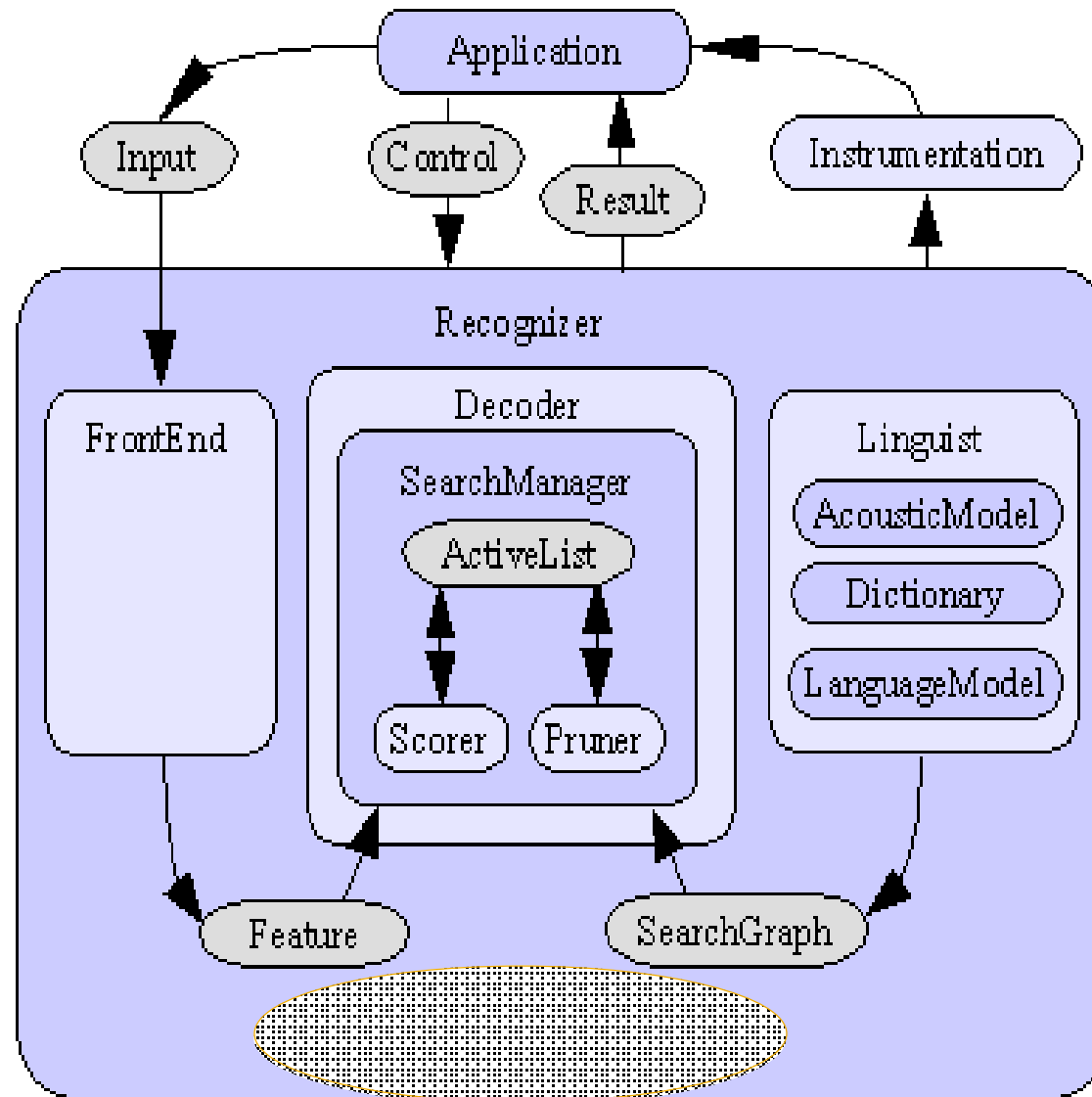




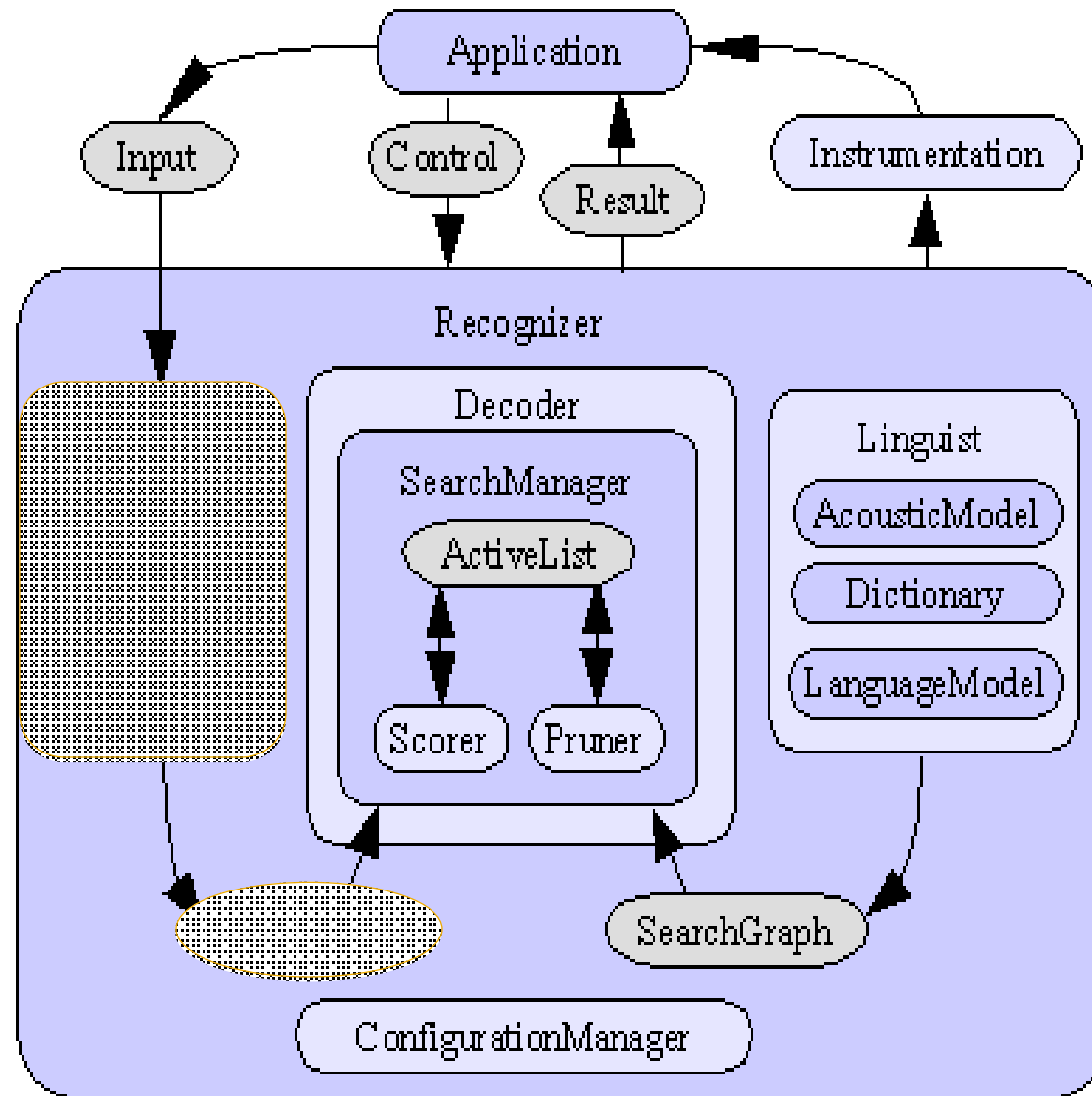
# Sphinx4 implementation



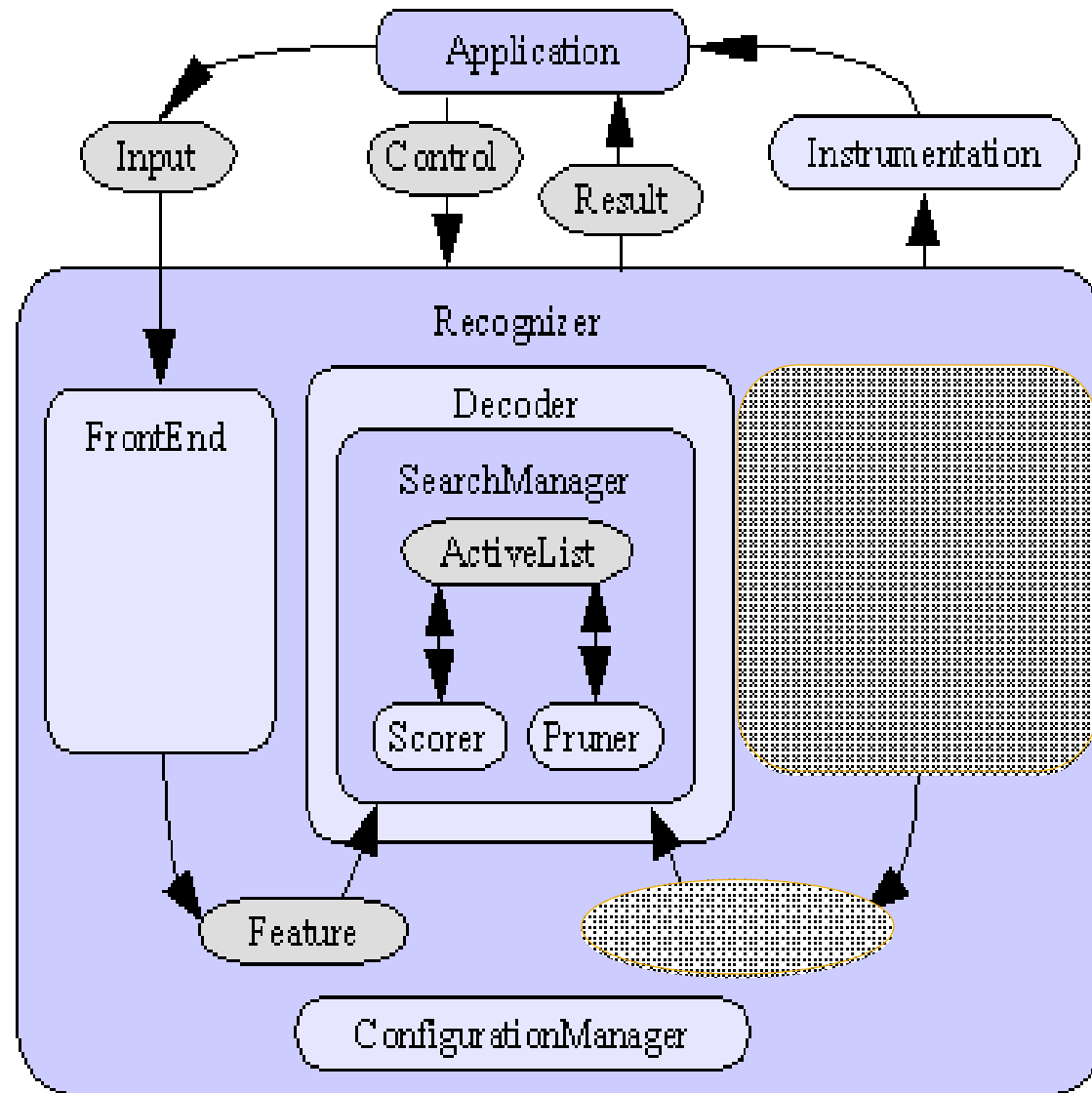
# Sphinx4 implementation



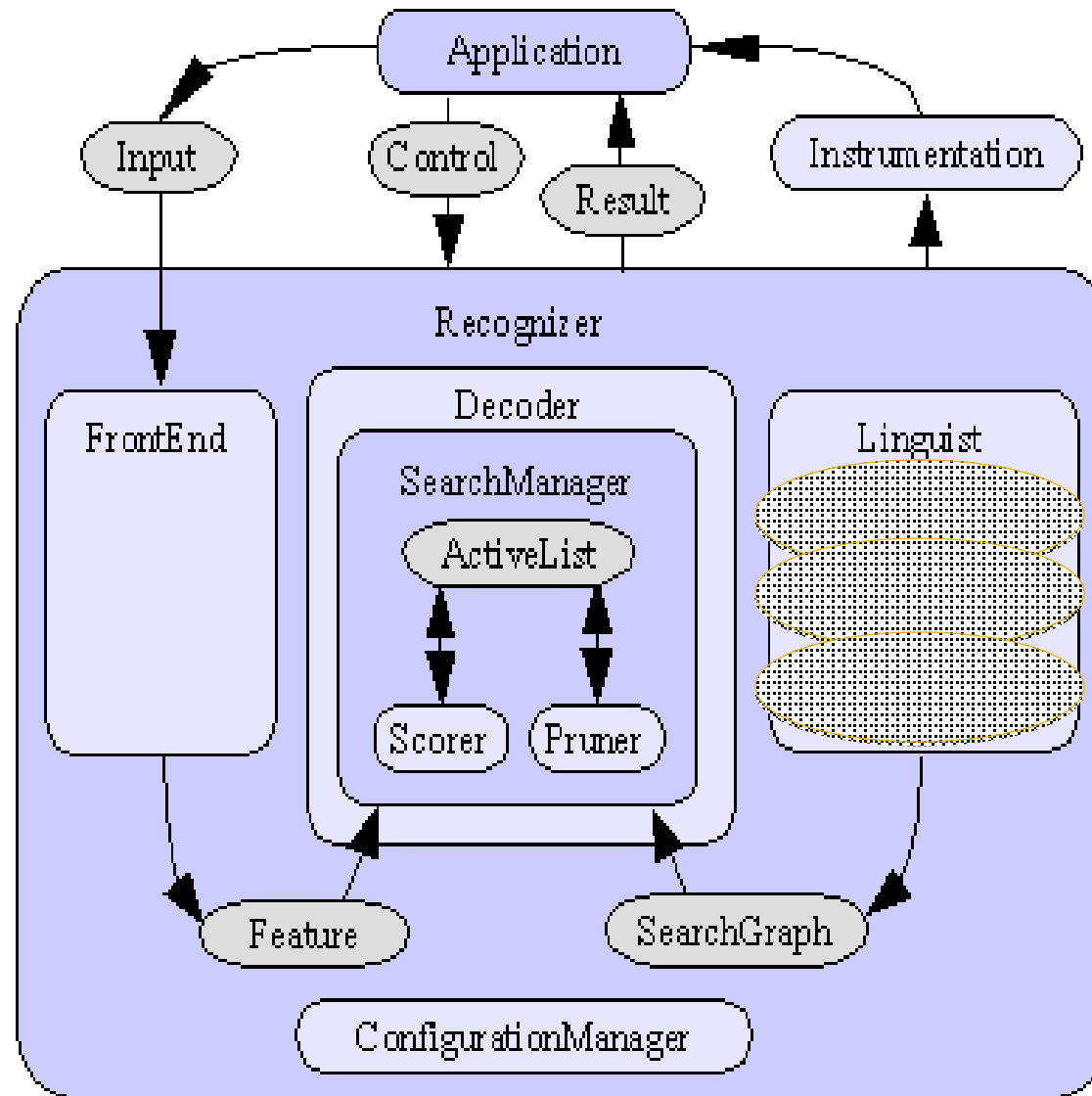
# Sphinx4 implementation



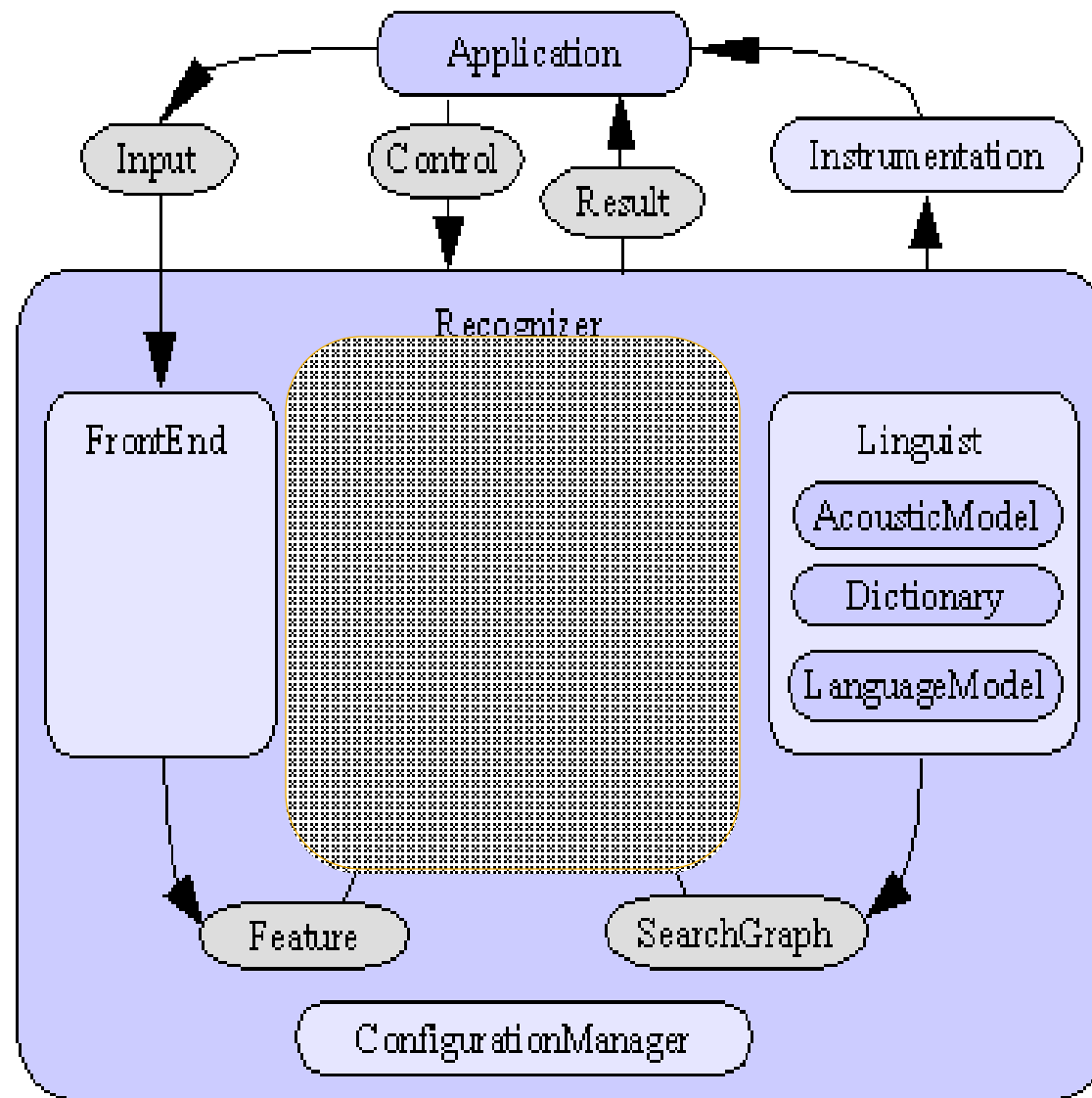
# Sphinx4 implementation



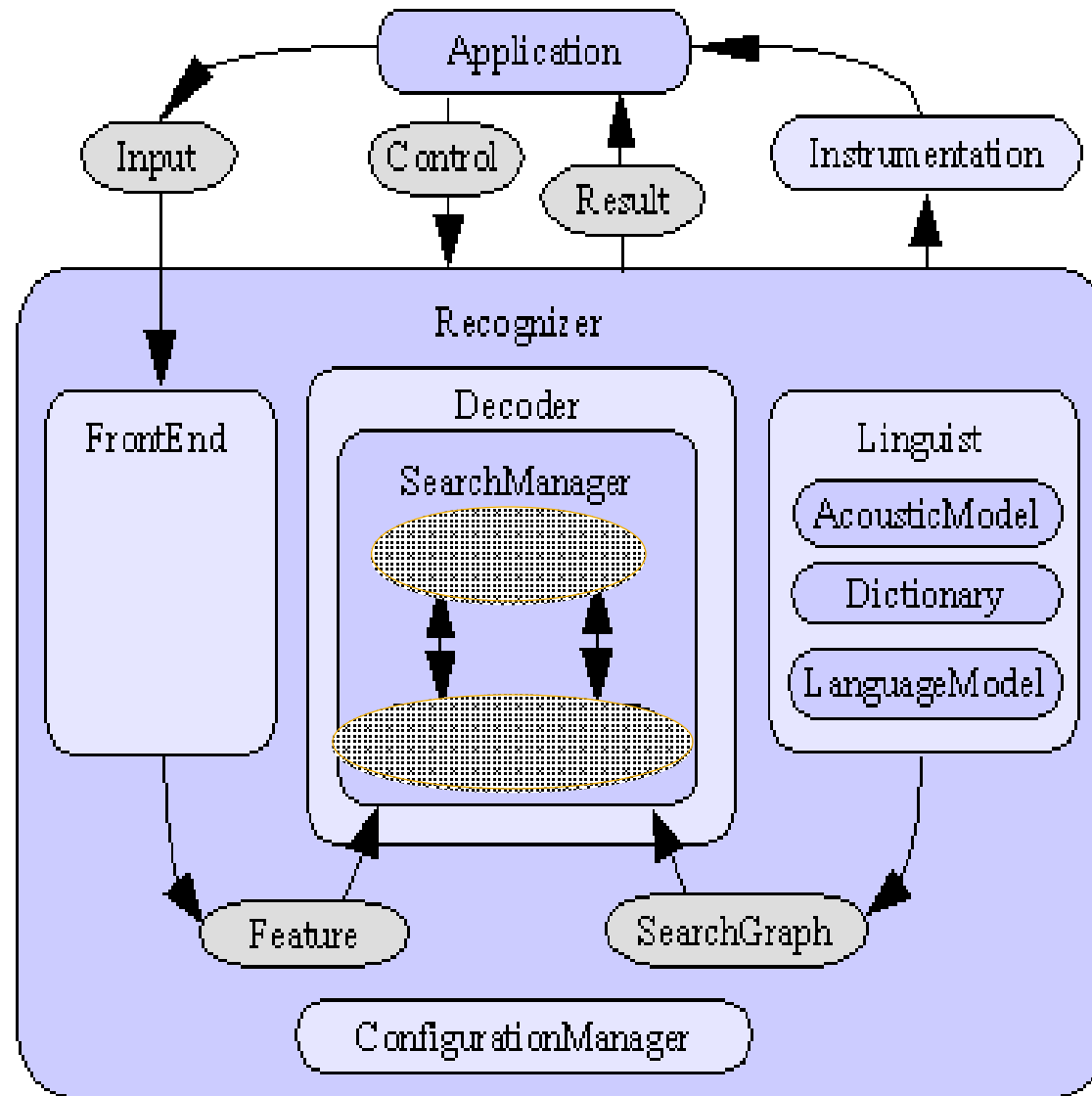
# Sphinx4 implementation



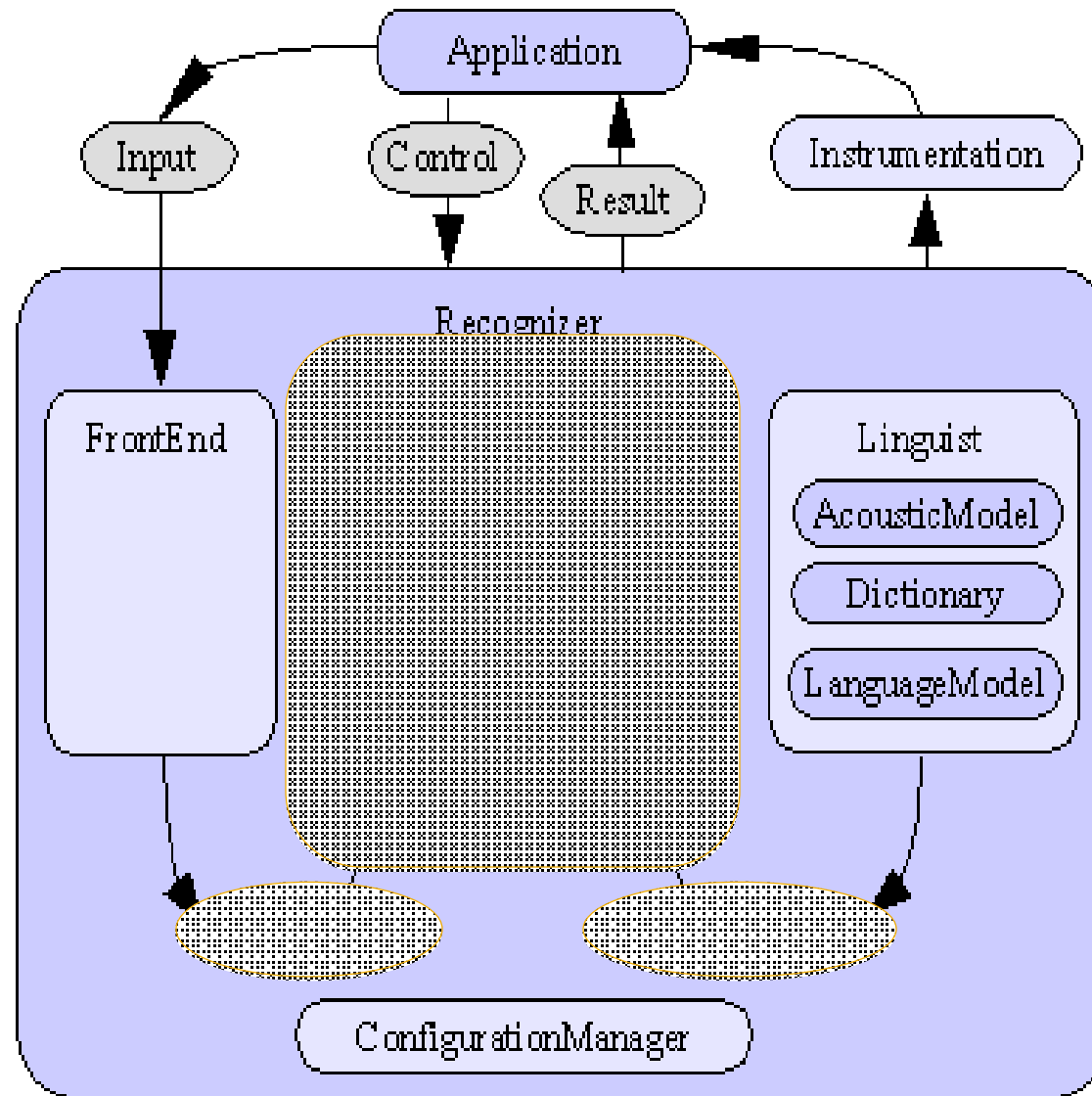
# Sphinx4 implementation



# Sphinx4 implementation

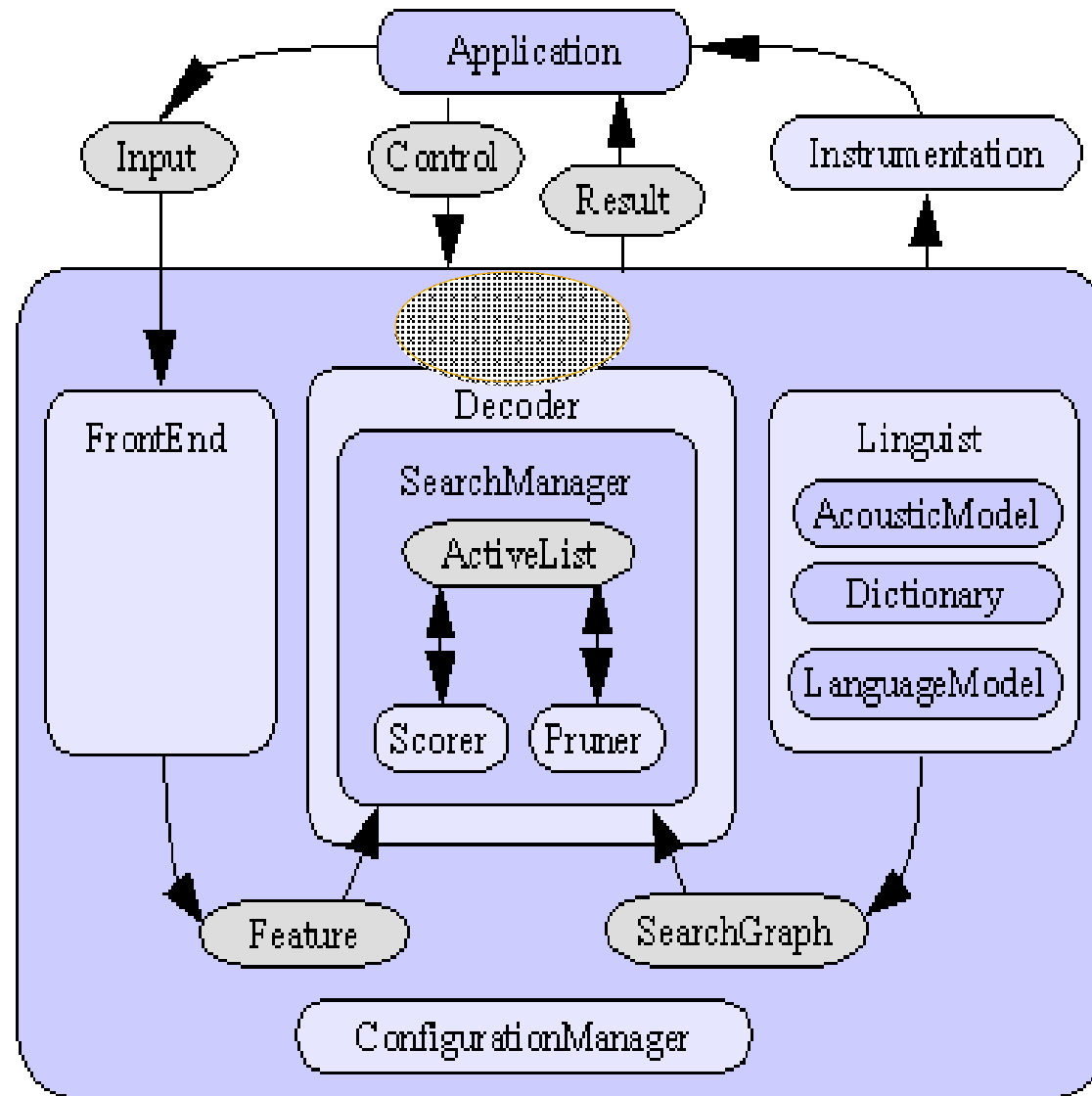


# Sphinx4 implementation

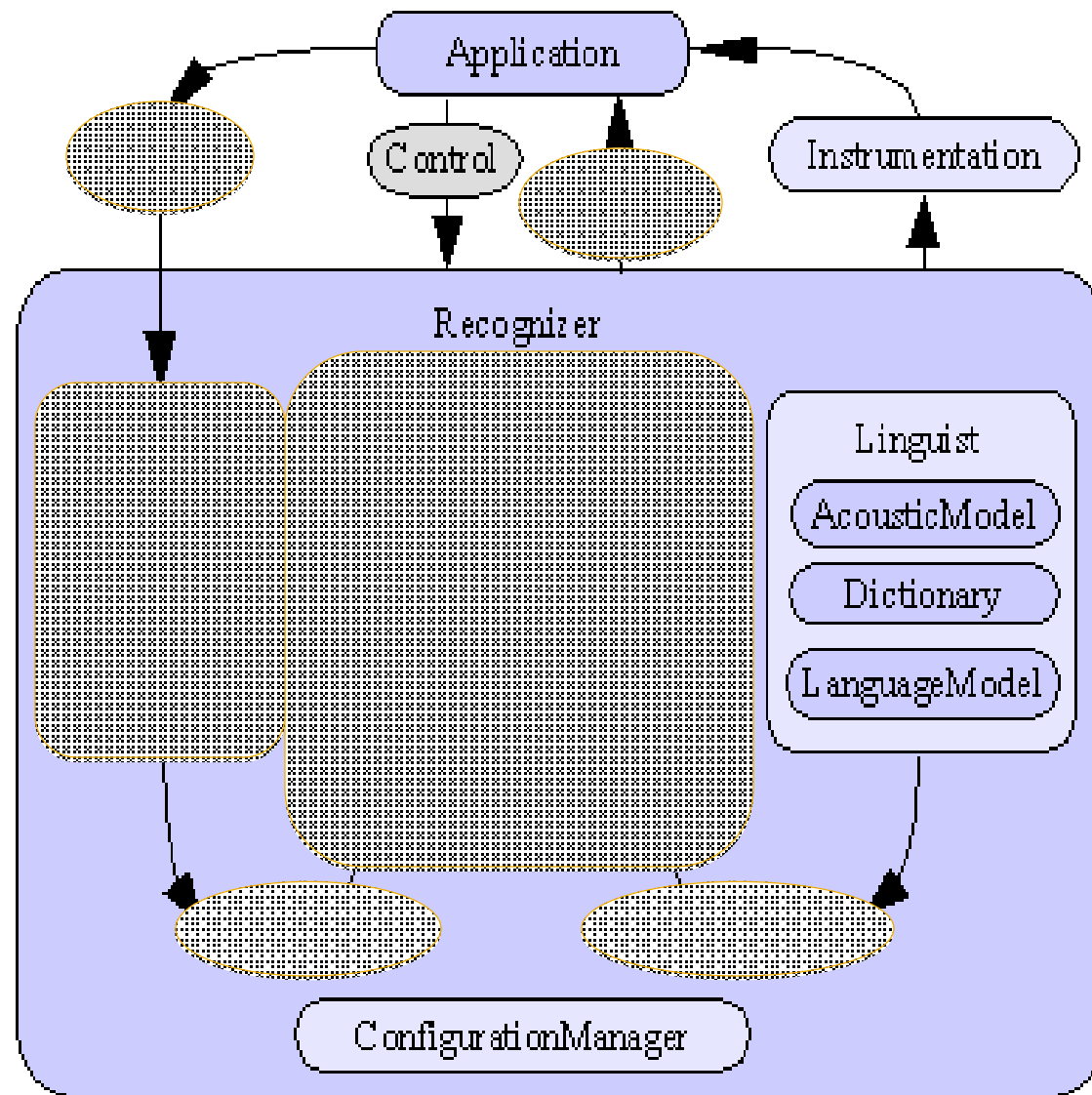




# Sphinx4 implementation



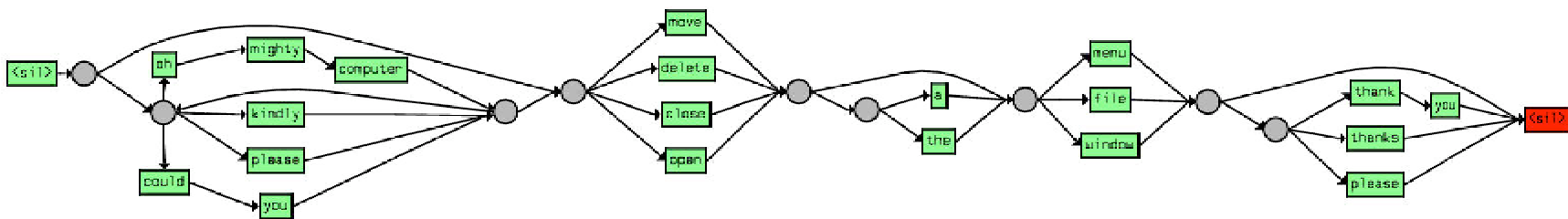
# Sphinx4 implementation



- Q. Is a search graph created for every recognition result or one for the recognition app?
- A. This depends on which Linguist is used. The flat linguist generates the entire search graph and holds it in memory. It is only useful for small vocab recognition tasks. The lexTreeLinguist dynamically generates search states allowing it to handle very large vocabularies

- Q. How does the Viterbi algorithm save computation over exhaustive search?
- A. The Viterbi algorithm saves memory and computation by reusing subproblems already solved within the larger solution. In this way probability calculations which repeat in different paths through the search graph do not get calculated multiple times
- Viterbi cost =  $n^2 - n^3$
- Exhaustive search cost =  $2^n - 3^n$

- Q. Does the linguist use a grammar to construct the search graph if it is available?
- A. Yes, a grammar graph is created



- Q. What algorithm does the Pruner use?
- A. Sphinx4 uses absolute and relative beam pruning

# Sphinx4 Configuration parameters

- Absolute Beam Width - # active search paths  
<property name="absoluteBeamWidth" value="5000"/>
- Relative Beam Width - probability threshold  
<property name="relativeBeamWidth" value="1E-120"/>
- Word Insertion Probability - Word break likelihood  
<property name="wordInsertionProbability" value="0.7"/>
- Language Weight - Boosts language model scores  
<property name="languageWeight" value="10.5"/>
- Silence Insertion Probability - Likelihood of inserting silence  
<property name="silenceInsertionProbability" value=".1"/>
- Filler Insertion Probability - Likelihood of inserting filler words  
<property name="fillerInsertionProbability" value="1E-10"/>

## Sphinx4 Configuration parameters

- Silence Insertion Probability - Likelihood of inserting silence
- `<property name="silenceInsertionProbability" value=".1"/>`
- Filler Insertion Probability - Likelihood of inserting filler words
- `<property name="fillerInsertionProbability" value="1E-10"/>`



## Sphinx4 resources & references

- Common Sphinx4 FAQs can be found online:  
<http://cmusphinx.sourceforge.net/sphinx4/doc/Sphinx4-faq.html>
- Speech and Language Processing 2<sup>nd</sup> Ed.  
Daniel Jurafsky and James Martin  
Pearson, 2009
- Artificial Intelligence 6<sup>th</sup> Ed.  
George Luger  
Addison Wesley, 2009
- Sphinx Whitepaper  
<http://cmusphinx.sourceforge.net/sphinx4/#whitepaper>
- Sphinx Forum  
<https://sourceforge.net/projects/cmusphinx/forums>

## Remaining problems

- **Robustness** - graceful degradation, not catastrophic failure
- **Portability** - independence of computing platform
- **Adaptability** - to changing conditions (different mic, background noise, new speaker, new task domain, new language even)
- **Language Modelling** - is there a role for linguistics in improving the language models?
- **Confidence Measures** - better methods to evaluate the absolute correctness of hypotheses.
- **Out-of-Vocabulary (OOV) Words** - Systems must have some method of detecting OOV words, and dealing with them in a sensible way.
- **Spontaneous Speech** - disfluencies (filled pauses, false starts, hesitations, ungrammatical constructions etc) remain a problem.
- **Prosody** - Stress, intonation, and rhythm convey important information for word recognition and the user's intentions (e.g., sarcasm, anger)
- **Accent, dialect and mixed language** - non-native speech is a huge problem, especially where code-switching is commonplace